Modern Computational Approaches to Nonlinear Discrete Optimization and Applications in Process Systems Engineering

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

the degree of

DOCTOR OF PHILOSOPHY

in

CHEMICAL ENGINEERING

DAVID E. BERNAL NEIRA

B.S., CHEMICAL ENGINEERING CUM LAUDE, UNIVERSIDAD DE LOS ANDES

B.S., PHYSICS, UNIVERSIDAD DE LOS ANDES

M.S., CHEMICAL ENGINEERING, UNIVERSIDAD DE LOS ANDES

CARNEGIE MELLON UNIVERSITY

Pittsburgh, Pennsylvania

May, 2021

Copyright © 2021, David E. Bernal Neira

All rights reserved

Acknowledgments

This has been a fantastic journey. I have only been able to enjoy it thanks to the wonderful individuals who have helped and accompanied me along the way. The first person that I need to thank is Ignacio E. Grossmann for his mentorship during these years. Leading by example, he proves that the path for success is determined by discipline, hard work, and politeness. He inspires all around him, myself included, to be better every day. I sincerely appreciate having the privilege of being advised by you, and I aspire to become as good as a person and researcher as you are.

I also want to thank the members of my doctoral committee, Larry Biegler, Nick Sahinidis, John Hooker, Sridhar Tayur, and Egon Balas†. I have had the pleasure of interacting with them during my time at Carnegie Mellon University, and I hope that after this step, I can continue the very enriching conversations that we have had. I will keep in my memories the discussions with Larry on nonlinear optimization over a beer during the happy hours, the advice and guidance that Nick provided me during my job search process, and the ethical discussion with John after a linear programming lecture. I want to particularly thank Sridhar, whose support and belief in me allowed me to fulfill a long-life dream of working at NASA and whose vision and perspectives on life will be guiding me for the days to come. Finally, I would like to thank Egon for trusting me as his Teaching Assistant and book reviewer and providing me with the beautiful chance to learn and work with a living legend; you will be greatly missed.

I would also like to thank many mentors that I have encountered during my journey. My undergraduate and master's advisor Jorge Mario Gómez, who set me on the right track towards my Ph.D. at CMU and encouraged me to finish my undergraduate in Physics (which later proved to be really useful!), will always have my gratitude. Samewise thanks to Stefan Vigerske and Francisco Trespalacios, who were my guides during my first research project at GAMS, which later opened the doors to my Ph.D. research. Linlin Yang, Nagore Sabio, and Dimitri Papageorgiou, who mentored me during my first internship at ExxonMobil Research and Engineering, will always hold a special place in my heart, given their support. Davide Venturelli, Raouf Dridi, Stuart Hadfield, and Eleanor Rieffel gave me the chance and tools to work at NASA Quantum Artificial Intelligence Laboratory. This experience

opened a new research path for me and a whole life of new opportunities. Luis Zuluaga gave me the chance to collaborate outside of CMU and be part of the Quantum Computing and Optimization Lab at Lehigh University. Lastly, I will always appreciate the trust that Stuart Harwood and Dimitar Trenev, who advised me at my second time at ExxonMobil, deposited on me while working with them entirely remotely amid a global pandemic.

My gratitude also goes to my collaborators, without whom I would not have achieved even a fraction of this work. To Jan Kronqvist, Can Li, Zedong Peng, Qi Chen, Cristiana Lara, Lijie Su, Luis Ricardez-Sandoval, Andreas Lundell, Tapio Westerlund, David Liñán, Kyle Booth, Hedayat Alghassi, Rodolfo Quintero, Claudio Gambella, Andrea Simonetto, Kevin Furman, and Faramroze Engineer: Thank you! I would also like to acknowledge Lixin Tang and Aldo Vecchietti. They hosted me at their home institutions and gave me the chance to interact with your amazing groups at Shenyang University and INGAR, respectively. Moreover, the thanks extend to those that I have been lucky enough to advise during my time at CMU, including Haokun Yang, Yunshan Liu, Rahul Joglekar, Saeed Syed, Felicity Gong, Justin Yuliu, Carolina Carrillo, and Daniel Ovalle.

This list would not be complete without those members of the larger Process Systems Engineering and Optimization communities at CMU. This group of Ph.D. and M.Sc. students, postdoctoral appointees, international visitors, and faculty made up the best environment for professional and personal growth that I could have wished for. In particular, it has been an honor to belong to the Grossmann research group and the Center of Advanced Process Decision Making (CAPD) at the CMU Department of Chemical Engineering.

Many names are (unintentionally) left out of these acknowledgments, and several of the groups mentioned above overlap. At the same time, all these people belong to a single larger group, to which I owe everything: my friends. Special thanks to my friends at Bogotá, who would keep me in their thoughts regardless of the distance, and Pittsburgh, whose company would help me enjoy life beyond work while at CMU. I am grateful to you.

I acknowledge funding from Carnegie Mellon University through the CAPD and from the US Department of Energy, Office of Fossil Energy's Crosscutting Research, Simulation-Based Engineering Program through the Institute for the Design of Advanced Energy Systems (IDAES). Funding for the work on Chapter 9 was provided by the Defense Advanced Research Projects Agency (DARPA), ONISQ grant W911NF2010022, titled *The Quantum Computing Revolution and Optimization: Challenges and Opportunities*. The project was also supported by the Oak Ridge National Laboratory OLCF grant ENG121, which provided us with in-kind access to D-Wave's quantum annealers. Moreover, I acknowledge the Univer-

sities Space Research Association, Amazon Web Services, and National Science Foundation support through the NSF 20-073 grant for enabling quantum computing platform access for funding and support during the design and delivery phase of the Quantum Integer Programming (QuIP) course during the Fall 2020 semester at CMU. Thank you to all those that have taken the course so far, not only officially at CMU and IIT-Madras, but also independently.

I would like to acknowledge the time and company of Paula Sarmiento during several years of this journey. Finally, I would not have been able to get this far without the unending love and unconditional support from my family, both extended and immediate; although always close.

To Martha, Oscar, Samuel, and Laura: this is for you.

David E. Bernal Neira
Pittsburgh, PA, USA
April 2021

Agradecimientos

Este ha sido un viaje fantástico, el cual solo he sido capaz de disfrutar gracias a todos esos individuos que me han ayudado y acompañado en este camino. A la primera persona que debo agradecer es a Ignacio E. Grossmann por su mentoría a lo largo de estos años. Siendo un referente y guía por medio de ejemplo, él demuestra que el camino para el éxito está determinado por la disciplina, el trabajo duro y la decencia. Él nos inspira a todos a su alrededor, incluyéndome, a ser mejores cada día. Sinceramente aprecio haber tenido el privilegio de que hayas sido mi asesor y aspiro a que algún día pueda ser tan buen ser humano e investigador como tú lo eres.

También quiero agradecer a los miembros de mi comité de doctorado, Larry Biegler, Nick Sahinidis, John Hooker, Sridhar Tayur y Egon Balas. He tendio el placer de interactuar con ellos durante mi tiempo en Carnegie Mellon University (CMU) y espero que luego de haber tomado este paso, podemos continuar nuestras muy enriquecedoras conversaciones. Siempre recordaré las discusiones con Larry acerca de optimización no lineal durante los happy hour del departamento con una cerveza en mano, los consejos y guías de Nick durante mi proceso de aplicación a trabajos académicos y las discusiones de ética que tuvimos con John luego de sus clases de optimización lineal. También quiero agradecer particularmente a Sridhar, cuyo apoyo y confianza me han permitido cumplir el sueño de toda una vida de trabajar en la NASA y cuya visión y perspectivas de la vida serán una guía para el resto de mis días. Finalmente, quiero agradecerle a Egon por confirmar en mi como su asistente de docencia y revisor de su libro y así darme la gran oportunidad de aprender y trabajar con una leyenda; que falta nos haces a todos.

También quiero aprovechar para agradecerle a los muchos mentores que he encontrado en el camino. A mi asesor de pregrado y maestría, Jorge Mario Gómez, quien me puso en el camino correcto para comenzar mi doctorado en CMU y me impulsó a terminar mi carrera de Física (¡la cual al final resultó ser muy útil!), muchísimas gracias. De la misma forma a Stefan Vigerske y Francisco Trespalacios, quienes fueron mis guías durante mi primer proyecto de investigación en GAMS, el cual terminó abriendo las puertas a mi investigación de doctorado. Linlin Yang, Nagore Sabio y Dimitri Papageorgiou, quienes fueron mis mentores durante mi primera práctica en ExxonMobil, siempre ocuparán un espacio especial

en mi corazón. Davide Venturelli, Raouf Dridi, Stuart Hadfield y Eleanor Rieffel me dieron la oportunidad y herrramientas para trabajar en el laboratorio de Cuántica e Inteligencia Artificial de la NASA. Esta experiencia abrió un nuevo camino de investigación para mi y una vida entera de nueva oportunidades. Luis Zuluaga me dio la oportunidad de colaborar por fuera de CMU y ser parte del Laboratorio de Computación Cuántica y Optimización de la Universidad de Lehigh. Por último, siempre voy a apreciar la confianza que Stuart Harwood y Dimitar Trenev, quienes fueron mis mentores en la segunda oportunidad que tuve en ExoonMobil, tuvieron en mi al trabajar en un proyecto de manera completamente virtual y en medio de una pandemia global.

Mi gratitud también va a mis colaboradores, sin los cuales no hubiese completado siquiera una fracción de este trabajo. A Jan Kronqvist, Can Li, Zedong Peng, Qi Chen, Cristiana Lara, Lijie Su, Luis Ricardez-Sandoval, Andreas Lundell, Tapio Westerlund, David Liñán, Kyle Booth, Hedayat Alghassi, Rodolfo Quintero, Claudio Gambella, Andrea Simonetto, Kevin Furman y Faramroze Engineer: ¡Gracias! También quiero reconocer a Lixin Tang y Aldo Vecchietti. Ellos me acogieron en sus instituciones y me dieron la oportunidad de interactuar con los maravillosos grupos de la Univesidad de Shenyang y del INGAR, respectivamente. Además, los agradecimientos se extienden a aquellos que tuve la oportunidad de ayudar a dirigir durante mi tiempo en CMU, incluyendo a Haokun Yang, Yunshan Liu, Rahul Joglekar, Saeed Syed, Felicity Gong, Justin Yuliu, Carolina Carrillo y Daniel Ovalle.

Esta lista no estaría completa sin aquellos miembros de las comunidades de Ingeniería de Procesos y Optimización de CMU. Este grupo de estudiantes de maestría y doctorado, post-doctores, visitantes internacionales y profesores componen el mejor ambiente para desarrollo profesional y personal que pude haber deseado. En particular, para mi ha sido un honor pertenecer al grupo de investigación Grossmann y al Center of Advanced Process Decision Making (CAPD) del departamento de Ingeniería Química de CMU.

He dejado muchos nombres por fuera (sin hacerlo intencionalmente) de estos agradecimientos y varios de estos grupos se sobrelapan. Al mismo tiempo, todas estas personas pertenecen a un solo gran grupo, al cual le debo todo: mis amigos. Gracias especiales a mis amigos en Bogotá, quienes siempre me mantuvieron en sus pensamientos a pesar de la distancia, y en Pittsburgh, cuya compañía me daba motivos para disfrutar la vida más allá del trabajo en CMU. Le agradezco.

Reconozco financiamiento de Carnegie Mellon University a través del CAPD y del Departamento de Energía de los Estados Unidos de América, la oficina de investigación en energía fósil y el programa de ingeniería basada en simulación a través del Instituto de

Diseño de Sistemas de Energía Avanzados (IDAES). Financiamiento para el trabajo en el capítulo 9 fue dado por la Agencia de Proyectos de Investiagación Avanzada para Defensa (DARPA), convocatoria ONISQ W911NF2010022, titulada *La revolución de computación cuántica y optimización: Retos y oportunidades*. El proyecto también estuvo apoyado el el Laboratorio Nacional de Oak Ridge por medio de la convocatoria OLCF ENG121, dándonos acceso al equipo cuántico D-Wave. Además, reconozco el apoyo de la Asociación de Investigación Universitaria del Espacio (USRA), Amazon Web Services y la Academia Nacional de Ciencias (NSF) a través de la convocatoria 20-073 para acceso a plataformas de computación cuántica por al apoyo y financiamiento durante las etapas de diseño y presentación del curso de Programación Entera Cuántica (Quantum Integer Programming) durante el semestre de otoño de 2020 en CMU. Gracias a todos aquellos que han tomado dicho curso hasta ahora, no solamente oficialmente en CMU y IIT-Madras, pero también independientemente.

Quisiera reconocer el tiempo y compañía de Paula Sarmiento durante varios años de este viaje. Finalmente, no hubiese podido llegar hasta este punto sin el amor y apoyo incondicional de mi familia, tanto extendida como inmediata; aunque siempre cercana.

A Martha, Oscar, Samuel y Laura: Esto es para ustedes.

David E. Bernal Neira
Pittsburgh, PA, EE.UU.
Abril 2021

Abstract

Nonlinear discrete optimization problems arise in many different disciplines, given the modeling versatility associated with nonlinear constraints and discrete decision variables. In Process Systems Engineering, such problems appear in applications ranging from optimal process design and synthesis, process planning, scheduling, and control, and molecular design. Albeit their many applications that arise from its universal modeling capabilities, finding optimal solutions to these optimization problems is a challenging task, given the computational complexity associated with their solution. The design of novel algorithms and the correct modeling of these problems arise among the different ways to overcome this complexity. In particular, tackling these problems with the correct combination of mathematical modeling and solution procedure is an efficient strategy to address them. The objective of this Thesis is to propose new solutions and modeling methods for nonlinear discrete optimization problems, which lead to improvements with respect to the existing solution approaches.

We initially pose the discrete nonlinear optimization problems in the context of Mathematical Programming. The problems that we consider solving here can be classified as Mixed-Integer Nonlinear Programming (MINLP) problems. In Chapter 2 we provide a review on the different solution algorithms and existing software to deterministically solve a subclass of MINLP problems called convex MINLP. Among those algorithms, we consider the Outer-approximation (OA) method, which decomposes the MINLP into a Mixed-Integer Nonlinear Programming (MILP) problem and a Nonlinear Programming (NLP) problem. We perform a large computational study comparing the performance of

more than sixteen different software implementations, *solvers*, by solving over 350 convex MINLP problems from benchmark library MINLPLib. This large study allowed us to identify how the different solvers perform based on features from the problem to be solved.

Chapter 3 presents the implementation of the feasibility pump algorithm in the commercial MINLP solver DICOPT. This algorithm is being used as a preprocessing step to enhance the solver's capabilities to find feasible solutions early in the search for the optimal solutions. The approach described and implemented in this chapter improved the solver performance becoming the default setting for DICOPT when solving convex MINLP problems.

In Chapter 4 we propose a new algorithm for convex MINLP, the Center-cut algorithm. Using a decomposition of the problem similar to OA, this algorithm relies on finding the Chebyshev center of the linear approximation of the nonlinear constraints. Although this algorithm is deterministic, in the sense that we provide convergence guarantees for it, it behaved remarkably well in finding feasible solutions quickly.

Chapter 5 presents the derivation of scaled quadratic underestimators for convex functions and their usage in an OA framework, denoted Outer-approximation with quadratic cuts (OA-QCUT). Using those quadratic underestimators, the decomposition in OA then requires the solution of a Mixed-Integer Quadratically Constrained Programming (MIQCP) problem, which more closely underestimates the nonlinearities in convex MINLP problems, achieving a reduction in iterations when solving these problems.

Chapter 6 then presents another modification of OA, where auxiliary Mixed-Integer Quadratic Programming (MIQP) problems are solved at each iteration of OA. These auxiliary MIQP problems minimize a quadratic distance metric to the best-found solution in the algorithm while guaranteeing an improvement in the estimated objective function, hence stabilizing the OA method. These methods are successful at reducing the total number of iterations in OA at the expense of the solution of the auxiliary problem, which we prove

needs not be solved to optimality, leading to performance improvements of the OA method when solving convex MINLP.

Chapter 7 generalizes the concepts of Chapter 6 by showing that the auxiliary mixed-integer problem can have any regularization objective function. We prove the convergence guarantees of this method and show its equivalence of integrating a trust-region constraint in OA. This method is denoted Regularized Outer-approximation (ROA). We implemented these algorithms as part of the open-source **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy and tested them extensively with all convex MINLP problems in the library MINLPLib. The results suggest an improvement of the existing OA and LP/NLP methods by using regularization.

Chapter 8 tackles the more efficient solution of convex MINLP problems from a different perspective, its modeling. Having identified that one of the primary sources of convex MINLP problems is problems that enforce nonlinear constraints given a discrete choice, we consider a higher level modeling alternative known as Generalized Disjunctive Programming (GDP). The GDP modeling framework uses logical variables and disjunctions to represent these nonlinear discrete optimization problems, which later can be transformed into MINLP problems via reformulations. One of the reformulations is the Hull reformulation (HR), which derives a higher dimensional description of the disjunctive set whose projection into the space of the original variables yields its convex hull. For GDP problems with convex constraints, the HR requires implementing the perspective of the convex functions. This function is non-differentiable at zero, leading to computational challenging MINLP problems and motivating approximation schemes to this problem. We derive a new representation of these problems by modeling convex constraints in GDP problems using conic sets. The reformulation of these problems results in a Mixed-Integer Conic Programming (MICP) problem, which can be efficiently tackled using solvers that take

advantage of the conic structure of the problems. In particular, the HR of these conic GDP problems can be described exactly using conic inequalities allowing the use of these tight problem formulations while avoiding the approximation of the perspective function. We performed a large computational study with over 400 convex GDP problems, of which 200 were derived from Process Systems Engineering and Machine Learning applications. Our results indicate how the proper modeling of the disjunctive constraints allows for specialized algorithms to solve them, hence leading to performance improvements.

Chapter 9 also considers the formulation of constrained optimization problems for better solution performance. Contrary to the previous chapters, the solution methods intended to be used here rely on quantum computing. Discrete optimization using quantum computing requires a reformulation into Quadratic Unconstrained Binary Optimization (QUBO) problems. This chapter considers the Maximum k Coloring Subgraph (MkCS) problem, a problem arising from Graph Theory and with applications in science and engineering. We propose a nonlinear formulation of this problem involving bilinear equality constraints, whose QUBO reformulation is more amenable to quantum algorithms. We show the difference between the formulation through a large computational study involving the reformulation and solution of these problems using Quantum Annealing. The nonlinear formulation behaved better than the linear formulation in terms of the QUBO problem size, both in terms of the original problem and its *embedded* version for the existing Quantum Annealing hardware, and in solution time, proving an advantage of the proposed nonlinear formulation.

Chapter 10 considers the Vehicle Routing Problem with Time Windows (VRPTW) and its QUBO reformulation for solving it via quantum algorithms. We study three different formulations of the discrete problem, a route-based, a sequence-based, and an arc-based formulation, and compare them in terms of metrics relevant to the solution of the problem

through QUBO methods. The route-based formulation was amenable to preprocessing using classical computing methods. In contrast, the sequence- and arc-based formulations showed better asymptotic behavior in terms of problem size when considering larger instances, motivating a combination of the formulations to take better advantage of quantum computing algorithms to solve these routing problems. We implement and simulate the solution via quantum algorithms of the sequence-based formulation of an instance inspired by the Maritime Inventory Routing Problem (MIRP), which is small enough to simulate classically and large enough to make it practically hard to perform a complete enumeration of its solutions. This study allowed us to observe a trade-off between different quantum algorithms implementable in existing gate-based quantum computers. The Variational Quantum Eigensolver (VQE) was preferable when the access to the quantum computer is limited, and the Quantum Approximate Optimization Algorithm (QAOA) was favored when more samples from the quantum circuit can be obtained. Moreover, we consider a continuous version of the sequence-based formulation and apply an Alternating Direction Method of Multipliers (ADMM) decomposition heuristic method relying on QUBO subproblem solution, showing the potential of decomposition methods based on QUBO solving as an interesting future research avenue.

Finally, we conclude with Chapter A in the Appendix, where we formulate the minor-embedding problem as an Integer Programming (IP) problem and solve it directly using existing solvers and through a tailored decomposition algorithm. This problem appears in the precompilation step required for existing Quantum Annealers to solve arbitrary QUBO problems. This IP formulation allows for solutions to the problems with optimality guarantees, contrary to the heuristic methods used in practice to address this problem, given its complexity.

The results in this Thesis show how the study of the mathematical structure of discrete

nonlinear optimization problems leads to their more efficient solution. We consider incorporating these improvements through problem formulation or solution algorithm design, with a clear focus on decomposition techniques. We anticipate our results to motivate further research in nonlinear discrete optimization, with a strong focus on applications in Process Systems Engineering. Moreover, we show how the techniques traditionally applied to classical solution methods can be adapted to take advantage of unconventional computing approaches, such as quantum computing, to better solve these problems.

Thesis supervisor: Ignacio E. Grossmann Epper

Title: Rudolph R. and Florence Dean University Professor of Chemical Engineering

Resumen

Los problemas de optimización discreta no lineal surgen en muchas disciplinas, dada la versatilidad de modelado asociada con las restricciones no lineales y las variables de decisión discretas. En Ingeniería de Sistemas de Procesos, estos problemas aparecen en aplicaciones que van desde el diseño y síntesis óptimos de procesos, la planeacion y control de procesos y el diseño molecular. A pesar de sus muchas aplicaciones, que surgen a partir de sus capacidades de modelamiento universal, encontrar soluciones óptimas a estos problemas de optimización es una tarea desafiante, dada la complejidad computacional asociada con su solución. El diseño de algoritmos novedosos y el correcto modelado de estos problemas surgen entre las diferentes formas de superar esta complejidad. En particular, abordar estos problemas con la combinación correcta de modelos matemáticos y procedimientos de solución es una estrategia eficaz para abordarlos. El objetivo de esta Tesis es proponer nuevas soluciones y métodos de modelado para problemas de optimización discretos no lineales, que conduzcan a mejoras con respecto a los enfoques de solución existentes.

Inicialmente planteamos los problemas de optimización discreta no lineal en el contexto de la Programación Matemática. Los problemas que consideramos resolver aquí se pueden clasificar como problemas de programación mixto entero no lineal (Mixed-Integer Nonlinear Programming MINLP). En el Capítulo 2 proporcionamos una revisión de los diferentes algoritmos de solución y el software existente para resolver determinísticamente una subclase de problemas MINLP llamados MINLP convexos. Los problemas MINLP convexos tienen la cualidad de que sus restricciones no lineales son convexas, lo que da lugar a algoritmos

de solución eficientes. Entre esos algoritmos, consideramos el método de aproximación externa (Outer Approximation OA), que descompone el MINLP en un problema de programación no lineal de enteros mixtos (Mixed-Integer Linear Programming MILP) y un problema de programación no lineal (Nonlinear Programming NLP). Realizamos un gran estudio computacional comparando el rendimiento de más de dieciséis implementaciones de software diferentes, *solvers*, resolviendo más de 350 problemas MINLP convexos de la biblioteca de referencia MINLPLib. Este gran estudio nos permitió identificar cómo se desempeñan los diferentes solucionadores en función de las características del problema a resolver.

El Capítulo 3 presenta la implementación del algoritmo de bomba de factibilidad en el solver comercial para MINLP, DICOPT. Este algoritmo se utiliza como un paso de preprocesamiento para mejorar las capacidades del solver encontrando soluciones factibles al principio de la búsqueda de las soluciones óptimas. El enfoque descrito e implementado en este capítulo mejoró el rendimiento del solver convirtiéndose en la configuración predeterminada para DICOPT al resolver problemas de MINLP convexos.

En el Capítulo 4 proponemos un nuevo algoritmo para MINLP convexo, el algoritmo de corte central. Usando una descomposición del problema similar a OA, este algoritmo se basa en encontrar el centro de Chebyshev de la aproximación lineal de las restricciones no lineales. Aunque este algoritmo es determinístico, en el sentido de que le damos garantías de convergencia, se comportó notablemente bien en la búsqueda rápida de soluciones factibles.

El Capítulo 5 presenta la derivación de subestimadores cuadráticos escalados para funciones convexas y su uso en el método de OA, denotado por aproximación externa con cortes cuadráticos (OA-QCUT). Usando esos subestimadores cuadráticos, la descomposición en OA luego requiere la solución de un problema de programación mixto entero con restric-

ciones cuadráticas (Mixed-Integer Quadratically Contstrained Programming MIQCP), que subestima más de cerca las no linealidades en los problemas convexos MINLP, logrando una reducción en las iteraciones al resolver estos problemas.

El Capítulo 6 luego presenta otra modificación de OA, donde problemas auxiliares de programación mixta etnera cuadrática (Mixed-Integer Quadratic Programming MIQP) se resuelven en cada iteración de OA. Estos problemas auxiliares MIQP minimizan una métrica de distancia cuadrática a la mejor solución encontrada en el algoritmo al tiempo que garantizan una mejora en la función objetivo estimada, estabilizando así el método OA. Estos métodos tienen éxito en reducir el número total de iteraciones en OA a expensas de la solución del problema auxiliar, que demostramos no necesita ser resuelto de manera óptima, lo que lleva a mejoras en el rendimiento del método OA al resolver MINLP convexos.

El Capítulo 7 generaliza los conceptos del Capítulo 6 mostrando que el problema auxiliar de mixto entero puede tener cualquier función objetivo de regularización. Demostramos las garantías de convergencia de este método y mostramos su equivalencia de integrar una restricción de región de confianza en OA. Este método se denomina Aproximación externa regularizada (ROA). Implementamos estos algoritmos como parte del código abierto **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy y fueron probados extensamente con todos los problemas de MINLP convexos en la biblioteca MINLPLib. Los resultados sugieren una mejora de los métodos de OA y LP/NLP existentes mediante el uso de la regularización.

El Capítulo 8 aborda la solución más eficiente de problemas convexos MINLP desde una perspectiva diferente, su modelado. Habiendo identificado que una de las principales fuentes de problemas convexos del MINLP son los problemas que imponen restricciones no lineales dada una elección discreta, consideramos una alternativa de modelado de nivel superior conocida como Programación Disyuntiva Generalizada (Generalized Disjunctive

Programming GDP). El marco de modelado de GDP utiliza variables lógicas y disyunciones para representar estos problemas de optimización discretos no lineales, que luego pueden transformarse en problemas MINLP mediante reformulaciones. Una de las reformulaciones es la reformulación de casco (Hull Reformulation HR), que deriva una descripción multi-dimensional del conjunto disyuntivo cuya proyección en el espacio de las variables originales produce su casco convexo. Para problemas de GDP con restricciones convexas, la HR requiere implementar la perspectiva de las funciones convexas. Esta función no es diferenciable en cero, lo que genera problemas computacionales desafiantes para MINLP y motiva esquemas de aproximación a este problema. Derivamos una nueva representación de estos problemas modelando restricciones convexas en problemas de GDP usando conjuntos cónicos. La reformulación de estos problemas da como resultado un problema de programación mixta entera cónica (Mixed-Integer Conic Programming MICP), que se puede abordar de manera eficiente utilizando solucionadores que aprovechan la estructura cónica de los problemas. En particular, la HR de estos problemas GDP cónicos se puede describir exactamente utilizando desigualdades cónicas, lo que permite el uso de estas formulaciones evitando la aproximación de la función de perspectiva. Realizamos un gran estudio computacional con más de 400 problemas de GDP convexo, de los cuales 200 se derivaron de aplicaciones de Ingeniería de Sistemas de Procesos y Aprendizaje dem Máquinas (Machine Learning ML). Nuestros resultados indican cómo el modelado adecuado de las restricciones disyuntivas permite que algoritmos especializados las resuelvan, lo que conduce a mejoras en el rendimiento.

El Capítulo 9 también considera la formulación de problemas de optimización restringidos para un mejor rendimiento en su solución. A diferencia de los capítulos anteriores, los métodos de solución que se pretenden utilizar aquí se basan en la computación cuántica. La optimización discreta mediante la computación cuántica requiere una reformulación en

problemas de optimización binaria cuadrática sin restricciones (Quadratic Unconstrained Binary Optimization QUBO). Este capítulo considera el problema del subgrafo de color máximo de k (Maximum k Coloring Subgraph MkCS), un problema que surge de la teoría de grafos y con aplicaciones en ciencia e ingeniería. Proponemos una formulación no lineal de este problema que involucra restricciones de igualdad bilineales, cuya reformulación QUBO es más tratable por algoritmos cuánticos. Mostramos la diferencia entre diferences formulaciones a través de un gran estudio computacional que involucró la reformulación y solución de estos problemas utilizando al algoritmo de temple cuántico (Quantum Annealing). La formulación no lineal se comportó mejor que la formulación lineal en términos del tamaño del problema QUBO, tanto en términos del problema original y su versión *embebida* para el hardware de recocido cuántico existente, y en el tiempo de solución, demostrando una ventaja de la propuesta formulación no lineal.

El Capítulo 10 considera el problema de enrutamiento de vehículos con ventanas de tiempo (Vehicle Routing Problem with Time Windows VRPTW) y su reformulación QUBO para resolverlo mediante algoritmos cuánticos. Estudiamos tres formulaciones diferentes del problema discreto, una formulación basada en ruta, una basada en secuencia y una basada en arco, y las comparamos en términos de métricas relevantes para la solución del problema a través de métodos QUBO. La formulación basada en rutas fue susceptible de preprocesamiento utilizando métodos informáticos clásicos. Por el contrario, las formulaciones basadas en secuencias y arcos mostraron un mejor comportamiento asintótico en términos de tamaño del problema al considerar instancias más grandes, lo que motiva una combinación de las formulaciones para aprovechar mejor los algoritmos de computación cuántica para resolver estos problemas de enrutamiento. Implementamos y simulamos la solución a través de algoritmos cuánticos de la formulación basada en secuencia de una instancia inspirada en el Problema de enrutamiento de inventario marítimo (Maritime

Inventory Routing Problem MIRP), que es lo suficientemente pequeño para simular de manera clásica y lo suficientemente grande como para que sea prácticamente difícil realizar una enumeración completa de sus soluciones. Este estudio nos permitió observar una contrapartida entre diferentes algoritmos cuánticos implementables en computadoras cuánticas existentes basadas en compuertas (gates). El Eigensolver variacional cuántico (Variational Quantum Eigensolver VQE) es ventajoso cuando el acceso a la computadora cuántica es limitado y el algoritmo de optimización aproximada cuántica (Quantum Approximate Optimization Algorithm QAOA) fue favorable cuando se pueden obtener más muestras del circuito cuántico. Además, consideramos una versión continua de la formulación basada en secuencia y aplicamos un método heurístico de descomposición, el Método de dirección alterna de multiplicadores (Alternating Direction Method of Multipliers ADMM) que se basa en la solución del subproblema QUBO, mostrando el potencial de los métodos de descomposición basados en la resolución de QUBO como una interesante avenida de investigación futura.

Finalmente, concluimos con el Capítulo A en el Apéndice, donde formulamos el problema de embedido (Embedding) como un problema de Programación Entera (Integer Programming IP) y lo resolvemos directamente usando solvers existentes y mediante un algoritmo de descomposición especializado. Este problema aparece en el proceso de precompilación requerido por los equipos de temple cuántico (Quantum Annealers) existentes para resolver problemas arbitrarios de QUBO. Esta formulación de IP permite solucionar los problemas con garantías de optimalidad, contrario a los métodos heurísticos utilizados en la práctica para abordar este problema, dada su complejidad.

Los resultados de esta Tesis muestran cómo el estudio de la estructura matemática de problemas de optimización discretos no lineales conduce a su solución más eficiente. Consideramos incorporar estas mejoras mediante la formulación de problemas o el diseño de

algoritmos de solución, con un claro enfoque en las técnicas de descomposición. Anticipamos que nuestros resultados motiven más investigaciones en optimización discreta no lineal, con un fuerte enfoque en aplicaciones en Ingeniería de Sistemas de Procesos. Además, mostramos cómo las técnicas aplicadas tradicionalmente a los métodos de solución clásicos se pueden adaptar para aprovechar los enfoques computacionales no convencionales, como la computación cuántica, para resolver mejor estos problemas.

Director de tesis: Ignacio E. Grossmann Epper

Título: Rudolph R. and Florence Dean University Professor of Chemical Engineering

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, we elaborate on the main challenges addressed by this work, describe the literature relevant to nonlinear discrete optimization modeling and solution methods, and summarize the remaining chapters of the dissertation.

1.1 Main Challenges and Goals

This dissertation covers several algorithmic and formulation approaches to discrete nonlinear optimization problems. In terms of the algorithmic approaches to address these challenging optimization problems, we rely on the concept of problem decomposition. Loosely speaking, the fact that the problems in consideration belong to the NP-Hard class means that for a given problem size S, it might be computationally more efficient to solve n subproblems of size S/n repeatedly than to solve the original problem once. This performance advantage can be exploited further when the subproblems have a mathematical structure that can be exploited to achieve faster solutions. This decomposition approach and iterative subproblem solution appear in all the methods proposed herein. Whether when considering the decomposition of nonlinear discrete problems in linear discrete and nonlinear continuous subproblems or when separating a quadratic unconstrained binary optimization, the problem in the iterative sampling of a quantum system to determine the values of the discrete variables connected to a classical continuous optimization procedure to maximize the objective function.

The challenge when solving these problems is choosing how this decomposition will be performed, particularly considering that the resulting subproblems have to be solved efficiently, besides guaranteeing or at least directing its convergence to the optimal solution of the problem. Moreover, the formulation of the discrete nonlinear optimization problems allows the problem structure to be more exposed to the solution methods, yielding better performance. The correct match between model formulation and solution method allows tackling these optimization problems practically.

This work aims to develop new techniques and tools to address these challenges. Specific goals of this Thesis are as follows:

1. Present a solver benchmark for convex MINLP problems, including recommendations based on problem characteristics for most performant solution technique.
2. Implement the feasibility pump algorithm in the commercial solver DICOPT, enhancing the solver's capabilities to address convex MINLP problems.
3. Propose the Center-cut algorithm, a deterministic solution method for convex MINLP problems with good performance at finding a feasible solution to these problems.
4. Extend the Outer-approximation method for convex MINLP problems to consider scaled quadratic cuts to define a Mixed-Integer Quadratically Constrained Programming (MIQCP) problem as the mater problem of this method.
5. Propose the use of auxiliary Mixed-Integer regularization subproblems to enhance the stability of the Outer-approximation method for convex MINLP, ultimately leading to improving time and iteration performance of the method.
6. Propose a nonlinear formulation of the Maximum k coloring subgraph problem in order for its QUBO formulation to be a better fit to be solved using Quantum Annealing.
7. Evaluate different formulations of the Vehicle Routing Problem with Time Windows

(VRPTW) to be more efficiently solved via variational algorithms on quantum computers.

1.2 Background

The objective of this Thesis is to present solution methods to nonlinear discrete optimization problems. These optimization problems are of particular interest to the Process Systems Engineering (PSE) discipline, which is defined by Pistikopoulos et al. [1] as follows:

> Process Systems Engineering (PSE) is the scientific discipline of integrating scales and components describing the behavior of a physicochemical system via mathematical modeling, data analytics, design, optimization, and control.

Given the inherent nonlinearity of physicochemical systems and the ubiquitous usage of logical conditions in optimization [2] usually through discrete variables, nonlinear discrete optimization arises as the ideal tool to tackle problems in PSE. In order to solve these optimization problems, the paradigm of Mathematical Programming is used. In Mathematical Programming, an optimization problem is written as the maximization or minimization of an objective subject to constraints, all in terms of values that can be modified to search for an optimal solution, *i.e.*, variables. Mathematical Programming problems can be classified by the nature of their objectives, constraints, and variables.

Optimization problems whose objectives and constraints can be represented by algebraic linear and nonlinear functions of both continuous and discrete variables are commonly referred to as Mixed-Integer Nonlinear Programming (MINLP) problems. MINLP is a highly versatile modeling paradigm, allowing even Universal Turing Machines to be encoded via a Minsky's register machine [3]. Its broad modeling capabilties lead to a wide variety of real-world optimization problems that can be modeled as MINLP in PSE [4–6], *e.g.*, process

flowsheet superstructure [7], equipment design [8–10], production planning [11, 12], process scheduling [13, 14], process control [15], and process synthesis [16, 17]. Moreover, there are several applications of MINLP beyond PSE, such as finance and portfolio optimization [18], other branches of engineering [19], cancer treatment planning [20], computational biology [21], and network design [22]. In fact, "most industrial processes can be modeled as MINLP," according to to Liberti [23].

Mixed-Integer Nonlinear Programming problems can be classified as NP-Hard problems [24]. The many applications of this modeling paradigm motivate the study of solution procedures even if they do not accept polynomial-time algorithms. This means that for the remainder of this Thesis, when we denote that we aim to solve these problems efficiently, we mean it in a practical sense. Moreover, the work included in this Thesis will endeavor to find the provably optimal solution to these optimization problems in a reasonable amount of time. Considering this, some of the algorithms proposed in this Thesis will be used as heuristic methods, in the sense that they will be used to generate good quality solutions quickly and to have the algorithms stop without global optimality certificates. This is the case for the feasibility pump and the Center-cut algorithms for convex MINLP problems and quantum annealing and variational quantum optimization for Quadratic Unconstrained Binary Optimization (QUBO) problems. This Thesis covers both algorithms and mathematical representation of the problems, called *formulations*, to efficiently address them. These techniques are applied to produce speedups with respect to the existing solution approaches.

A particular class of MINLP problems is where the constraints are convex functions. Although it is non-convex because of the nature of the discrete variables, this problem is known as convex MINLP [25, 26]. This class of MINLP is a subject of interest given the many applications that it can represent and the challenging algorithmic requirements that

need to be tackled to address their problems. For a review on convex MINLP, refer to Chapter 2 in this Thesis.

Among the solution techniques for convex MINLP, several have been adapted from the Mixed-Integer Linear Programming (MILP), including Branch & Bound [27] and Benders Decomposition [28]. In contrast, others generalize the solutions methods for convex continuous Nonlinear Programming (NLP) problems, such as the Extended Cutting Plane methods [29]. A particularly successful approach to convex MINLP is the outer-approximation (OA) method proposed by Duran and Grossmann [30], where an iterative solution of a convex NLP and an MILP subproblem is performed. The MILP is derived through first-order Taylor approximations, or gradient-based linearizations of the nonlinear constraints at the NLP solutions, and the NLPs arise from the problems appearing when fixing the values of the discrete variables at the MILP solution [25, 30]. Many of the current commercial tools to solve convex MINLP rely on the OA method [26].

In continuous convex programming, solutions methods have also been derived by generalizing Linear Programming (LP) notions and techniques. One of the most successful ones has been the proposal of convex optimization problems as problems defined over cones, or Conic Programming (CP) problems [31]. CP is a numerically stable alternative for convex programming [31], given that it exploits properties of the conic sets. Convex Programming problems described via algebraic convex nonlinear constraints of the form $f(\mathbf{x}) \leq 0$ can be equivalently posed as linear transformation of the variables belonging to convex sets $\mathcal{K}$, *i.e.,* $\mathbf{Ax} - \mathbf{b} \in \mathcal{K}$ [31, 32]. A generalization of CP where some variables are required to take discrete values is Mixed-Integer Conic Programming (MICP). MICP problems are highly expressible and can represent a wide range of optimization problem [33]. Many of these applications have been gathered in the problem library CBLib [34].

The automatic identification and translation of the two equivalent descriptions of convex

sets is a crucial feature for algorithmic solution software, *solvers*, development. This is since the description of problems using algebraic constraints is more natural for practitioners. However, the conic description of the problem allows taking advantage of mathematical properties such as conic duality for more stable solution procedures. Generic solvers have been designed to tackle CP problems, e.g., MOSEK [35], ECOS [36], and Hypatia [37]. This translation is not trivial [38–40]. However, it has been achieved for the quadratic case allowing for solution methods based on conic programming to be used for these problems. An alternative to translating practical optimization problems into CP is via Disciplined Convex Programming (DCP) [41], where strict rules of function definitions guarantee the problem's convexity and perform the translation such that they can be solved through generic conic solvers.

In the mixed-integer setting, solvers have been designed to take as input the MICP problem taking advantage of this form of the optimization problem structure, e.g., Mosek [35], and Pajarito [42–44]. Even for solvers that do not necessarily consider the conic representation of convex problems, identifying such structures leads to improvements in its performance, such as in SCIP [45, 46] and BARON [47]. There is a significant potential for MINLP solvers to perform automatic reformulations once they identify correct structures [48]. An example of the automatic identification of conic structures is Mixed-Integer Quadratically-constrained Quadratic Programming (MIQCQP) problems can now be tackled through Mixed-Integer Second-Order Conic Programming (MISOCP) methods in commercial solvers such as Knitro [49], Xpress [50], Gurobi [51], and CPLEX [52].

The discrete nature of the integer variables in mixed-integer programming problems has been exploited to derive efficient solution methods for these problems. In particular, deriving sets of extra inequalities, *cutting planes* or *cuts*, has allowed a considerable speedup in the solution of these problems, see [53]. One of the key disciplines for deriving such cut-

ting planes is Disjunctive Programming, which considers the optimization over disjunctive sets such as the one given by the domain of the discrete variables. In the convex nonlinear setting, the conic structure has been exploited to derive special cutting planes for MICP solution methods [54–56]. A source of these problems are those driven by *indicator variables*, that activate or deactivate sets of constraints [48], see a review by Bonami et al. [57].

Generalized Disjunctive Programming (GDP) was proposed by Grossmann and Lee [58] as an intuitive way of describing the logic behind applications. In this setting, sets of constraints are activated with logical variables linked to each other by logical constraints, including disjunctions. This mathematical description of the problem can be tackled directly by logic-based optimization methods [59], which generalize mixed-integer solution methods to the logical domain. Another way of solving these problems is through reformulations into mixed-integer programs, where the logical variables are mapped to binary or indicator variables. Depending on the linearity of the constraints within the GDP, the reformulations can yield an MILP or MINLP problem. The two most common reformulations are: the Big-M reformulations, where a large coefficient is added to make the constraints redundant in the case their associated indicator variable is inactive; and the Hull Reformulation (HR), where using Disjunctive Programming theory, a set of constraints in an extended space are derived such that their projection onto the space of the original variables is the convex hull of the disjunctive sets. These two reformulations yield different mixed-integer models, which can be characterized by size and tightness. The tightness of a mixed-integer model is measured through the difference of the optimal solution of the problem, ignoring the discrete constraints, known as the *continuous relaxation*, and the original problem optimal solution [60]. The Big-M and Hull reformulations offer a tradeoff between tightness and problem size. The HR is the tightest possible model, while the Big-M formulation does not require any additional continuous variables and constraints. Both the model size and

tightness are relevant to the efficiency of solution methods of mixed-integer programs [61].

For convex GDPs, the HR requires modeling the perspective function of the convex functions in the disjunctions, which can be complex for nonlinear functions given its non-differentiability at 0 [61, 62]. Perspective functions arise in formulations of convex MINLP since they are, in general, part of the reformulation of disjunctive programs. Moreover, the MINLP formulations involving the perspective function can be used either directly in tight formulations of convex disjunctive programs, either in the original variable space [57, 63] or in a higher dimensional space [58, 64], or indirectly through the generation of valid cutting-planes [65, 66]. A recent computational study shows the positive impact of perspective cuts in the MINLP framwork [46]. The importance of this perspective formulations and the challenges associated with their implementation have motivated its study, where customized versions have been derived for special cases [48, 63, 67] or the proposal of ε-approximations for general convex functions [62, 64].

This dissertation presents a series of algorithms to tackle optimization problems using the Turing model and the von Neumann computing architecture [68]. Compared to novel unconventional computing architectures, Turing models of computation get the name of *classical* or *conventional* computing. In contrast, *unconventional* computing methods [69] deviate from the usage of the von Neumann architecture for performing computations. These unconventional methods aim to tackle shortcomings of classical computing, such as its limited capability to solve problems requiring a high level of parallelism [70] and the objective of facing NP problems more efficiently, without claiming to overcome the barriers given by the complexity NP class. Among the unconventional computing approaches, we consider the use of *Quantum computing*. Quantum computing denotes the use of phenomena explained through quantum mechanics, such as entanglement and superposition, to perform computation, where the fundamental unit of computation is the quantum bit, or

qubit. These phenomena cannot be efficiently simulated with Turing machines, showing the potential of Quantum Computing to perform certain operations more efficiently [71]. Algorithms have been designed to work on quantum computers, *quantum algorithms*, with the goal of obtaining speedups compared to classical algorithms. Several examples of quantum algorithms with provable speedup with respect to the best-known classical algorithms have been found, *e.g.,* Shor's algorithm for integer factorizaton [72] and Grover's algorithm for unstructured search [73]. The implementation of these quantum algorithms is done through a series of quantum operators, or *quantum gates*, applied in sequence to a set of qubits. This sequence of gates followed by measurements of the qubits is denoted as a *circuit*. For an comprenhensive references in Quantum Computing, we refer the reader to the books by Rieffel and Polak [71] and Nielsen and Chuang [74].

Quantum algorithms have been developed to address convex optimization problems, such as linear programs (LP) and semidefinite programs (SDP). Convex optimization problems are a sub-class of continuous optimization where the decision variables are continuous, and the objective and feasible region described by the constraints are convex [75]. In general, solution algorithms for convex optimization problems require the iterative solution of systems of linear equations, akin to Newton methods. Interior-point methods are fundamental since these require a polynomial number of iterations with respect to the input size. Quantum algorithms for linear algebra, whose most prominent example is the HHL method from Harrow, Hassidim, and Lloyd [76], yield approximate solutions to linear systems of equations. Thus, quantum algorithms for convex problems are mainly based on exploiting these quantum methods for linear algebra as subroutines in interior-point methods [77, 78].

The class of discrete optimization problems has also been studied from a Quantum Computing perspective. The fundamental challenge in discrete optimization is the combinatorial growth in the solution space. For instance, consider a problem with n binary (*e.g.,* 0 or

1) decision variables. The solution is one of 2^n possible combinations, and so brute-force enumeration quickly becomes impractical as n increases. Meanwhile, the state of n qubits is described mathematically as a vector in a complex Hilbert space with dimension 2^n. This vast search space and the peculiar properties of Quantum Computing, including entanglement, superposition, and (destructive) interference, have inspired the development of algorithms for performing optimization on quantum computers. These algorithms are usually applied to Quadratic Unconstrained Binary Optimization (QUBO) problems, a particular discrete optimization problem with 0-1 binary variables, and a quadratic, *i.e.*, degree two polynomial, function of these variables as an objective. This class of problems has a well-known correspondence with the Ising model in physics, which describes ferromagnetism in statistical mechanics and phase transitions occurring in these systems. Many classic combinatorial optimization problems can be formulated as a QUBO or Ising model [79]. The energy function or Hamiltonian of an Ising model of n spins has a direct extension to a quantum mechanical Hamiltonian of n qubits. Consequently, a minimum energy state of n qubits can be used to give a solution to the original QUBO.

The Quantum Adiabatic Algorithm was one of the first quantum algorithms suggested for discrete optimization [80]. This algorithm is inspired by the quantum adiabatic theorem, which states conditions under which a system's energy ranking is preserved as it evolves according to the time-dependent Schrödinger equation. These conditions typically include the assumption that the evolution is "slow enough," depending on a gap condition on the spectrum of the time-dependent Hamiltonian[81]. In the Quantum Adiabatic Algorithm, an optimization problem is formulated as an Ising model. The Hamiltonian of this Ising model is called the final or driver Hamiltonian. Meanwhile, a quantum system described by a different Hamiltonian is prepared in its known lowest energy state; this Hamiltonian is called the initial or mixing Hamiltonian. By slowly interpolating between the initial and

final Hamiltonians, the quantum system evolves in a quasi-steady state fashion, always approximating the lowest energy state of the interpolated Hamiltonian. More precisely, the adiabatic theorem may be applied to assert that there is a high (arbitrarily close to one) probability that the quantum system is in the lowest energy state at each instant in time. In particular, the quantum system at the final time has a high probability of predicting the solution of the original optimization problem.

The ideas behind the Quantum Adiabatic Algorithm have since been generalized to a new model of Quantum Computing called Adiabatic Quantum Computing [81]. Adiabatic Quantum Computing is a computational model that relies on quantum mechanical processes happening under the assumption of adiabaticity. It serves as an idealized framework to perform algorithm analysis but cannot be implemented directly in practice. These theoretical advances, as well as the original hope that the Quantum Adiabatic Algorithm might yield speedups for hard combinatorial problems, has led to the analog implementation of the method in a non-ideal setting of finite temperature and an open environment, in an algorithm known as *quantum annealing*[82]. In terms of optimization algorithms, quantum annealing is a meta-heuristic for solving QUBO problems [83]. The specialized devices for running quantum annealing are called Quantum Annealers. These devices have been built using superconducting electronics to represent the qubits, while the quadratic and linear interactions included in the energy function are implemented using external magnetic fields. The best-known Quantum Annealers are produced by D-Wave systems, with up to 5000 qubits [84].

This general body of theory has also inspired algorithms intended to be implemented on gate-based quantum computers. In general, a quantum algorithm for solving discrete optimization problems on gate-based computers can be summarized in the following steps:

1. Map the optimization problem to a QUBO problem or an Ising problem

2. Assign the logical identity of each binary variable in the QUBO or spin variable in the Ising model to a qubit in a system
3. Apply a sequence of gates to set the system in a complete superposition state, equivalent to the lowest energy state of the Mixing Hamiltonian
4. Apply a sequence of gates, or circuit, such that the probability of the outcome measurement is the optimal solution of the problem is maximized
5. Measure the state by reading the qubits, which will output a set of values for each qubit optimistically being the optimal solution to the optimization problem
6. Repeat this procedure several times and return the best-found solution

The main challenge in the design of the algorithm is in the definition of the circuit that maximizes the probability of obtaining the optimal solution, given that these operators of gates are matrices of complex numbers of dimension $2^n \times 2^n$. Classically simulating these systems becomes increasingly expensive given the dimensionality explosion of the problem, while the quantum computer performs this calculation naturally. This procedure needs to be repeated several times given the probabilistic nature of the quantum systems, motivating its repeated measurements or *shots* to bound the found solutions.

The first among these is the Quantum Approximate Optimization Algorithm [85] (QAOA). Once again, this method applies to QUBO problems represented as Ising models. This method approximates the evolution of the system in the Quantum Adiabatic Algorithm with a discrete sequence of quantum gates; the specific approximation is a "Trotterization" of the evolution. The quantum gates are parameterized and must be tuned or optimized so that the algorithm produces a quantum state with a high probability of being in the lowest energy state and consequently to yield a solution to the optimization problem. This algorithm is also an *approximation algorithm*, in the sense that performance guarantees can be theoretically derived in the form of a guaranteed solution quality to be obtained when

applied to a well-defined family of problems [86]. In some cases, the optimal parameter settings of the quantum gates can be determined through careful analysis of the problem. In general, the approach is to embed the quantum algorithm in a classical optimization loop to search for the gate parameters.

This general structure characterizes the current wave of hybrid quantum-classical algorithms, the prototype of which is the variational quantum eigensolver [87] (VQE). The variational quantum eigensolver uses a parameterized quantum circuit to evaluate the system's energy when applied to an Ising model. The energy of the system is then minimized with respect to the circuit parameters. This optimization problem is continuous but typically stochastic due to the noise in evaluating the energy. While the form of the parameterized circuit might be inspired by the circuit used in the quantum approximate optimization algorithm, there is more freedom to develop heuristic circuit structures that might be more efficient in implementing specific hardware architectures.

Nonlinear discrete optimization is a challenging computational task requiring novel approaches from the solution algorithm and the problem formulation to be practically solved. These optimization problems arise in PSE and in other disciplines, motivating the efforts to solve them efficiently. In this dissertation, we aim to provide several different approaches to tackle these problems.

1.3 Dissertation Overview

The following subsections provide summaries of their respective chapters.

Chapter 2. A review and comparison of solvers for convex MINLP

Chapter 2 first reviews the deterministic software for solving convex MINLP problems as well as a comprehensive comparison of a large selection of commonly available solvers. Solvers are broadly classified into two large groups, those based on Branch & Bound and those based on MILP Decomposition Techniques. As a test set, we have used all MINLP instances classified as convex in the problem library MINLPLib [88], resulting in a solver benchmark against 335 instances. A summary of the most common methods for solving convex MINLP problems is given to highlight the differences between the solvers better. To show how the solvers perform on problems with different properties, we have divided the test set into subsets based on the continuous relaxation gap, the degree of nonlinearity, and the relative number of discrete variables. The results also provide guidelines on how well suited a specific solver or method is for particular types of MINLP problems.

Chapter 3. Feasibility Pump implementation in DICOPT

In Chapter 3, we discuss the implementation of the feasibility pump algorithm in the commercial solver DICOPT. The solver DICOPT is based on the Outer-approximation algorithm used for solving MINLP problems. This algorithm is very effective for solving some types of convex MINLPs. However, it has been observed that DICOPT has difficulties solving instances in which some of the nonlinear constraints are so restrictive that nonlinear subproblems generated by the algorithm are infeasible. This problem is addressed in this paper with a feasibility pump algorithm, which modifies the objective function to find feasible solutions efficiently. It has been implemented as a preprocessing algorithm, which is used to initialize both the incumbent and the mixed-integer linear relaxation of OA. Computational comparisons with previous versions of DICOPT on a set of convex MINLPs demonstrate the effectiveness of the proposed algorithm in terms of solution quality and

solution time.

Chapter 4. Center-cut Algorithm for Convex MINLP

Chapter 4 introduces the Center-cut algorithm for convex MINLP problems. This algorithm can either be used as a primal heuristic or as a deterministic solution technique. Like several other algorithms for convex MINLP, the Center-cut algorithm constructs a linear approximation of the original problem. The main idea of the algorithm is to use the linear approximation differently to find feasible solutions within a few iterations. The algorithm chooses trial solutions as the center of the current linear outer approximation of the nonlinear constraints, making the trial solutions more likely to satisfy the constraints. The ability to find feasible solutions using few iterations makes the algorithm well suited as a primal heuristic, and we prove that the algorithm finds the optimal solution within a finite number of iterations. Numerical results show that the algorithm obtains feasible solutions quickly and is able to obtain good solutions.

Chapter 5. Outer-approximation with Quadratic Cuts

Chapter 5 presents the use of scaled quadratic cuts based on scaling the second-order Taylor expansion terms for the Outer-approximation (OA) and Partial Surrogate Cuts (PSC) decomposition methods for solving convex MINLP problems. The scaled quadratic cut is proved to be a stricter and tighter underestimation for convex nonlinear functions compared to classical supporting hyperplanes, which results in the improvement of OA and PSC-based solution methods. We integrate scaled quadratic cuts strategy with multi-generation cuts for both OA and PSC and develop six types of MINLP solution methods. These cuts are incorporated in the master problem of the decomposition methods leading to an MIQCP problem. Numerical results of benchmark MINLP problems demonstrate the effectiveness

of the proposed solution methods with scaled quadratic cuts.

Chapter 6. Use of Regularization and Second-Order Information for Outer-approximation

Chapter 6 presents two new methods for solving convex MINLP problems based on the OA method. The first method is inspired by the level method and uses a regularization technique to reduce the step size when choosing new integer combinations. The second method combines ideas from both the level method and the sequential quadratic programming technique and uses a second-order approximation of the Lagrangean when choosing the new integer combinations. The main idea behind the methods is to choose the integer combination more carefully at each iteration to obtain the optimal solution in fewer iterations than the original Outer-approximation method. We prove rigorously that both methods will find and verify the optimal solution in a finite number of iterations. Furthermore, we present a numerical comparison of the methods based on 109 test problems to illustrate their advantages.

Chapter 7. Alternative Regularizations for Outer-approximation

In Chapter 7, we extend the regularization framework from Kronqvist, Bernal, and Grossmann [89] presented in Chapter 6 by incorporating several new regularization functions and develop a regularized single-tree search method for solving convex MINLP problems. We propose a set of regularization functions based on distance-metrics and Lagrangean approximations, used in the projection problem for finding new integer combinations to be used within the Outer-approximation (OA) method. The new approach, called Regularized Outer-approximation (ROA), has been implemented as part of the open-source **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy. We compare the OA

method with seven regularization function alternatives for ROA. Moreover, we extend the LP/NLP Branch & Bound method proposed by Quesada and Grossmann [90] to include regularization in an algorithm denoted RLP/NLP. We provide convergence guarantees for both ROA and RLP/NLP. Finally, we perform an extensive computational experiment by considering all convex MINLP problems in the benchmark library MINLPLib. The computational results show clear advantages of using regularization in combination with the OA method.

Chapter 8. Easily Solvable Convex MINLP Derived from Generalized Disjunctive Programming using Cones

Chapter 8 presents a different approach to solving convex MINLP problems more efficiently. Instead of tackling the solution algorithm, this chapter covers the formulations of the convex MINLP problems. Through Generalized Disjunctive Programming (GDP), we model different application problems involving the disjunction over convex sets. These GDP problems can be reformulated into MINLP through the Big-M and Hull Reformulation (HR). If the constraints in the disjunctions are nonlinear, the HR involves modeling the perspective function, a function that is non-differentiable at a solution with binary variables being equal to zero. By representing the convex inequalities as conic sets, we propose a conic GDP problem, which can be reformulated into a Mixed-Integer Conic Programming (MICP) problem. These problems can be directly tackled by solvers that take advantage of the conic structure of the problem, yielding more efficient and stable performance than the solvers based on gradients. We introduce the reformulation into MICP problems through the Big-M and HR reformulation. With the particularity that both these MICP problems are representable using the same cones as the original GDP and avoiding the numerical difficulties associated with HR's perspective function. We solve 425 GDP problems classified

into two groups: those with quadratic constraints and those with exponential constraints. The problems originated from PSE applications, Machine Learning (ML) problems, and randomly generated instances. Our results prove that the conic representation of convex GDP problems allows an exact reformulation through HR, which results in a more efficiently solvable problem given the potential usage of conic programming-based solvers.

Chapter 9. Characterization of QUBO Reformulations for the Maximum k-colorable Subgraph Problem

Chapter 9 considers the maximum k-colorable subgraph (MkCS) problem, a constrained combinatorial optimization (COPT) problem, and presents two formulations, a linear and a nonlinear formulation, for this problem. The nonlinear formulation is a better fit for addressing it through quantum annealing after a reformulation into Quadratic Unconstrained Binary Optimization (QUBO). Quantum devices can be used to solve COPT problems thanks to penalty methods to embed the COPT problem's constraints in its objective to obtain a QUBO reformulation of the COPT. However, the particular way this penalty is carried out affects the value of the penalty parameters and the number of additional binary variables that are needed to obtain the desired QUBO reformulation. In turn, these factors substantially affect the ability of quantum computers to efficiently solve these constrained COPT problems.

The MkCS problem arises in channel assignment in spectrum sharing networks, VLSI design, human genetic research, and cybersecurity. We derive two QUBO reformulations for the MkCS problem and fully characterize the range of the penalty parameters used in the QUBO reformulations. Furthermore, the nonlinear formulation needs not introduce additional binary variables when reformulating it into a QUBO. To illustrate the benefits of obtaining and characterizing these QUBO reformulations, we benchmark the different

QUBO reformulations of the MkCS problem by performing numerical tests on D-Wave's quantum annealing devices. The results presented in this chapter show the advantages of the nonlinear formulation when addressing the problem through Quantum Annealing.

Chapter 10. Formulating and Solving Routing Problems on Quantum Computers

The determination of vehicle routes fulfilling connectivity, time, and operational constraints is a well-studied combinatorial optimization problem. The NP-hard complexity of vehicle routing problems has promoted the adoption of tailored exact approaches, metaheuristics, and metaheuristics on classical computing devices. The ongoing evolution of Quantum Computing hardware and the recent advances of quantum algorithms, *i.e.,* VQE and QAOA, for mathematical programming, make decision-making for routing problems an avenue of research worthwhile to be explored on quantum devices. In Chapter 10, we propose several mathematical formulations for inventory routing cast as vehicle routing with time windows and comment on their strengths and weaknesses. The optimization models are compared from a Quantum Computing perspective, specifically with metrics to evaluate the difficulty in solving the underlying quadratic unconstrained binary optimization problems. Finally, the solutions obtained on simulated quantum devices demonstrate the relative benefits of different algorithms and their robustness when put into practice.

Chapter 11. Conclusions

Chapter 11 concludes this Thesis, providing a critical review, a listing of the most relevant contributions, achievements, and results, as well as a discussion of future research directions.

Appendix A. Integer Programming Techniques for Minor-Embedding in Quantum Annealers

A significant limitation of current generations of quantum annealers is the sparse connectivity of manufactured qubits in the hardware graph. This technological limitation has generated considerable interest, motivating efforts to design efficient and adroit minor-embedding procedures that bypass sparsity constraints. In Appendix A, starting from a previous equational formulation by Dridi, Alghassi, and Tayur [91], we propose integer programming (IP) techniques for solving the minor-embedding problem. The first approach involves a direct translation from the previous equational formulation to IP, while the second decomposes the problem into an assignment master problem and fiber condition checking subproblems. The proposed methods can detect instance infeasibility and provide bounds on solution quality, capabilities not offered by currently employed heuristic methods. We demonstrate the efficacy of our methods with an extensive computational assessment involving three families of random graphs of varying sizes and densities. The direct translation as a monolithic IP model can be solved with existing commercial solvers yielding valid minor-embeddings, but it is outperformed, overall, by the decomposition approach. Our results demonstrate the promise of our methods for the studied benchmarks, highlighting the advantages of using IP technology for minor-embedding problems.

1.4 Notation

During this Thesis, we use a similar notation to the one used by Ben-Tal and Nemirovski [31] and Alizadeh and Goldfarb [92]. We use lower case boldface letters, e.g., $\mathbf{x}, \mathbf{c}$, to denote column vector, and uppercase boldface letters, e.g., $\mathbf{A}, \mathbf{X}$, to denote matrices. Sets are denoted with uppercase calligraphic letters, e.g., $\mathcal{S}, \mathcal{K}$. Subscripted vectors denote $\mathbf{x_i}$ denote

the i^{th} block of $\mathbf{x}$. The j^{th} component of the vectors $\mathbf{x}$ and $\mathbf{x_i}$ are indicated as x_j and x_{ij}. The set $\{1,\dots,J\}$ is represented by the symbol $[\![J]\!]$. Moreover, the subscript $[\![J]\!]$ of a vector $\mathbf{x}$ is used to define the set $\mathbf{x}_{[\![J]\!]} := \{\mathbf{x}_1,\dots,\mathbf{x}_J\}$. We use $\mathbf{0}$ and $\mathbf{1}$ for the all zeros and all ones vector, respectively, and 0 and I for the zero and identity matrices, respectively. The vector e_j will be the vector with a single 1 in position j, and its remaining elements being 0. The dimensions of the matrices and vectors will be clear from the context. We use $\mathbb{R}^k$ to denote the set of real numbers of dimension k, and for set $\mathcal{S} \subseteq \mathbb{R}^k$, we use $\text{cl}(\mathcal{S})$ and $\text{conv}(\mathcal{S})$ to denote the closure and convex hull of $\mathcal{S}$, respectively.

For concatenated vectors, we use the notation that "," is row concatenation of vectors and matrices, and ";" is column concatenation. For vectors, $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$, the following are equivalent.

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{pmatrix} = (\mathbf{x}^\top, \mathbf{y}^\top, \mathbf{z}^\top)^\top = (\mathbf{x}; \mathbf{y}; \mathbf{z}). \tag{1.1}$$

The projection of a set $\mathcal{S} \subseteq \mathbb{R}^k$ onto the vector $\mathbf{x} \in X \subseteq \mathbb{R}^n$, with $n \leq k$ is denoted as $\text{proj}_{\mathbf{x}}(\mathcal{S}) := \{\mathbf{x} \in X : \exists \mathbf{y} : (\mathbf{x}; \mathbf{y}) \in \mathcal{S}\}$.

If $\mathcal{A} \subseteq \mathbb{R}^k$ and $\mathcal{B} \subseteq \mathbb{R}^l$ we denote their Cartesian product as $\mathcal{A} \times \mathcal{B} := \{(\mathbf{x}; \mathbf{y}) : \mathbf{x} \in \mathcal{A}, \mathbf{y} \in \mathcal{B}\}$.

For $\mathcal{A}_1, \mathcal{A}_2 \subseteq \mathbb{R}^k$ we define the Minkowski sum of the two sets as $\mathcal{A}_1, \mathcal{A}_2 = \{\mathbf{u} + \mathbf{v} : \mathbf{u} \in \mathcal{A}_1, \mathbf{v} \in \mathcal{A}_2\}$.

Chapter 2

A review and comparison of solvers for convex MINLP*

2.1 Introduction

In this chapter, we present a review of deterministic software for solving convex MINLP problems as well as a comprehensive comparison of a large selection of commonly available solvers. As a test set, we have used all MINLP instances classified as convex in the problem library MINLPLib, resulting in a test set of 335 convex MINLP instances. A summary of the most common methods for solving convex MINLP problems is given to better highlight the differences between the solvers. To show how the solvers perform on problems with different properties, we have divided the test set into subsets based on the continuous relaxation gap, the degree of nonlinearity, and the relative number of discrete variables. The results also provide guidelines on how well suited a specific solver or method is for particular types of MINLP problems.

This chapter intends to give an overview of commonly available deterministic solvers for convex MINLP problems and present a thorough numerical comparison of the most common solvers. Most optimization solvers are connected to one or more of the well-established modeling environments for MINLP optimization, such as, AIMMS [93], AMPL

*Published as: Jan Kronqvist, David E Bernal, Andreas Lundell, and Ignacio E Grossmann. "A review and comparison of solvers for convex MINLP". *Optimization and Engineering* 20.2 (2019), pp. 397–455.

[94], and GAMS [95]. In recent years, there has also been a growing interest in optimization modeling in Python and Julia [96]; JuMP is a modeling environment for optimization embedded in Julia [97], and Pyomo is a similar environment in Python [98]. Several MINLP solvers also offer interfaces to MATLAB, and through OPTI Toolbox, it is also possible to access several solvers in MATLAB [99].

The solvers considered in the numerical comparison are AlphaECP, Antigone, AOA, BARON, BONMIN, Couenne, DICOPT, Juniper, KNITRO, LINDO, Minotaur, Muriqui, Pavito, SBB, SCIP, and SHOT. These were chosen based on criteria like availability, active development, and support for a file format available in MINLPLib [100]. Some of these are global solvers and therefore not limited to convex problems. However, most global solvers have convexity identification techniques or manual strategies that the user can set to deal with convex problems more efficiently. The convex solvers can also often be used as heuristic methods without guaranteeing to find the optimal solution for nonconvex MINLP problems.

In Section 2.2, the convex MINLP problem is defined, and a general overview of the most common algorithms for such problems is given in Section 2.3. Most solvers in the comparison utilize one or more of these solution methods, as described in Section 2.4, which contains a summary of the solvers considered. Section 2.5 describes the benchmark in detail, and the numerical results are, finally, presented in Section 2.6.

2.2 Convex MINLP problem formulation

A convex MINLP problem can, without loss of generality, be written as

$$\min_{\mathbf{x},\mathbf{y}\in\mathcal{N}\cap\mathcal{L}\cap\mathcal{Y}} \mathbf{c}_1^\top \mathbf{x} + \mathbf{c}_2^\top \mathbf{y}, \tag{P-MINLP}$$

where the sets $\mathcal{N}, \mathcal{L}$ and $\mathcal{Y}$ are given by

$$\begin{aligned} \mathcal{N} &= \{\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m \mid g_j(\mathbf{x}, \mathbf{y}) \leq 0 \quad \forall j = 1, \dots l\}, \\ \mathcal{L} &= \{\mathbf{x} \in \mathbb{R}^n, \mathbf{y} \in \mathbb{R}^m \mid \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}\}, \\ \mathcal{Y} &= \{\mathbf{y} \in \mathbb{Z}^m \mid \underline{y}_i \leq y_i \leq \overline{y}_i \quad \forall\, i = 1, 2, \dots, m\}. \end{aligned} \tag{2.1}$$

and $\mathcal{L} \cap \mathcal{Y}$ is assumed to be a compact set. The upper and lower bounds on the integer variables y_i are denoted as $\overline{y}_i$ and $\underline{y}_i$. To ensure convergence of methods such as Outer-approximation, it is assumed that all the integer variables are bounded either by the variables bounds or by the linear constraints, since unbounded variables can, *e.g.*, cause some of the subproblems to be unbounded. Most solvers do not require the variables to be bounded, however, to avoid such issues some solvers automatically assigns large bounds to unbounded variables. Generally, problem P-MINLP is considered as convex if all the nonlinear functions g_j are convex in the variables $\mathbf{x}$ and the relaxed integer variables $\mathbf{y}$. There has recently been an interest in nonsmooth convex MINLP, and some solution techniques have been presented see *e.g.*, [101] and [102]. However, most of the commonly available solvers only have guaranteed convergence for smooth problems and therefore we limit this study to problems where the nonlinear functions g_j are continuously differentiable.

2.3 Methods

This section describes the most commonly used algorithms for convex MINLP. The methods described are branch and bound, extended cutting plane, extended supporting hyperplane, Outer-approximation, generalized Benders decomposition, and LP/NLP-based branch and bound. This summary is not intended to give an in-depth analysis of the algorithms but to better exemplify the differences between the solvers. For a more detailed discussion about the algorithms see, *e.g.*, [103–106].

2.3.1 Branch and bound

Branch and bound (BB) was first presented as a technique for solving MILP problems by [107]. A few years later, it was noted by [27] that MINLP problems could be solved with a similar branch and bound approach, although the paper focused on linear problems. Solving convex MINLP problems with a BB approach was also studied by [108].

In the basic form, BB solves the MINLP problem by relaxing the integer restrictions of the original problem and solving continuous (convex) NLP relaxations. Solving a continuous relaxation of problem P-MINLP results in a solution $(\mathbf{x}^k, \mathbf{y}^k)$, which provides a valid lower bound. Suppose all components of $\mathbf{y}^k$ take on integer values. In that case, it is also an optimal solution to the MINLP problem. Otherwise, the continuous relaxation is divided (branched) into two new NLP subproblems by adding the constraints $y_i \leq \lfloor y_i^k \rfloor$ and $y_i \geq \lceil y_i^k \rceil$ to the relaxed problem. The branching variable y_i is a variable that takes on a fractional value and usually chosen based on some criteria, *e.g.*, the variable furthest away from an integer value. A new lower bound can be obtained by solving the new subproblems (child nodes). If one of the subproblems returns an integer solution, it also provides a valid upper bound. The search procedure is often represented by a tree, where the nodes are connected to their parent node and represent the subproblems. If one of the nodes does not provide an integer solution, it is branched into two new nodes creating two new subproblems. In case one of the nodes obtains an optimum worse than the upper bound, or if the subproblem is infeasible, then the node can be pruned since an optimal solution cannot exist in that part of the search space. This approach of solving convex NLP problems in each node is often referred to as NLP-based branch and bound (NLP-BB).

Obtaining a tight continuous relaxation is of great importance within BB to avoid large search trees. [65] presented a branch and cut method for convex MINLP problems that uses cutting planes to strengthen the continuous relaxation. Several techniques have been

proposed for obtaining cuts to strengthen the continuous relaxation for MINLP problems, *e.g.,* lift-and-project cuts [109–111], Gomory cuts [54, 112], and perspective cuts [66].

Compared to BB techniques for MILP problems, NLP-BB involves computationally more demanding subproblems; it is often not unusual to explore more than 100,000 nodes for a modest-sized problem! Techniques to efficiently integrate the NLP solver and not solving all subproblems to optimality have also been proposed by [113], and [114]. Another BB approach is to solve LP relaxations in the nodes and construct a polyhedral approximation of the nonlinear constraints. A polyhedral branch and cut technique, solving LP relaxations in the nodes, was presented by [115].

Many important details on BB, such as branching strategies, have been left out for brevity. For more details on BB see, *e.g.,* [116] and [117].

2.3.2 Extended cutting plane

The extended cutting plane (ECP) algorithm was first presented by [118] and can be seen as an extension of Kelley's cutting plane method for convex NLP problems presented by [119]. In its original form, the ECP method is intended for convex MINLP problems. By some modifications, given the name generalized alpha ECP (GAECP), it can be applied to pseudoconvex problems as shown by [120].

The ECP algorithm uses linearization of the nonlinear constraints to construct an iteratively improving polyhedral outer approximation of the set $\mathcal{N}$. The trial solutions are obtained by solving the following MILP subproblems,

$$(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}) \in \underset{\mathbf{x},\mathbf{y} \in \mathcal{N}_k \cap \mathcal{L} \cap \mathcal{Y}}{\arg\min} \; \mathbf{c}_1^\top \mathbf{x} + \mathbf{c}_2^\top \mathbf{y}, \tag{MILP-k}$$

where the set $\mathcal{N}_k$ is given by

$$\mathcal{N}_k = \left\{ g_j(\mathbf{x}^i, \mathbf{y}^i) + \nabla g_j(\mathbf{x}^i, \mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x} - \mathbf{x}^i \\ \mathbf{y} - \mathbf{y}^i \end{bmatrix} \leq 0 \quad \forall i = 1, 2 \ldots k, \; j \in J_i \right\}. \tag{2.2}$$

Here J_i is an index set containing the indices of either the most violated or all violated constraints in iteration i. Set $\mathcal{N}_k$ is, thus, a polyhedral approximation of set $\mathcal{N}$, constructed by first-order Taylor series expansions of the nonlinear constraints generated at the trial solutions $(\mathbf{x}^k, \mathbf{y}^k)$. The linearizations defining $\mathcal{N}_k$ are usually referred to as cutting planes since they cut off parts of the search space that cannot contain the optimal solution. Due to convexity, $\mathcal{N} \subseteq \mathcal{N}_k$ and therefore, the solution of problem MILP-k provides a valid lower bound of problem P-MINLP.

In the first iteration, the set $\widehat{\mathcal{N}}_0$ can simply be defined as $\mathbb{R}^{n+m}$. New trial solutions are then obtained by solving subproblem MILP-k, and the procedure is repeated until a trial solution satisfies all the constraints within a given tolerance. Once a trial solution satisfies all nonlinear constraints, it is also the optimal solution since the solution was obtained by minimizing the objective within a set containing the entire feasible region. For more details on the ECP algorithm see, *e.g.*, [118] or [120].

2.3.3 Extended supporting hyperplane

The extended supporting hyperplane (ESH) algorithm was presented by [121] as an algorithm for solving convex MINLP problems. The ESH algorithm uses the same technique as the ECP algorithm for obtaining trial solutions. However, it uses a different technique for generating the polyhedral outer approximation $\mathcal{N}_k$. It has been observed that the cutting planes used to construct the polyhedral outer approximation in the ECP algorithm are, in general, not as tight as possible, see [121]. Using a one-dimensional root search, the ESH algorithm can obtain supporting hyperplanes to the set $\mathcal{N}$ at each iteration and use these to

construct a polyhedral outer approximation $\mathcal{N}_k$.

First, a strict interior point $(\mathbf{x}_{\text{int}}, \mathbf{y}_{\text{int}})$ is obtained by solving the following convex NLP problem

$$\begin{aligned} \min_{(\mathbf{x},\mathbf{y})\in\mathcal{L},\mu\in\mathbb{R}} \quad & \mu \\ \text{s.t.} \quad & g_j(\mathbf{x},\mathbf{y}) \leq \mu \quad \forall j = 1,2,\ldots,l. \end{aligned} \tag{2.3}$$

The interior point should preferably be as deep as possible within the interior of $\mathcal{N}$, which is approximated by minimizing the l_∞-norm of the nonlinear constraints.

Similar to the ECP algorithm, the trial solutions $\left(\mathbf{x}^k_{\text{MILP}}, \mathbf{y}^k_{\text{MILP}}\right)$ are obtained by solving problem MILP-k. These solutions provide a valid lower bound on the optimal solution of problem P-MINLP. However, they will not be directly used to construct the set $\mathcal{N}_k$ as in the ECP method.

To construct the polyhedral outer approximation, we define a new function F as the point-wise maximum of the nonlinear constraints, according to

$$F(\mathbf{x},\mathbf{y}) = \max_j \left\{ g_j(\mathbf{x},\mathbf{y}) \right\}. \tag{2.4}$$

A new sequence of points $\left(\mathbf{x}^k, \mathbf{y}^k\right)$ is now defined as

$$\begin{aligned} \mathbf{x}^k &= \lambda^k \mathbf{x}_{\text{int}} + (1-\lambda^k)\mathbf{x}^k_{\text{MILP}}, \\ \mathbf{y}^k &= \lambda^k \mathbf{y}_{\text{int}} + (1-\lambda^k)\mathbf{y}^k_{\text{MILP}}, \end{aligned} \tag{2.5}$$

where the interpolation parameters λ^k are chosen such that $F(\mathbf{x}^k, \mathbf{y}^k) = 0$. The interpolation parameters λ^k can be obtained by a simple one-dimensional root search. The points $\left(\mathbf{x}^k, \mathbf{y}^k\right)$ are now located on the boundary of the feasible region, and linearizing the active nonlinear constraints at this point results in supporting hyperplanes to the set $\mathcal{N}$. The set $\mathcal{N}_k$ is, thus, constructed according to Eq.(2.2) using the points $\left(\mathbf{x}^k, \mathbf{y}^k\right)$.

The ESH algorithm also uses a preprocessing step to obtain supporting hyperplanes of the set $\mathcal{N}$ by solving linear programming (LP) relaxations. Solving MILP subproblems and

generating supporting hyperplanes is repeated until a trial solution satisfies all nonlinear constraints. The tighter polyhedral outer approximation usually gives the ESH algorithm an advantage over the ECP algorithm. It has been shown in [101] that the ESH algorithm can also be successfully applied to nonsmooth MINLP problems with pseudoconvex constraint functions.

2.3.4 Outer-approximation

The Outer-approximation (OA) method was first presented by [122], with additional properties for convex MINLP problems described in [25]. Some modifications of the OA method have been presented to handle nonconvex problems more efficiently, see, *e.g.*, [123], and [124]. For more details on the basic convex approach discussed in this paper, see, *e.g.*, [**grossmann2002review**].

OA is a decomposition technique that obtains the original problem's optimal solution by solving a sequence of MILP and NLP subproblems. Like ECP and ESH, OA also constructs an iteratively improving polyhedral outer approximation $\mathcal{N}_k$ of the nonlinear feasible region defined by the set $\mathcal{N}$. However, OA only uses the polyhedral approximation for choosing the integer combination $\mathbf{y}^k$. In contrast, the corresponding continuous variables $\mathbf{x}^k$ are chosen by solving a convex NLP subproblem.

In each iteration, the polyhedral outer approximation is used to construct problem MILP-k, referred to as the MILP master problem. A new integer combination $\mathbf{y}^k$ is then obtained by solving problem MILP-k. Once the integer combination $\mathbf{y}^k$ is obtained, the following NLP subproblem is formed

$$\begin{aligned} (\mathbf{x}^k, \mathbf{y}^k) \in \underset{(\mathbf{x},\mathbf{y})\in\mathcal{N}\cap\mathcal{L}}{\arg\min}\ & \mathbf{c}_1^\top \mathbf{x} + \mathbf{c}_2^\top \mathbf{y} \\ \text{s.t.}\quad & \mathbf{y} = \mathbf{y}^k. \end{aligned} \tag{NLP-fixed}$$

If problem NLP-fixed is feasible, a valid upper bound can be obtained from the solution

$(\mathbf{x}^k, \mathbf{y}^k)$, and the solution is used to improve the polyhedral approximation according to Eq. (2.2). The polyhedral outer approximation is updated by either linearizing all constraints or only the active constraints.

problem NLP-fixed may also be infeasible in some iteration. If $\mathbf{y}^k$ is an infeasible integer combination, the corresponding continuous variables can be obtained by solving the following convex subproblem

$$\begin{aligned}(\mathbf{x}^k, \mathbf{y}^k, r^k) \in \underset{(\mathbf{x},\mathbf{y})\in\mathcal{L}, r\in\mathbb{R}}{\arg\min}\ & r \\ \text{s.t.}\quad & \mathbf{y} = \mathbf{y}^k, \\ & g_j(\mathbf{x}, \mathbf{y}) \leq r \quad \forall j = 1, 2, \ldots, l,\end{aligned} \tag{NLP-feasibility}$$

which minimizes the constraint violation with respect to the l_∞-norm. The solution to problem NLP-feasibility does not provide a lower bound. However, using the infeasible solution $(\mathbf{x}^k, \mathbf{y}^k)$ to update the polyhedral outer approximation according to Eq. (2.2), ensures that the infeasible integer combination $\mathbf{y}^k$ cannot be obtained again by the MILP master problem, *cf.* [25].

The OA algorithm is usually initiated by solving a continuous relaxation of the MINLP problem, giving an initial lower bound and a solution that can be used to construct the polyhedral approximation $\widehat{\mathcal{N}}_0$ [124]. It is also possible to use integer cuts to exclude specific integer combinations, as suggested by [122]. Solving the MILP master problems MILP-k provides a lower bound on the optimum. The procedure is repeated until the upper and lower bound are within a given tolerance.

In general, OA results in tighter polyhedral outer approximations than the ECP algorithm and may, therefore, require fewer iterations. For a feasible integer combination, OA will, in general, result in a tighter polyhedral outer approximation than ESH. However, for an infeasible integer combination, ESH can give a tighter approximation. OA may thus

require fewer iterations than both ESH and ECP to solve specific problems. However, since each iteration is somewhat more computationally demanding, the methods are difficult to compare directly.

2.3.5 Generalized Benders decomposition

Generalized Benders decomposition (GBD) was first presented by [28] and is a generalization of Benders decomposition, a partitioning procedure for solving MILP problems [125]. As noted by [90], GBD is closely related to OA, and the main difference is the derivation of the master problem. In GBD, the master problem is projected onto the space defined by the integer variables, and the master problem is, thus, only expressed in the integer variables. Here we will not present the full derivation of GBD but use the same approach as [**grossmann2002review**] to derive the master problem. For more details on GBD see, *e.g.*, [126] or [106].

Given an integer combination $\mathbf{y}^k$, the corresponding continuous variables can be obtained by solving either one of the problems NLP-fixed or NLP-feasibility. If problem NLP-fixed is feasible, it provides a valid upper bound, as well as values for the continuous variables $\mathbf{x}^k$ and the optimal Lagrangean multipliers λ^k and μ^k. A valid cut is then given by

$$\mathbf{c}_1^\top \mathbf{x}^k + \mathbf{c}_2^\top \mathbf{y} + \sum_{j=1}^{l} \lambda_j^k \nabla_{\mathbf{y}} g_j(\mathbf{x}^k, \mathbf{y}^k)^\top (\mathbf{y} - \mathbf{y}^k) + (\mu^k)^\top \mathbf{B}\mathbf{y} \leq \alpha, \tag{2.6}$$

where $\nabla_{\mathbf{y}}$ denotes the gradient with respect to the integer variables. Here, α is a new auxiliary variable used for describing the objective function of the MILP subproblems. Note that the left-hand side of Eq. (2.6) is a first order Taylor series expansion of the Lagrangean function of problem NLP-fixed at the point $(\mathbf{x}^k, \mathbf{y}^k, \lambda^k, \mu^k)$ with respect to the $\mathbf{x}$ and $\mathbf{y}$ variables, and that the gradient with respect to the $\mathbf{x}$ variables will be zero. The cut in Eq. (2.6) can be shown to be a surrogate constraint of the linearization in Eq. (2.2) in which the continuous

variables $\mathbf{x}$ are projected out, *cf.* [90] or [**grossmann2002review**].

If problem NLP-fixed is infeasible with the integer combination $\mathbf{y}^k$, problem NLP-feasibility is solved to obtain the continuous variables $\mathbf{x}^k$ as well as the optimal multipliers λ^k and μ^k. A valid cut in the integer space is then given by,

$$\sum_{j=1}^{l} \lambda_j^k \Big(g_j(\mathbf{x}^k, \mathbf{y}^k) + \nabla_{\mathbf{y}} g_j(\mathbf{x}^k, \mathbf{y}^k)^\top (\mathbf{y} - \mathbf{y}^k) \Big) + (\mu^k)^\top \mathbf{B}\mathbf{y} \leq 0. \tag{2.7}$$

For more details on the cuts see, *e.g.,* [90]. The master problem for obtaining new integer combinations, is then given by,

$$\begin{aligned}
\min_{\mathbf{y}\in\mathcal{Y}, \alpha\in\mathbb{R}} \quad & \alpha \\
\text{s.t.} \quad & \mathbf{c}_1^\top \mathbf{x}^k + \mathbf{c}_2^\top \mathbf{y} + \sum_{j=1}^{l} \lambda_j^k \nabla_{\mathbf{y}} g_j(\mathbf{x}^k, \mathbf{y}^k)^\top (\mathbf{y} - \mathbf{y}^k) + (\mu^k)^\top \mathbf{B}\mathbf{y} \leq \alpha \quad \forall k \in K_f, \\
& \sum_{j=1}^{l} \lambda_j^k \Big(g_j(\mathbf{x}^k, \mathbf{y}^k) + \nabla_{\mathbf{y}} g_j(\mathbf{x}^k, \mathbf{y}^k)^\top (\mathbf{y} - \mathbf{y}^k) \Big), \\
& \qquad + (\mu^k)^\top \mathbf{B}\mathbf{y} \leq 0 \quad \forall k \in K \backslash K_f,
\end{aligned}$$

where K_f contains the indices of all iterations where problem NLP-fixed is feasible and the index set K contains all iterations. Solving the master problem provides a lower bound on the optimal solution and gives a new integer combination $\mathbf{y}^{k+1}$. The procedure is repeated until the upper and lower bounds are within the desired tolerance.

Since the cuts obtained by equations (2.6) and (2.7) can be viewed as surrogate cuts of the linear constraints included in the OA master problem; GBD generates weaker cuts than OA at each iteration and usually requires more iterations to solve a given problem. However, the master problems in GBD may be easier to solve since they contain fewer variables than OA, and only one cut is added in each iteration.

A compromise between OA and GBD has been proposed by [90], where the continuous variables are classified into linear or nonlinear based on how they are involved in the

original MINLP problem. By projecting out the nonlinear continuous variables, one can derive a Lagrangean cut similar to GBD while still retaining the linear constraints involving continuous variables in the master problem. The given method has been coined as partial surrogate cuts (PSC). As proved in [90], it results in a tighter linear relaxation compared to GBD while still only adding one cut per iteration.

2.3.6 LP/NLP-based branch and bound

When solving a convex MINLP problem with either ECP, ESH, GBD, or OA, most of the total solution time is usually spent on solving the MILP subproblems. The MILP problems are also quite similar in consecutive iterations since they only differ by a few linear constraints. To avoid constructing many similar MILP branch and bound trees, [90] presented a method that integrates OA within BB, called LP/NLP-based branch and bound (LP/NLP-BB). The main idea is to construct a single branch and bound tree, where the MILP master problem is dynamically updated.

An initial polyhedral outer approximation is constructed by solving a continuous relaxation and linearizing the constraints at the relaxed solution, as in OA. The polyhedral outer approximation is used to construct the first MILP master problem. The branch and bound procedure, where an LP relaxation is solved in each node, is initiated. Once an integer solution is obtained at a given node, the integer combination is used as in OA. If the NLP problem NLP-fixed with the given integer combination is feasible; it provides an upper bound, and new linearizations are generated. If it is infeasible, new linearizations can be obtained by solving the feasibility problem NLP-feasibility.

The new linearizations are then added to all open nodes. The LP relaxation is resolved for the node that returned the integer combination. The branch and bound procedure continues normally by solving LP relaxations, giving a more accurate approximation of

the nonlinear constraints. Here, the search must continue down each node until either the LP relaxation returns an integer solution that satisfies all nonlinear constraints, the LP relaxation obtains an objective value worse than the upper bound, or until the LP relaxation becomes infeasible. As in regular BB, a lower bound is provided by the lowest optimal solution of the LP relaxations in all open nodes, and the search continues until the upper and lower bounds are within a given tolerance. The LP/NLP-BB procedure, thus, only generates a single branch and bound tree and is sometimes referred to as a single-tree OA.

Numerical results have shown that LP/NLP-BB technique can result in significantly fewer nodes than the total number of nodes explored in the multiple MILP master problems in OA [122, 127]. Implementations of the LP/NLP-BB algorithm have shown promising results, *cf.* [128], [129] or [130].

2.3.7 Solver enhancement techniques

Most solvers are not based on a single algorithm but combine several techniques to improve their performance. In this section, we briefly describe some preprocessing techniques and primal heuristics that can be integrated with all the previously mentioned methods to improve the practical performance. Other important techniques to improve the performance include various cutting planes as well as different branching rules. For more details on cutting planes for convex MINLP see, *e.g.,* [104], [131] and [109]. Overviews of branching rules are given by [132] and [133].

2.3.7.1 Preprocessing

Preprocessing includes various techniques for modifying the problem into a form more favorable for the actual solver. The preprocessing procedures can result in tighter relaxations or reduce the problem size. [104] classified MINLP presolving techniques into two

significant categories: housekeeping and reformulations. Housekeeping includes bound tightening and removal of redundant constraints. In contrast, reformulations can include improving coefficients in the constraints and disaggregation of constraints.

There are two main approaches for tightening the variable bounds, feasibility-based bound tightening [134, 135], and optimization-based bound tightening [136]. Feasibility-based bound tightening analyzes the constraints sequentially to improve the variable bounds. In contrast, optimization-based bound tightening solves a sequence of relaxed problems. Each variable is maximized and minimized to obtain optimal bounds.

By reformulating the original problem, it is in some cases possible to obtain significantly tighter relaxations. Within MILP, it is well known that different problem formulations can result in tighter or weaker continuous relaxations; the uncapacitated facility location problem is a good example of when disaggregation of some constraints leads to a tighter continuous relaxation [137]. Similar techniques can also be used to obtain tighter relaxations for MINLP problems. Some nonlinear constraints can also be disaggregated to obtain a lifted reformulation of the problem, where the nonlinear constraint is split into several constraints by introducing new variables. Such lifted reformulations were proposed by [115], where it was shown that a lifted reformulation results in tighter polyhedral outer approximations. In a recent paper by [138], it was shown that several MINLP solvers, based on ECP, ESH, and OA, could be drastically improved by utilizing a reformulation technique based on lifting. Lifted reformulations of MINLP problems have also been studied by [139], and [42]. Some further reformulation techniques for MINLP problems are also presented in [140].

2.3.7.2 Primal heuristics

Primal heuristics is a common term for algorithms and techniques intended to obtain good feasible solutions with relatively little computational effort compared to solving the original problem. The use of primal heuristics began in the field of MILP, and for instance, [141] claimed that primal heuristics were one of the most important improvements in MILP solvers within the last decade. In recent years, there has also been an interest in primal heuristics for MINLP problems. Several algorithms have been proposed for this task. Such algorithms are, *e.g.,* undercover [142], feasibility pumps [143], rounding heuristics [144], and the Center-cut algorithm [145]. Another technique for obtaining feasible solutions in solvers based on ECP, ESH or OA, is to check the alternative solutions in the solution pool provided by the MILP solver [121]. A detailed summary of several primal heuristics for MINLP problems is given by [146], and [147].

Finding a good feasible solution to an MINLP problem can improve the performance of MINLP solvers, as shown by the numerical results in [146] and [148]. Having a good feasible solution can, *e.g.,* reduce the size of the search tree in BB-based solvers and provide a tight upper bound. Obtaining a tight upper bound is especially important in solvers based on the ECP or ESH algorithm because neither of the algorithms will, in their basic form, obtain a feasible solution before the very last iteration.

2.4 Solvers

This section is intended as an introduction to commonly available MINLP solvers and to describe their main properties. Most of the solvers are not based on a single "pure" algorithm, but they combine several techniques and ideas to improve their performance. On top of this, MINLP solver technology has evolved from the more mature NLP and

MILP fields, and most MINLP solvers rely heavily on such subsolvers. Among the MILP solvers, the most recognized commercial solvers are CPLEX [52], Gurobi [149], and XPRESS [150]. The solvers GLPK [151] and CBC [152], the latter is a part of the COIN-OR initiative [153], and are among the most recognized open-source solvers for MILP. All of these solvers implement an arsenal of methods within a branch and cut framework. In the NLP case, solvers like CONOPT [154], Knitro [155], Mosek [156], and SNOPT [157] are well-known commercial options, and IPOPT [158] is a well-known open-source solver (also part of the COIN-OR initiative). There exists more variability in the algorithms behind NLP solvers, *e.g.*, CONOPT implements a Generalized Reduced Gradient (GRG) method, while IPOPT, Knitro, and Mosek use an interior-point method, and SNOPT uses a sequential quadratic programming (SQP) approach; see [159] for a review in NLP.

Besides convexity, some of the solvers mentioned here also require an algebraic formulation of the problem. By analyzing the problem structure and applying different reformulations, it is possible to obtain tighter relaxations. Furthermore, some of the NLP solvers also require the nonlinear functions to be twice continuously differentiable to guarantee convergence, which in turn imposes additional restrictions on some of the MINLP solvers.

In this section, we only mention the main features of the solvers, and for more details, see the references given in the solver sections. A summary of solvers and software for MINLP problems was previously also given by [88]. The solvers are implemented in various programming languages, either available as standalone executables or libraries accessible from algebraic modeling software like GAMS, AMPL, and AIMMS. Other solvers have been implemented directly in the same programming languages as their modeling systems, *e.g.*, MATLAB, Python-Pyomo, Julia-JuMP. The solvers used in the numerical comparison are listed in alphabetical order below.

2.4.1 AlphaECP

License type: Commercial

Interfaces: GAMS, NEOS

URL: www.gams.com/latest/docs/S_ALPHAECP.html

AlphaECP (**A**lpha **E**xtended **C**utting **P**lane) is a solver based on the αECP algorithm developed by T. Westerlund's research group at bo Akademi University and implemented in GAMS by T. Lastusilta. Using the GAECP algorithm [120] the solver also has guaranteed convergence for pseudoconvex MINLP problems. AlphaECP mainly solves a sequence of MILP subproblems to generate a polyhedral outer approximation through cutting planes. However, to speed-up convergence, it occasionally solves NLP subproblems with fixed integer variables as well. To improve the capabilities of handling nonconvex problems, the algorithm also employs some heuristic techniques described in [160]. An essential feature of AlphaECP is the technique to initially only solve MILP problems to feasibility [120]. This often results in a significant reduction in total solution time since fewer MILP subproblems are solved to optimality. AlphaECP can use all the NLP and MILP subsolvers available in GAMS.

2.4.2 ANTIGONE

License type: Commercial

Interfaces: GAMS, NEOS

URL: www.gams.com/latest/docs/S_ANTIGONE.html

ANTIGONE (**A**lgorithms for co**NT**inuous / **I**nteger **G**lobal **O**ptimization of **N**onlinear **E**quations) is a global optimization solver developed by R. Misener and C.A. Floudas at Princeton University. As a global solver, ANTIGONE is not limited to only convex problems but can also solve a variety of nonconvex problems. It uses reformulations and

decomposes nonlinear functions into constant, linear, quadratic, signomial, linear fractional, exponential, and other general nonconvex terms. Convex relaxations are then generated for the decomposed nonconvex terms, and the relaxations are solved in a branch and cut framework [161, 162]. ANTIGONE uses the local solvers CONOPT or SNOPT for finding feasible solutions and CPLEX for lower bounding MILP relaxations. The solver also uses both feasibility- and optimality-based bound tightening to reduce the search space and obtain tighter relaxations.

2.4.3 AOA

License type: Commercial (source code available)

Interfaces: AIMMS

URL: www.aimms.com/english/developers/resources/solvers/aoa

AOA (**A**IMMS **O**uter **A**pproximation) is a module implemented in the AIMMS language [163]. As the name suggests, the solver is based on OA and implements both normal OA and the LP/NLP-BB methods. The latter is the recommended one for convex problems and generates linearizations as lazy constraints utilizing MILP solver callbacks. AOA may use nonlinear preprocessing and a multi-start technique to improve the performance and its capabilities for solving nonconvex problems. The source code of AOA is included in AIMMS, so the user can fully customize the algorithm [164]. Since the AOA algorithm utilizes MILP callbacks, only CPLEX and Gurobi are available as linear subsolvers. For solving NLP problems CONOPT, IPOPT, Knitro, Minos, and SNOPT can be used.

2.4.4 BARON

License type: Commercial

Interfaces: Standalone; AIMMS, AMPL, GAMS, JuMP, MATLAB, NEOS, Pyomo, YALMIP

URL: www.minlp.com/baron

BARON (**B**ranch and **R**educe **O**ptimization **N**avigator) is a global MINLP solver developed by N.V. Sahinidis's research group [115, 165]. The solver uses a polyhedral branch and bound technique and solves LP relaxations in the BB nodes. However, BARON also uses MILP relaxations as described by [166] and [167] and nonlinear relaxations [168]. Nonconvex problems are handled by generating convex underestimators and concave overestimators combined with a spatial branch and bound technique. The solver utilizes automatic reformulations and convexity identification to decompose nonconvex functions into simpler functions with known convex or concave relaxations. The reformulations can also result in tighter lifted polyhedral outer approximations as shown by [115]. BARON also uses advanced bound tightening and range reduction techniques to reduce the search space and uses local search techniques as primal heuristics. BARON includes IPOPT, FilterSD, or FilterSQP for solving NLP subproblems, and it can also utilize any available NLP solver in GAMS. Cbc, CPLEX, or Xpress can be used as LP and MILP subsolvers.

2.4.5 BONMIN

License type: Eclipse Public License (EPL 1.0)

Interfaces: Standalone; AMPL, C++, GAMS, JuMP, MATLAB, NEOS, OS, Pyomo, YALMIP

URL: projects.coin-or.org/bonmin

BONMIN (**B**asic **O**pen-source **N**onlinear **M**ixed **In**teger Programming) is an open-source solver for MINLP problems developed by P. Bonami in a collaboration between Carnegie Mellon University and IBM Research as part of the COIN-OR initiative [129]. The solver implements several algorithms, and the user can choose between NLP-BB, LP/NLP-BB, OA, feasibility pump, OA-based branch and cut, and a hybrid approach. Some computational results and detailed descriptions of the main algorithms are given by [129] and [169]. As

subsolvers, BONMIN uses IPOPT for NLP and Cbc or CPLEX for MILP problems.

2.4.6 Couenne

License type: Eclipse Public License (EPL 1.0)

Interfaces: Standalone; AMPL, C++, GAMS, JuMP, NEOS, OS, Pyomo

URL: projects.coin-or.org/couenne

Couenne (**C**onvex **O**ver and **U**nder **E**nvelopes for **N**onlinear **E**stimation) is a global open-source solver for MINLP problems. It was developed as part of the COIN-OR initiative by P. Belotti in a collaboration between Carnegie Mellon University and IBM Research. The solver implements an LP-based spatial branch and bound technique as its main algorithm, in addition to bound reduction techniques and primal heuristics [170, 171]. Couenne features routines for calculating valid linear outer approximations of nonconvex constraints. It is currently the only global MINLP solver available in the COIN-OR Optimization Suite. Couenne uses IPOPT as NLP subsolver, CBC or CPLEX as MILP subsolver and CLP, CPLEX, Gurobi, SoPlex or Xpress as LP subsolver.

2.4.7 DICOPT

License type: Commercial

Interfaces: GAMS, NEOS

URL: www.gams.com/latest/docs/S_DICOPT.html

DICOPT (**Di**screte **C**ontinuous **Opt**imizer) is a solver based on the OA method, developed by I.E. Grossmann's research group at Carnegie Mellon University. The solver implements the equality relaxation and augmented penalty methods combined with OA [124]. In the equality relaxation, the nonlinear equality constraints are relaxed as inequalities using the signs of the corresponding Lagrangean multipliers, given by one of the NLP subproblems.

The augmented penalty method relaxes the linearizations with slack variables which are penalized in the objective of the MILP master problem of OA. Both methods are intended as heuristics for nonconvex MINLP problems. However, if the equality constraints relax as convex inequalities, the methods become rigorous. A feasibility pump algorithm is implemented as a primal heuristic to improve the solver's performance [148]. DICOPT can use any available MILP and NLP subsolvers available in GAMS.

2.4.8 Juniper

License type: Open-source (MIT)

Interfaces: JuMP

URL: www.github.com/lanl-ansi/juniper.jl

Juniper is an open-source MINLP solver implemented in Julia. It is developed by O. Kröger, C. Coffrin, H. Hijazi, and H. Nagarajan at Los Alamos National Laboratory. The solver implements an NLP-BB method with branching heuristics and primal heuristics, such as the feasibility pump [172]. The solver also uses parallelization capabilities available in Julia to solve multiple NLP subproblems in parallel. For nonconvex problems, it acts as a heuristic. It can use any NLP solver available in JuMP for solving subproblems, and it can optionally use a MIP solver for its feasibility pump.

2.4.9 Knitro

License type: Commercial

Interfaces: AIMMS, AMPL, C++, C#, Fortran, Java, JuMP, GAMS, NEOS, Pyomo, Python, YALMIP

URL: www.artelys.com/en/optimization-tools/knitro

Knitro is a commercial optimization software currently developed by Artelys [155]. Knitro

includes several algorithms for dealing with continuous problems, such as interior-point and active-set algorithms. The solver uses BB techniques for problems with discrete variables, and the solver has three methods for dealing with MINLP problems. The first method uses an NLP-BB algorithm, and the second method is based on the LP/NLP-BB algorithm. The third method, based on a sequential quadratic programming approach, is mainly intended for problems with expensive function evaluations and can handle MINLP problems where the discrete variables are not relaxable, *e.g.*, functions given by black-box simulators [173]. By using default settings, the solver will automatically choose which method to use.

2.4.10 LINDO

License type: Commercial

Interfaces: C, C++, Delphi, Excel/What's Best!, Fortran, Java, JuMP, GAMS, LINGO, MATLAB, NEOS, .NET, Ox, Python, R

URL: www.lindo.com

LINDO is a global solver developed by LINDO Systems Inc. [174]. It includes specific algorithms for solving LP, quadratic programming (QP), conic programming, semidefinite programming (SDP), and general NLP problems. For mixed-integer problems, LINDO uses a branch and cut approach [175]. The solvers deal with nonconvex problems by using reformulations and convex relaxations within a BB framework. LINDO also performs preprocessing combined with bound tightening and uses several local search techniques to find reasonable solutions quickly. Nonsmooth functions such as abs, min, floor, etc., are dealt with automatically via reformulation techniques. The solver can recognize convex quadratic, conic, and SDP terms. An option for turning off the global search strategies, *i.e.*, for convex problems, is available. To solve general nonlinear problems, LINDO requires the NLP solver CONOPT. Furthermore, if the solver Mosek is available, it efficiently solves

conic and SDP problems.

2.4.11 Minotaur

License type: Open-source

Interfaces: Standalone; AMPL, C++

URL: wiki.mcs.anl.gov/minotaur

Minotaur (**Mixed**-**I**nteger **N**onlinear **O**ptimization **T**oolkit: **A**lgorithms, **U**nderestimators, and **R**elaxations) is an open-source toolkit for solving MINLP problems developed in collaboration between Argonne National Laboratory, Indian Institute of Technology Bombay, and the University of Wisconsin-Madison [130]. It implements several different algorithms in a common framework. Currently, Minotaur has two main approaches for convex MINLP based on the NLP-BB and LP/NLP-BB algorithms. It also has a global strategy for quadratically constrained QP. Minotaur implements both a nonlinear presolver and a feasibility pump heuristic. Minotaur can use both filterSQP and IPOPT as NLP subsolvers and CLP or CPLEX as LP subsolvers.

2.4.12 Muriqui

License type: Open-source (MIT)

Interfaces: Standalone; AMPL, C++

URL: www.wendelmelo.net/software

Muriqui is an open-source MINLP solver developed by W. Melo, M. Fampa, and F. Raupp, recently presented by [176]. The solver has several algorithms implemented, *e.g.*, ECP, ESH, OA, LP/NLP-BB, and NLP-BB, as well as some heuristics approaches [177]. The solver provides a platform for using the most common algorithms for convex MINLP with several customizable parameters for the end-user. For solving the resulting MILP and NLP

subproblems, Muriqui can use CPLEX, Gurobi, Xpress [150], Mosek, Glpk [151], IPOPT, and Knitro. Muriqui also utilizes callback functionality and lazy constraints in CPLEX and Gurobi to perform the single-tree search in LP/NLP-BB and the hybrid algorithm.

2.4.13 Pavito

License type: Open-source (MPL 2.0)

Interfaces: JuMP

URL: www.github.com/juliaopt/pavito.jl

Pavito is an open-source solver for convex MINLP implemented in Julia by C. Coey, M. Lubin, and J.P. Vielma. Its functionality was previously part of the Pajarito solver for conic MINLP. However, the NLP functionality was recently moved into the Pavito solver [178]. Contrary to the other solvers presented in this chapter, Pajarito uses a conic problem formulation based on a conic Outer-approximation algorithm [42]. Conic MINLP formulations can be built using the Disciplined Convex Programming (DCP) modeling paradigm, which may require a reformulation of the test instances. Because of this, Pajarito is left out of the numerical comparison and only Pavito is included. Pavito is based on an OA algorithm and can perform a single-tree search similar to LP/NLP-BB by utilizing callbacks to the MILP subsolver. Pavito can use any MILP and NLP subsolver available in JuMP for solving subproblems.

2.4.14 SBB

License type: Commercial

Interfaces: GAMS, NEOS

URL: www.gams.com/latest/docs/S_SBB.html

SBB (**S**imple **B**ranch and **B**ound) is a solver in GAMS based on NLP-BB, developed by

ARKI Consulting and Development A/S. SBB is based on a BB method that solves nonlinear relaxations in NLP problems at each node. For nonconvex MINLP problems, it works as a heuristic approach without convergence guarantees. For improved robustness, the solver has functionality for dealing with NLP solver failures by changing either subsolver or subsolver parameters. The solver also uses primal heuristics through the GAMS Branch-Cut-and-Heuristic facility [179]. SBB can use any of the available NLP solvers in GAMS for solving the relaxed subproblems. However, it works best with solvers that take advantage of a near-optimal starting point such as CONOPT, Minos, and SNOPT.

2.4.15 SCIP

License type: free for academic use (ZIB academic license); commercial

Interfaces: Standalone; AMPL, C, GAMS, JuMP, MATLAB, NEOS, Java, Pyomo, Python

URL: `scip.zib.de`

SCIP (**S**olving **C**onstraint **I**nteger **P**rograms) was originally developed by T. Achterberg at the Zuse Institute Berlin in cooperation with TU Darmstadt, RWTH Aachen, and the University of Erlangen-Nürnberg, as a general framework based on branching for constraint integer and mixed-integer programming using branch-cut-and-price, *cf.* [180]. The solver is intended to be modular, and it utilizes plugins to make it easy to modify [181]. SCIP was extended by [182] to solve convex and nonconvex MINLP problem by utilizing polyhedral outer approximations and a spatial branch and bound technique. The solver uses LP relaxations and cutting planes to provide strong dual bounds while using Constraint Programming to handle arbitrary (nonlinear) constraints and propagation to tighten domains of variables. A variety of primal heuristics and bound tightening techniques are also utilized in the solver. SCIP includes SoPlex for solving the LP subproblems and utilizing CLP, CPLEX, Gurobi, Mosek, or XPress if available. Furthermore, the solver uses IPOPT for

solving NLP subproblems in the nonlinear strategy.

2.4.16 SHOT

License type: Open-source (EPL 2.0)

Interfaces: Standalone; C++, GAMS

URL: www.github.com/coin-or/shot

SHOT (**S**upporting **H**yperplane **O**ptimization **T**oolkit) is an open-source solver for convex MINLP developed by A. Lundell, J. Kronqvist and T. Westerlund at bo Akademi University [121, 183]. The solver utilizes polyhedral outer approximations, generated mainly by the ESH method, and iteratively constructs an equivalent MILP problem for its lower bound. For the upper bound, SHOT utilizes primal heuristics, such as solving fixed NLP problems. If either CPLEX and Gurobi are used as MILP subsolver, SHOT can use a single-tree approach similar to the LP/NLP-BB technique. The supporting hyperplanes are then dynamically added by utilizing callbacks and lazy constraints, enabling the MILP solver to continue without rebuilding the branch and bound tree. If Cbc is used as MILP subsolver, a multi-tree strategy is used. The tight integration with the MILP solvers enables SHOT to fully benefit from their cut generating procedures, advanced node selection, and branching techniques. SHOT also includes the functionality to solve MIQP subproblems with CPLEX and Gurobi. The NLP problems are solved with either IPOPT or any of the applicable solvers in GAMS. A version of SHOT with reduced functionality, *e.g.*, only utilizing the multi-tree approach, is also available for Wolfram Mathematica [184].

2.4.17 Other MINLP solvers

Besides the solvers mentioned above, there are a few others solvers capable of solving convex MINLP problems that the authors are aware of. It should be noted that the solvers

left out of the numerical comparison are not necessarily inferior compared to the other solvers. These solvers have been left out of the comparison due to one of the following reasons: not publicly available, not maintained within the last years, or unable to read the problem formats available in MINLPlib.

bnb is a MATLAB implementation of NLP-BB by K. Kuipers at the University of Groningen. The solver uses the fmincon routine in MATLAB's Optimization Toolbox for solving the integer relaxed subproblems. The MATLAB code for the solver can be downloaded from www.mathworks.com/matlabcentral/fileexchange/95-bnb.

FICO Xpress-SLP is a solver currently developed by FICO [185] and is available as both standalone binaries and a FICO Xpress-MOSEL module. The solver has an interface to Python and can be used in several other programming environments through the BCL Builder Component Library [185]. Xpress-SLP is a local solver designed for large-scale nonconvex problems, and global optimality is only guaranteed for convex problems. For general MINLP problems, the solver uses a mixed-integer successive linear programming (MISLP) approach. With the MISLP approach, it is, *e.g.*, possible to use an NLP-BB technique where the NLP subproblem at each node is solved using a successive linear programming (SLP) technique. For certain types of problems, such as convex MIQP, MIQCQCP, and MISOCP, the solver detects convexity and automatically reverts to FICO Xpress's purpose written solvers. The solver also includes some heuristic approaches for quickly obtaining solutions. More information about FICO and its solvers can be found at www.fico.com/en/products/fico-xpress-optimization.

FilMINT is an MINLP solver developed by K. Abhishek, S. Leyffer, and J. Linderoth based on the NLP/LP-BB algorithm [128]. The solver is built on top of the MILP solver MINTO

[186] and uses filterSQP [187] for solving NLP relaxations. By utilizing functionality in MINTO, FilMINT can combine the NLP/LP-BB algorithm with features frequently used by MILP solvers, such as cut generation procedures, primal heuristics, and enhanced branching and node selection rules. There is an AMPL interface available for FilMINT, and the solver can also be used through the NEOS server. For more details, we refer to [128].

fminconset is an implementation of NLP-BB in MATLAB by I. Solberg. The NLP subproblems are solved with MATLAB's fmincon routine in the Optimization Toolbox. The solver is available to download from www.mathworks.com/matlabcentral/fileexchange/96-fminconset.

GAECP (**G**eneralized **A**lpha **E**xtended **C**utting **P**lane) is a solver based on the GAECP algorithm [120] developed by T. Westerlund. The solver also uses supporting hyperplanes as in the ESH algorithm and can guarantee convergence for MINLP problems with nonsmooth pseudoconvex functions. The solver is described in detail in [188].

MILANO (**M**ixed-**I**nteger **L**inear and **N**onlinear **O**ptimizer) is a MATLAB-based MINLP solver developed by H. Y. Benson at Drexel University. There are two versions of the solver available; one uses an NLP-BB technique, and the other is based on OA. The NLP-BB technique version uses an interior point NLP solver with warm-starting capabilities described in [189]. The solver can be downloaded from www.pages.drexel.edu/~hvb22/milano.

MindtPy (**M**ixed-**I**nteger **N**onlinear **D**ecomposition **T**oolbox in **Py**omo) is an open-source software framework implemented in Python for Pyomo by the research group of I. Grossmann at Carnegie Mellon University. This toolbox implements the ECP, GBD, and OA algorithms, together with primal heuristics. It relies on Pyomo to handle the result-

ing MILP and NLP subproblems, allowing any of the solvers compatible with Pyomo to be used with MindtPy [190]. The toolbox is available in the following repository www.github.com/bernalde/pyomo/tree/mindtpy.

MINLP_BB was developed by S. Leyffer and R. Fletcher as a general solver for MINLP problems [191]. The solver is based on NLP-BB and uses filterSQP for solving the continuous relaxations. There are interfaces to AMPL and Fortran. Furthermore, the solver can also be used in MATLAB through the TOMLAB optimization environment [192]. More information about the solver is available at wiki.mcs.anl.gov/leyffer.

MINOPT (**M**ixed **I**nteger **N**onlinearl **O**ptimizer) was developed by C.A. Schweiger and C.A. Floudas as a modelling language for a wide range of optimization problems [193]. For MINLP problems MINOPT offered several algorithms, such as variants of OA and GBD.

MISQP (**M**ixed-**I**nteger **S**equential **Q**uadratic **P**rogramming) is a solver based on a modified sequential quadratic programming (SQP) algorithm for MINLP problems presented by [194]. The solver is developed by K. Schittkowski's research group and the University of Bayreuth. MISQP is intended for problems where function evaluations may be expensive, *e.g.*, where some function values are obtained by running a simulation. Unlike some of the other solvers, MISQP does not need to evaluate functions at fractional values for integer variables which can be an important property, *e.g.*, for simulation-based optimization tasks. There is an interface in MATLAB through TOMLAB as well as a standalone Fortran interface. A more detailed description of the solver is available at tomwiki.com/MISQP.

Finally, there are a few other deterministic solvers that the authors are aware of, capable of handling convex MINLP problems but mainly focusing on nonconvex MINLP. These

solvers are: Decogo (**DECO**mposition-based **G**lobal **O**ptimizer; [195]), NOMAD [196], POD (**P**iecewise convex relaxation, **O**uter-approximation, and **D**ynamic discretization; [197]), LaGO (**L**agrangian **G**lobal **O**ptimizer; [198]). For more details on nonconvex MINLP see, *e.g.,* [199], [136] and [117].

2.5 Benchmark details

The objective of the forthcoming two sections is to compare some of the convex MINLP solvers mentioned in the previous section by applying them to a comprehensive set of test problems. There are some benchmarks available in literature, *e.g.,* [121], [200], [160] and [128]. However, these are limited to only a few of the solvers considered here or used a smaller set of test problems. The goal here is to give a comprehensive, up-to-date comparison of both open-source and commercial solvers available in different environments. The main interest has been to study how the solvers perform on a desktop computer. All the benchmarks were performed on a Linux-based PC with an Intel Xeon 3.6 GHz processor with four physical cores (able to process eight threads at once) and 32 GB memory. We have allowed the solvers to use a maximum of eight threads to replicate a real-world situation to solve the problems with all the available resources. However, it is worth mentioning that the solvers and subsolvers utilize parallelism to different extents.

In the comparison, we have included three versions of BONMIN: BONMIN-OA based on OA, BONMIN-BB based on NLP-BB, and BONMIN-HYB, which is a variant of the LP/NLP-BB algorithm. We have also included two versions of Minotaur: Minotaur-QG based on the LP/NLP-BB algorithm and Minotaur-BB based on NLP-BB. Two versions of Knitro were considered: Knitro-QG based on LP/NLP-BB and Knitro-BB based on NLP-BB. These different versions of the same solver were included since they represent different

Table 2.1: The table shows which subsolvers were used with each solver, and on which platform the solver was run on.

MINLP solver	Subsolvers used		Platform
	MILP/LP	NLP	
AlphaECP 2.10.06	CPLEX 12.8	CONOPT 3.17I	GAMS 25.1.2
Antigone 1.1	CPLEX 12.8	CONOPT 3.17I	GAMS 25.1.2
AOA	CPLEX 12.8	CONOPT 3.14V	AIMMS 4.59.4.1
BARON 18.5.8	CPLEX 12.8	CONOPT 3.17I	GAMS 25.1.2
BONMIN 1.8	CPLEX 12.8	IPOPT 3.12	GAMS 25.1.2
Couenne 0.5	CLP 1.16	IPOPT 3.12	GAMS 25.1.2
DICOPT 2	CPLEX 12.8	CONOPT 3.17I	GAMS 25.1.2
Juniper 0.2.0	CPLEX 12.8	IPOPT 3.12.1	JuMP 0.18.4
Knitro 10.3.0	–	–	GAMS 25.1.2
Lindo 11.0	CPLEX 12.8	CONOPT 3.17I	GAMS 25.1.2
Minotaur 05-21-2018	CPLEX 12.6.3	filterSQP 20010817	–
Muriqui 0.7.01	CPLEX 12.8	IPOPT 3.12.1	–
Pavito 0.1.0	CPLEX 12.8	IPOPT 3.12.1	JuMP 0.18.4
SBB	CPLEX 12.8	CONOPT 3.17I	GAMS 25.1.2
SCIP 5.0	CPLEX 12.8	IPOPT 3.12	GAMS 25.1.2
SHOT 0.9.3	CPLEX 12.8	CONOPT 3.17I	GAMS 25.1.2

approaches for solving the MINLP problems, and their performance varies significantly. The solvers in the comparison are implemented in and used from different environments (GAMS, AIMMS, and Julia/JuMP), and the subsolvers available may vary. Where possible, we have tried to use CONOPT and IPOPT as the NLP solver and CPLEX as the (MI)LP solver. The linear solver used in IPOPT was MA27 [201] which is the default if available. The latest version of CPLEX (12.8) was used; the exception is Minotaur, which we only got working with version 12.6.3. Couenne warns against using another LP solver than CLP, which we respected since it improved its performance significantly. For Minotaur, filterSQP was used as an NLP subsolver as it is recommended over IPOPT and overall performed better. A complete list of solvers and subsolvers used is given in Table 2.1.

The termination criteria used with the solvers is the relative objective gap between the

upper and lower objective bounds, where we used a tolerance of 0.1% with all the solvers. Specific solver options were given to ensure that the solvers did not terminate prematurely due to other built-in termination criteria, avoiding apparent solver failures. These options are listed in Appendix B.B. Except for these, default settings were used for all solvers. Furthermore, a wall clock time limit of 900 seconds was also used with all solvers. Even with the 15-minute time limit per problem, the total running time for the experiments was more than two weeks.

2.5.1 Problem sets

The problems considered here are from the problem library MINLPLib [88]⋆, which as of July 2018 consists of 1534 instances. These instances originate from many different sources and applications, as indicated in the library. Out of the instances, we have chosen all problems that satisfy the following criteria: classified as convex, containing at least one discrete variable and some nonlinearity (either in the objective function or in the constraints). We also excluded the instance `meanvarxsc` utilizing semicontinuous variables not supported by all of the solvers. The `smallinvSNPr*` instances were also excluded since they recently lost their convex classification in MINLPLib due to rounding errors in the problem format that made them slightly nonconvex. In total, 335 instances satisfied the given criteria, and these constitute our master benchmark test set. Some statistics of the problems are available in Table 2.2.

The problems selected represent various types of optimization problems with different properties such as the number of variables, number of discrete variables, and number of nonlinear terms. Some of the problems also represent different formulations of the same problems, *e.g.*, both big-M and convex hull formulation of disjunctions. Therefore, it is of

⋆http://www.minlplib.org

Table 2.2: Statistics of the convex MINLP instances used in the benchmark

objective function type	problem count
linear objective	244
quadratic objective	66
general nonlinear objective	25

	minimum	arithmetic mean	maximum
number of discrete variables	2	93	1500
number of variables	2	989	107,222
number of constraints	0	1213	108,217
number of nonlinear constraints	0	16	112
number of nonlinear variables	1	132	4521

interest to compare the solvers not only on the entire test set but also on smaller subsets with specific properties. We have partitioned the test set into groups to represent both integer and nonlinear properties and compare both the solvers and algorithms for different problems. The following criteria were used to partition the test problems into subsets:

Continuous relaxation gap. By solving a continuous relaxation of the MINLP problem and comparing the optimal objective value of the relaxed problem with the actual optimal objective value, we can determine the continuous relaxation gap. To avoid differences due to scaling, we use a relative value calculated as

$$\text{Relative continuous relaxation gap} = \frac{|z^* - \bar{z}|}{\max\{|z^*|, 0.001\}} \cdot 100\%, \tag{2.8}$$

where z^* denotes the optimal objective value, and $\bar{z}$ denotes the optimum of the continuous relaxation. The continuous relaxation gap varies significantly for the test problems: some instances have a gap larger than 1000%, and for some instances, it is smaller than 1%. Based on the gap, given by Eq. (2.8), we have divided the test problems into two subsets: problems with a large gap ($\geq 50\%$) and problems with

a small gap (< 50%). According to this classification, there are 151 problems with a large gap (average gap 188%) and 184 with a small gap (average gap 7.2%).

Nonlinearity. Some test problems are almost linear with only a few nonlinear terms, whereas some test problems are nonlinear in each variable. The test problems are here classified based on the following nonlinearity measure

$$\text{Degree of nonlinearity} = \frac{n_{\text{nonlin}}}{n_{\text{tot}}} \cdot 100\%, \tag{2.9}$$

where n_{nonlin} is the number of variables involved in a nonlinear term and n_{tot} is the total number of variables. The test problems are divided into the following two categories: problems with high degree of nonlinearity (≥ 50%), and problems with low degree of nonlinearity (< 50%). The set with high degree of nonlinearity contains 103 problems with an average nonlinearity measure of 89%, while the set with low degree of nonlinearity contains 232 problems with an average nonlinearity measure of 14%.

Discrete density. The number of discrete variables also varies significantly in the test problems. Some problems contain only a few discrete variables, while others contain only discrete variables. To avoid a division based mainly on the problem size, we have chosen to divide the problems based on the following measure

$$\text{Discrete density} = \frac{n_{\text{int}} + n_{\text{bin}}}{n_{\text{tot}}} \cdot 100\%. \tag{2.10}$$

Here n_{int} and n_{bin} are the number of integer and binary variables, and n_{tot} is the total number of variables. Again the test problems are divided into two subsets: problems with a high discrete density (≥ 50%) and problems with a low discrete density (< 50%). The first category contains 120 problems with an average discrete density of 81%, and the second category contains 215 problems with an average density of 27%.

A list of the problems in each category is given in Appendix B.A, which also shows the continuous relaxation gap, degree of nonlinearity, and discrete density for each test problem. Scatter plots are also presented in Appendix B.A that shows there is little to no correlation between the problem categories.

2.5.2 Reporting

All the results were analyzed using PAVER [202], a tool for comparing the performance of optimization solvers and analyzing the quality of the obtained solutions. The reports generated by PAVER and all the results obtained by the individual solvers are available at `andreaslundell.github.io/minlpbenchmarks`. The parameters used for generating the reports are also available within the reports.

A comment must be made regarding the choice of the parameter `gaptol` in PAVER, which was set to the value $1.002 \cdot 10^{-3}$ instead of the value used as termination criteria ($1 \cdot 10^{-3} = 0.1\%$). The small perturbation is needed due to differences in how the solvers calculate the relative gap. Some solvers calculate the relative gap by dividing the gap by the lower bound. In contrast, others divide by the smallest absolute value of either the upper or lower bound. For example, BARON and ANTIGONE would, without the small perturbation, seem to terminate prematurely on a large number of instances, and these would all be marked as failed by PAVER.

PAVER also calculates so-called **virtual best** and **virtual worst solvers**. The virtual best solver is the best (in our graphs the fastest) successful solver selected for each problem instance. The virtual worst is then the slowest for each instance. These virtual solvers provide a good comparison for how good or bad an individual solver is compared to all the solvers.

Since MINLPLib also provides a list of known optimal objective values and upper and

lower objective bounds, PAVER can compare the obtained solutions by the known bounds in MINLPLib. PAVER is, thus, also able to calculate the so-called primal gap, *i.e.,* the difference between the obtained solution and the best-known integer solution, which can be used to analyze the quality of the obtained solutions. For example, there are cases where the solver returns the optimal solution. However, it has not been able to verify optimality within the time limit. PAVER also uses known objective bounds available in MINLPLib to check whether the solvers obtained correct solutions and bounds for the test problems.

2.6 Results

The results are presented using solution profiles showing the number of individual problems a solver can solve as a function of time. Note that the profiles do not represent the cumulative solution time but show how many individual problems the solvers can solve within a specific time. We have not used performance profiles where the time is normalized with respect to best solver [204] since these are not necessarily good for comparing several solvers as noted by [205].

In all solution profiles in this section, we have chosen to divide the solvers into two categories to make the solution profiles more easily readable. The solvers are divided into MILP decomposition-based solvers and BB-based solvers. The division is not entirely straightforward since some of the solvers could fit into both categories. However, the division is only intended to make it easier to read the results. The solvers classified as **MILP decomposition-based solvers** are AlphaECP, AOA, BONMIN-OA, DICOPT, Knitro-QG, Minotaur-QG, Muriqui, Pavito, and SHOT. Solvers classified as **BB-based solvers** are ANTIGONE, BARON, BONMIN-BB, BONMIN-HYB, Couenne, Juniper, Knitro-BB, LINDO, Minotaur-BB, SBB, and SCIP.

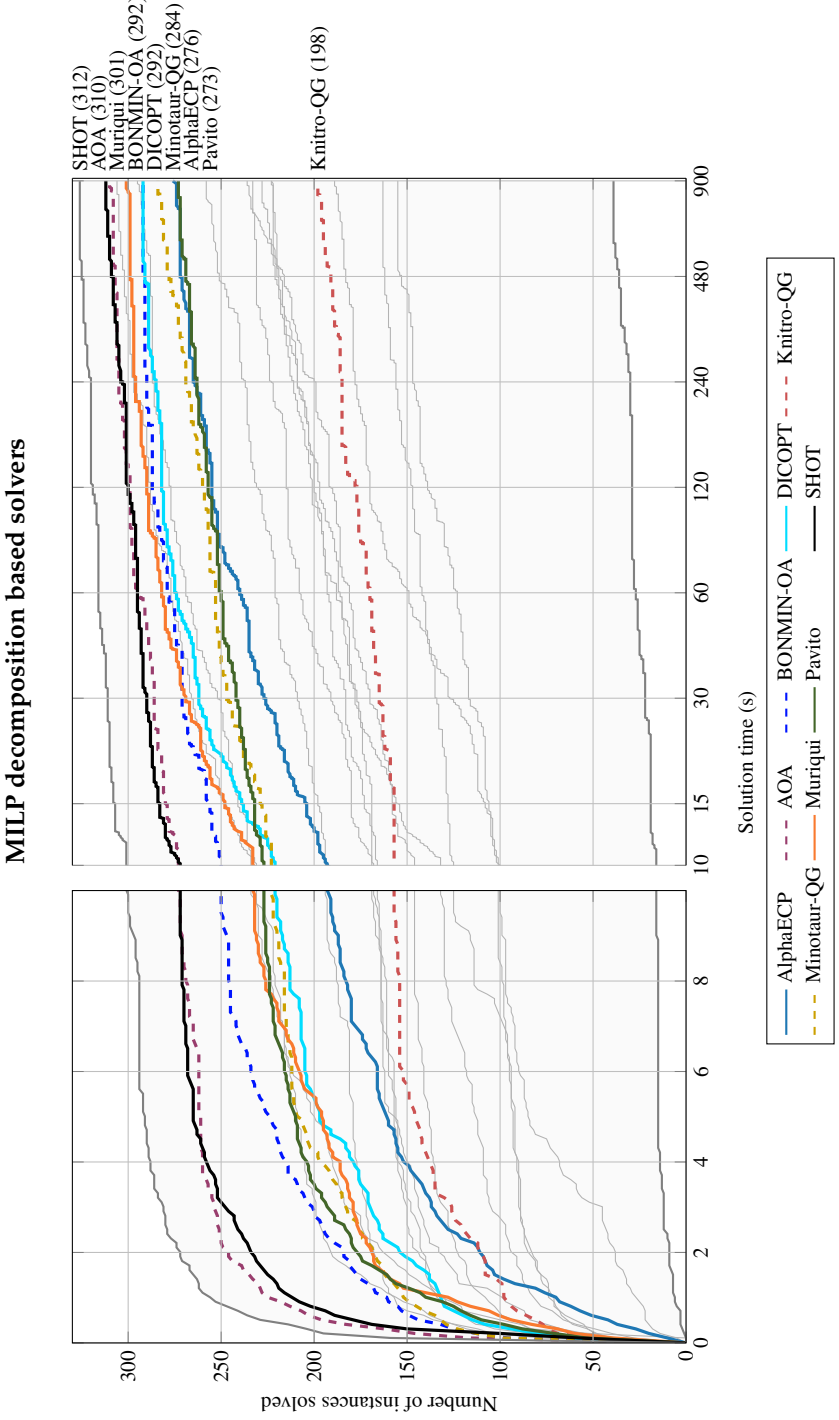


Figure 2.1: The solution profile indicates the number of solved convex MINLP instances in MINLPLib [203] as a function of time. A problem is regarded as solved if the relative objective gap, as calculated by PAVER [202], is $\leq 0.1\%$. The border lines on top/below the shaded area indicates the virtual best/worst solver.

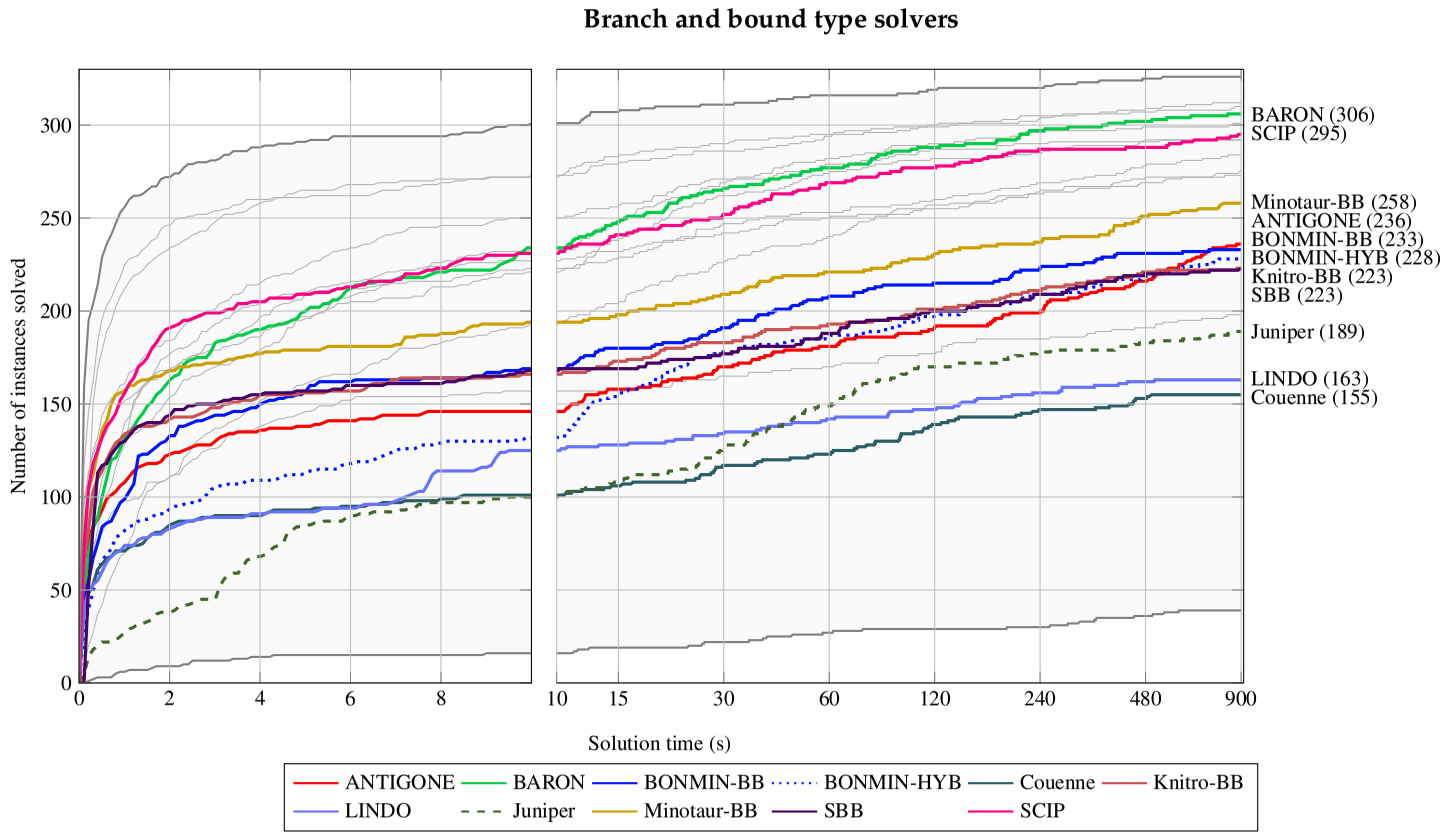


Figure 2.2: The solution profile indicates the number of solved convex MINLP instances in MINLPLib [203] as a function of time. A problem is regarded as solved if the relative objective gap, as calculated by PAVER [202], is $\leq 0.1\%$. The border lines on top/below the shaded area indicates the virtual best/worst solver.

Table 2.3: The table shows the time per problem (in seconds) for a solver to solve a certain number of instances. An empty cell in the table means the solver could not solve this many problems. For example, AOA solved 250 of the problems in under 2.2 seconds per problem, while BONMIN-HYB failed to solve this many instances at all. A problem is regarded as solved if the relative objective gap, as calculated by PAVER [202], is $\leq 0.1\%$.

Number of solved problems	25	39	50	100	150	175	200	225	250	260	270	280	290	300	305	306	310	312	326
ANTIGONE	0.1	0.2	0.3	0.7	11	38	241	598											
AOA	0.1	0.2	0.3	0.4	0.5	0.6	0.7	1.0	2.2	3.7	8.3	15	46	132	210	279	864		
AlphaECP	0.3	0.5	0.6	1.5	4.0	6.9	13	28	81	188	412								
BARON	0.1	0.2	0.3	0.5	1.6	2.6	5.0	9.3	16	22	39	78	147	361	647	824			
BONMIN-BB	0.1	0.2	0.3	1.1	4.0	12	42	288											
BONMIN-HYB	0.1	0.2	0.4	2.8	12	26	146	685											
BONMIN-OA	0.1	0.2	0.3	0.4	0.7	1.4	2.8	5.1	9.5	19	27	71	203						
Couenne	0.1	0.2	0.3	8.4	430														
DICOPT	0.1	0.2	0.3	0.4	1.9	3.5	5.4	11	21	27	49	91	457						
Juniper	0.9	2.1	3.1	9.5	61	195													
Knitro-BB	0.1	0.2	0.3	0.4	3.2	18	107												
Knitro-QG	0.1	0.2	0.3	1.4	5.7	84													
LINDO	0.1	0.2	0.3	7.2	141														
Minotaur-BB	0.1	0.2	0.3	0.4	0.7	3.5	16	90	460										
Minotaur-QG	0.1	0.2	0.3	0.4	1.0	2.5	4.3	12	38	119	277	590							
Muriqui	0.1	0.2	0.3	0.6	1.2	2.4	5.5	7.8	16	21	31	47	115	835					
Pavito	0.1	0.2	0.3	0.5	1.3	1.9	3.5	8.4	57	173	539								
SBB	0.2	0.3	0.4	0.5	2.6	25	116												
SCIP	0.1	0.2	0.3	0.4	0.9	1.5	3.2	8.2	27	39	67	141	561						
SHOT	0.1	0.2	0.3	0.4	0.5	0.6	0.8	1.5	3.1	4.3	7.1	12	27	115	267	308	538	746	
Virtual best	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.1	1.8	2.6	4.4	9.5	12	13	19	32	538
Virtual worst	40	598																	

The time scales are also divided into two parts to better highlight differences between the solvers. It is linear in the first 10 seconds and logarithmic between 10 and 900 seconds. In each plot, the solvers in the nonactive group are indicated with thin gray lines, while the others are as shown in the respective legends. The same line style is used for a specific solver in all figures. If there are several different strategies used with the same solvers, different line types (solid, dashed, dotted) are used while the color remains the same. In the profile's right margin, the solvers are ranked according to the number of solved problems (as indicated within parenthesis). The virtual best and virtual worst solvers are shown in the figures as the top and bottom thick gray lines, and the region between them is shaded.

Figures 2.1 and 2.2 show the solution profiles when applying the solvers on the complete set of test problems. As mentioned, the solution profiles indicate the number of problems that the individual solvers have solved as a function of time. A problem is defined as solved in this set of experiments if the relative objective gap, as calculated by PAVER, is $\leq 0.1002\%$ (see the note above). To better exemplify the differences between the solvers, the same data is used for generating Table 2.3 which shows "snapshots" of the solution profile for different levels of the number of solved problems.

Out of the BB-based solvers, BARON can solve the most instances (306) within the time limit, followed by SCIP (295) and Minotaur-BB (258). SHOT can solve the most problems out of the MILP decomposition-based solvers (312), closely followed by AOA (310) and Muriqui (301). SHOT and AOA are overall the fastest solvers for the test set. The virtual best solver is able to solve 326 of the problems, while the virtual worst only manages to solve 39. The virtual best and worst solvers indicate a considerable spread in the solvers' performance for different problems and highlight the importance of choosing a solver well suited for the problem type. The virtual best solver also shows that many test problems are manageable for at least one of the solvers while judging by the virtual worst solver. The

test set is challenging.

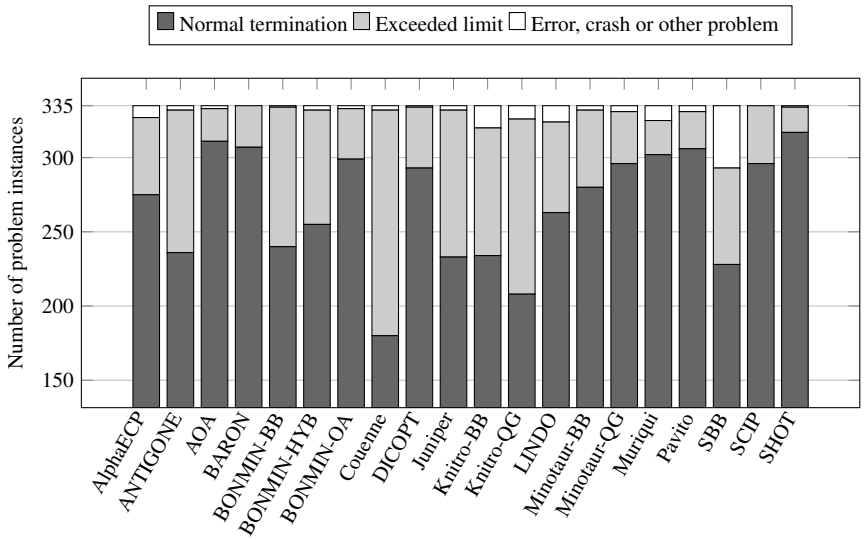


Figure 2.3: The solution status returned from the solvers.

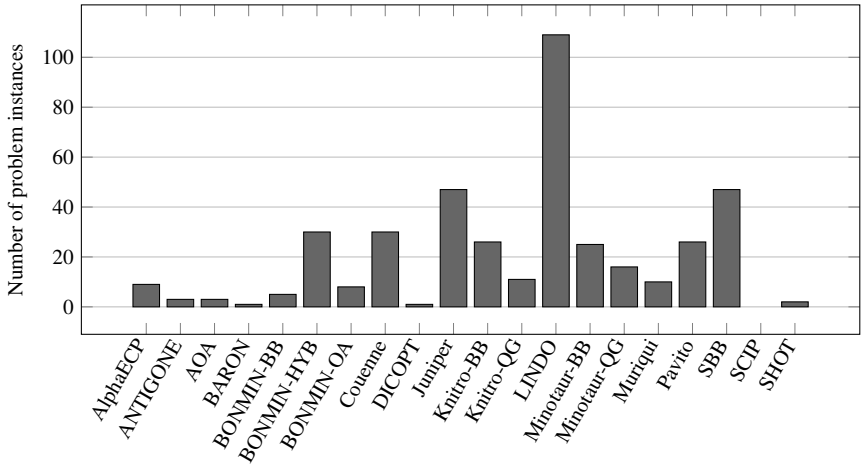


Figure 2.4: The number of solutions per solver flagged as failed by PAVER. Most often, the cause is that the returned solution is not within the bounds provided in MINLPLib.

Figure 2.3 presents statistics regarding the termination of the solvers, *e.g.*, how many errors and timeouts occurred. These values are as reported by the solver but also include solver crashes where no solution was returned. PAVER also verifies if the solver runs were

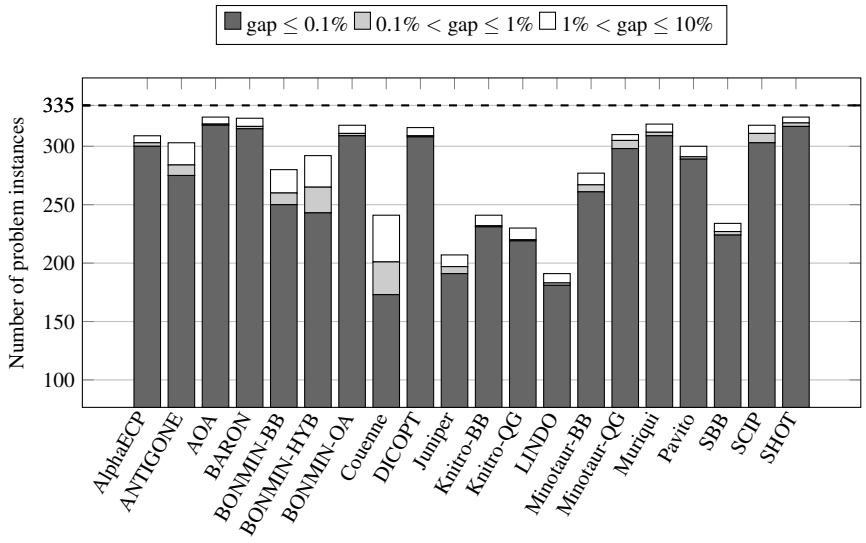


Figure 2.5: The number of instances in the benchmark where the solvers found a a solution within 0.1%, 1% and 10% of the best known objective value.

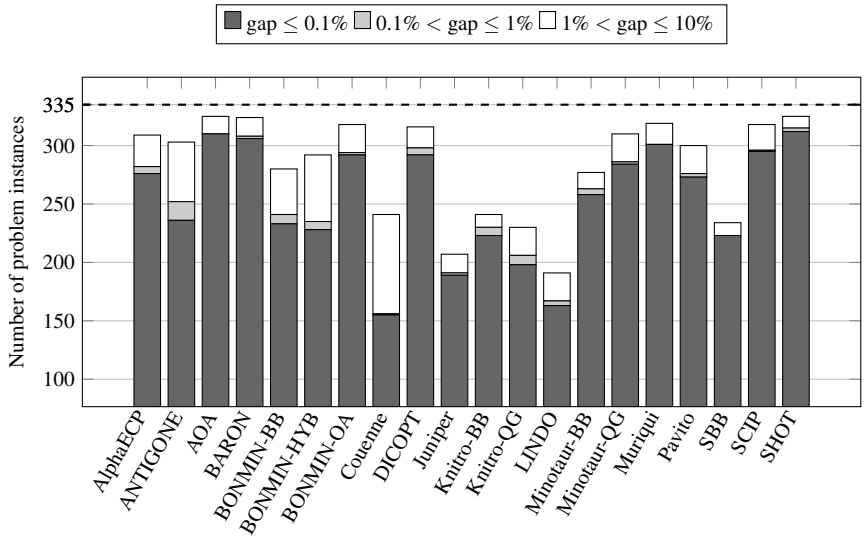


Figure 2.6: The number of instances in the benchmark where the solvers obtained an objective gap of 0.1%, 1% and 10%.

completed successfully, *e.g.,* by comparing the objective values returned to known values or bounds; if there is a discrepancy, these instances are given the status failed. Statistics on

instances marked as failed are shown in Figure 2.4.

Figure 2.5 shows the number of problems where the solver was able to obtain a solution within 0.1% and 1% of the best-known solution, but not necessarily able to verify optimality. The figure shows that none of the solvers could obtain a solution within 1% of the best-known solution for all of the problems, given the 900 second time limit. For example, BARON could obtain a solution within 1% of the optimum for 317 problems. SHOT obtained such a solution for 320 problems.

The number of instances solved to a relative objective gap, *i.e.*, difference between the upper and lower objective bound, of 0.1%, 1% and 10%, per solver is shown in Figure 2.6. By comparing Figures 2.5 and 2.6, it can be observed that some of the solvers can obtain a solution within 0.1% of the optimum to significantly more problems than they can verify as optimal. For example, AlphaECP and ANTIGONE seem to be struggling with obtaining a tight lower bound for some of the problems since they can obtain solutions within 0.1% for 300 (AlphaECP) and 275 (ANTIGONE) problems, while only verifying optimality for 276 and 236 instances respectively. Since ANTIGONE is a global solver without a user-selected convex strategy, it might fail to recognize some of the problems as convex and generate weaker relaxations. This would explain the difference between the number of optimal solutions found and the number of solutions verified as optimal.

Since it may be difficult to draw more detailed conclusions from the results in Figures 2.1 and 2.2, the next sections consider subsets of test problems with specific properties. A summary of the results for the different subsets is given in Section 2.6.4.

2.6.1 Impact of the continuous relaxation gap

This section considers problems with a large continuous relaxation gap and problems with a small continuous relaxation gap. Figure 2.7 shows the solution profiles of the solvers for

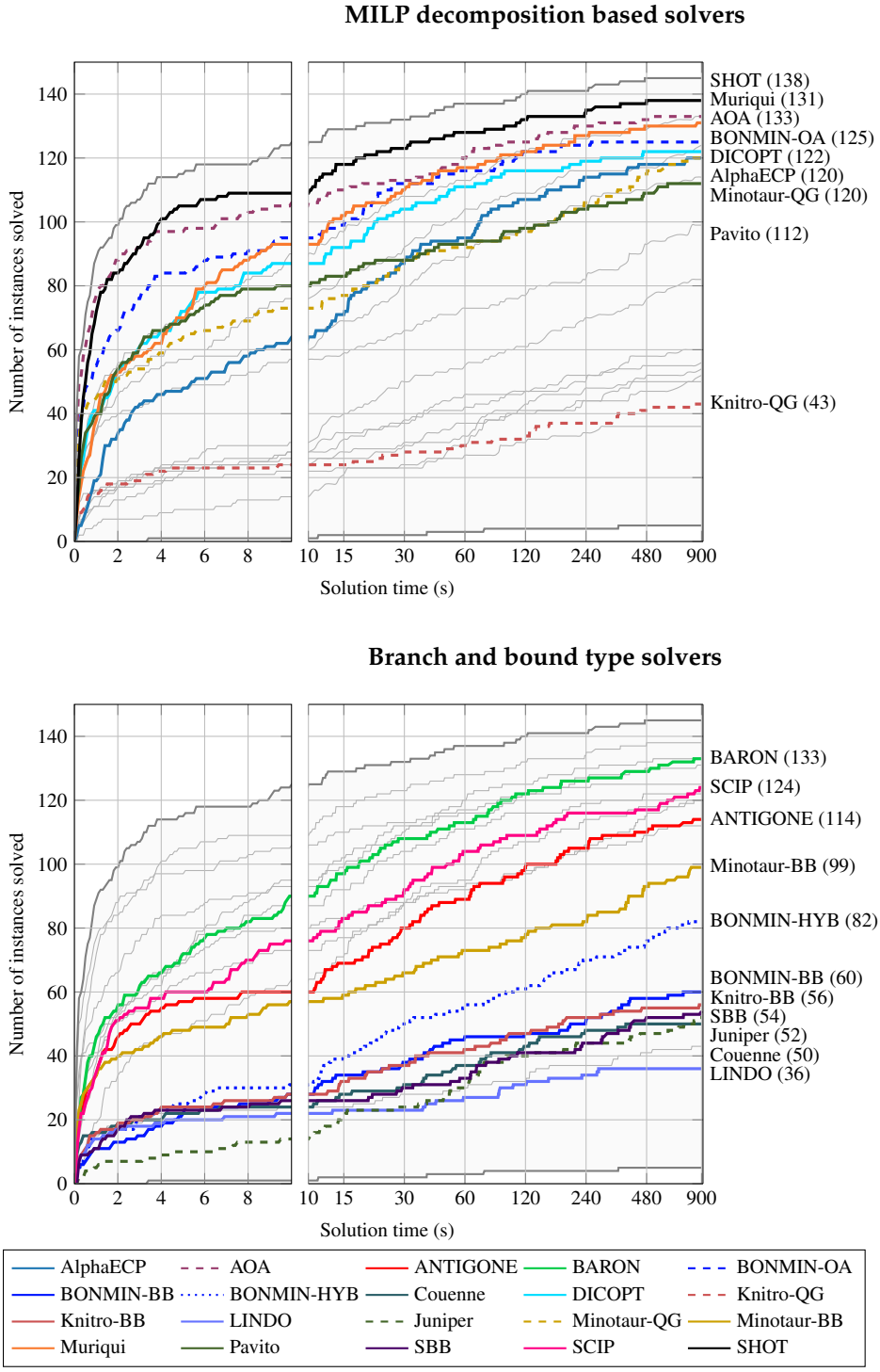


Figure 2.7: The solution profiles for problem instances with a high continuous relaxation gap as indicated in Appendix B.A.

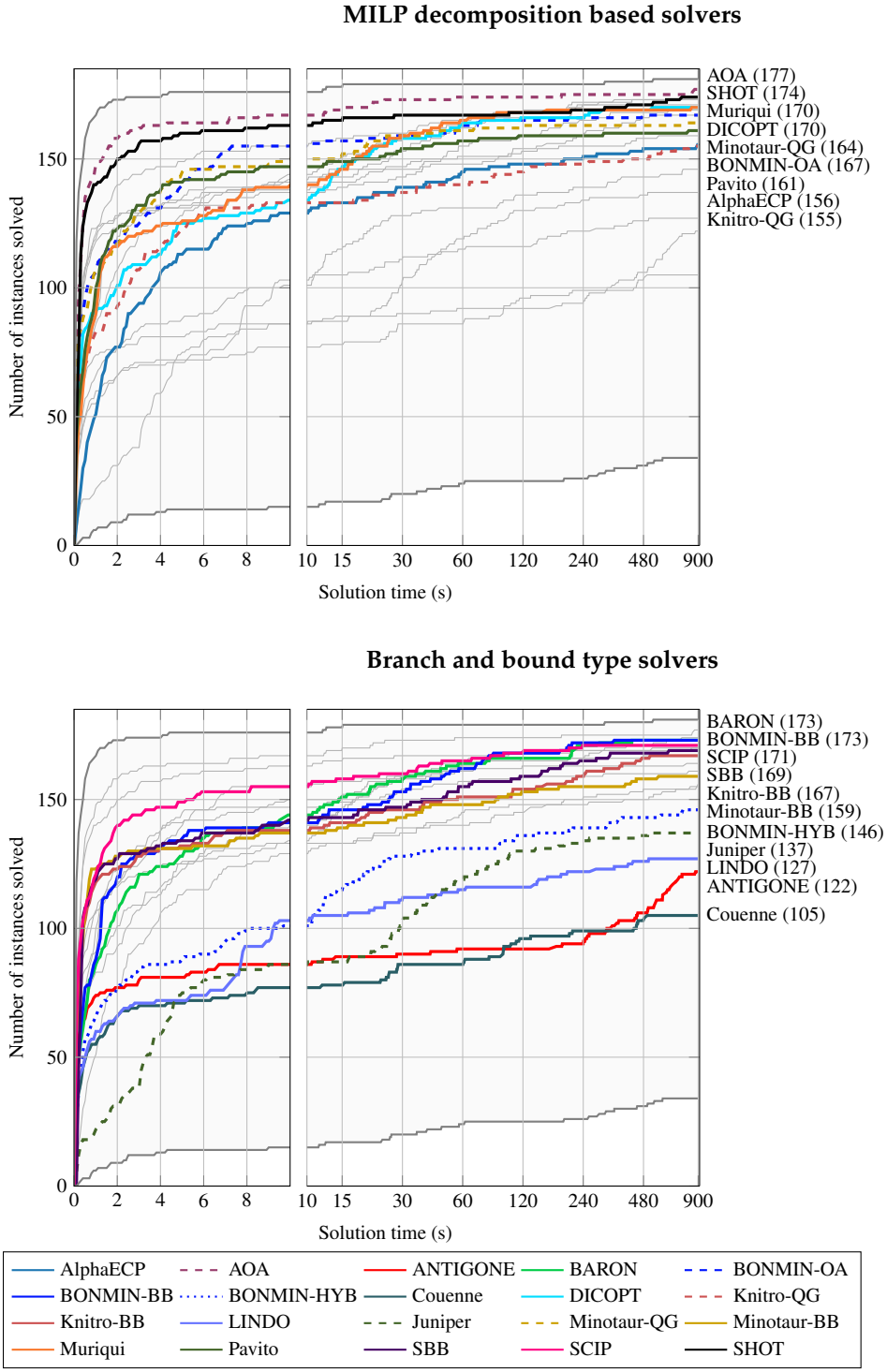


Figure 2.8: The solution profiles for problem instances with a low continuous relaxation gap as indicated in Appendix B.A.

the problems with a large gap, and Figure 2.8 shows the solution profiles for those with a small gap. By comparing the figures, there is a clear difference for the solvers based on a BB approach. These perform better on the problems with a small gap than those with a large continuous relaxation gap. For example, BONMIN-BB is one of the most efficient solvers for problems with a small gap in speed and number of solved problems. At the same time, it is outperformed by several solvers for problems with a large gap.

The NLP-BB-based solvers, BONMIN-BB, Juniper, Knitro-BB, Minotaur-BB, and SBB, solve significantly fewer problems with a large gap than the solvers based on either an ECP, ESH, or OA (AlphaECP, BONMIN-OA, DICOPT, and SHOT). The BB trees may become larger for problems with a large continuous relaxation gap, and the more expensive subproblems in each node may make the NLP-BB-based solvers suffer performance-wise. Using a polyhedral approximation within a BB framework, BARON and SCIP are not as strongly affected by the relaxation gap; this could partially be due to having simpler subproblems at each node.

Overall, the MILP decomposition-based solvers seem to be less affected by the continuous relaxation gap than the BB-based ones. Several of the MILP decomposition-based solvers, such as AOA and SHOT, are closely integrated with the MILP subsolver (CPLEX in this case) and rely on it for handling the integer requirements. This close integration enables the usage of several advanced features from the more mature MILP solvers. In contrast, NLP-BB-based solvers often need to manage the branching, handling the BB tree, cut generation, and other techniques on their own. One can expect this advantage to be more critical for problems that are challenging due to the integer requirements, which is often a trait of problems with large continuous relaxation gaps. As an example of the impact on the performance of an MILP decomposition-based solver that handles the integer requirements itself, consider Minotaur-QG (as it only uses CPLEX as an LP subsolver): The performance

difference between Minotaur-QG and AOA are significant, even though they utilize the same basic algorithm.

2.6.2 Impact of nonlinearity

In this section, problem types with a high and low degree of nonlinearity are compared, and the results are shown in Figures 2.9 and 2.10. Several solvers use linearizations to approximate the nonlinear functions in some steps of the solution procedure, whereas solvers using an NLP-BB approach directly treat the nonlinearity in each node of the BB tree. As expected, most of the solvers utilizing linearizations perform significantly better on the problems with a low degree of nonlinearity since BONMIN-OA, DICOPT, and SCIP are among the most efficient solvers in terms of both speed and number of problems solved. However, for problems with a high degree of nonlinearity, they are outperformed by the NLP-BB-based solvers BONMIN-BB, Knitro-BB, Minotaur-BB, and SBB.

SHOT and the LP/NLP-BB-based solvers AOA, Minotaur-QG, and Muriqui have quite similar behavior for both types of problems and perform well in both categories. These solvers rely on linearizations of the nonlinear constraints. Thus, one would expect them to be negatively affected by the degree of nonlinearity. However, they all use a relatively similar single-tree approach where NLP subproblems are solved in some of the nodes, which may help them cope with a high degree of nonlinearity. The LP/NLP-BB-based solver Knitro-QG also performs exceptionally well for problems with a high degree of nonlinearity.

Most affected by the degree of nonlinearity seem to be the NLP-BB-based solvers. For problems with a high degree of nonlinearity, they performed well, with several of the NLP-BB-based solvers being among the most efficient ones. Thus, in this case, there seems to be a clear advantage of directly treating the nonlinearities. However, for the problems with a

low degree of nonlinearity, the NLP-BB-based solvers did not perform as well compared to the other solvers.

2.6.3 Impact of discrete density

Finally, we compare how the solvers are affected by the relative number of discrete variables, *i.e.,* integer, and binary variables. Figures 2.11 and 2.12 show how the solvers perform for problems with high and low discrete density.

The MILP decomposition-based solvers perform similarly for both types of problems. No obvious conclusions can be drawn from the results. However, again there is a clear difference for the NLP-BB-based solvers. Surprisingly, many of these solvers performed better than the other ones on the high discrete density set of problems. One could expect a correlation between the discrete density and the continuous relaxation gap. A high discrete density would result in a high continuous relaxation gap. However, as shown in Figure B.1 in Appendix B.A, there is no correlation between the two for this set of test problems. Thus, by analyzing the results, there is no clear reason why the NLP-BB-based solvers perform better for problems with a high discrete density. However, one should keep in mind that the test set is somewhat limited.

Both BARON and SHOT perform well on both sets of test problems while performing somewhat better on problems with a low discrete density. The OA approach seems to be well suited for problems with a low discrete density. DICOPT is one of the most efficient solvers, and BONMIN-OA also manages to solve a large portion of the problems. AOA and Muriqui, integrating the LP/NLP-BB method through callbacks and lazy constraints with the MILP solver, also perform well on both categories.

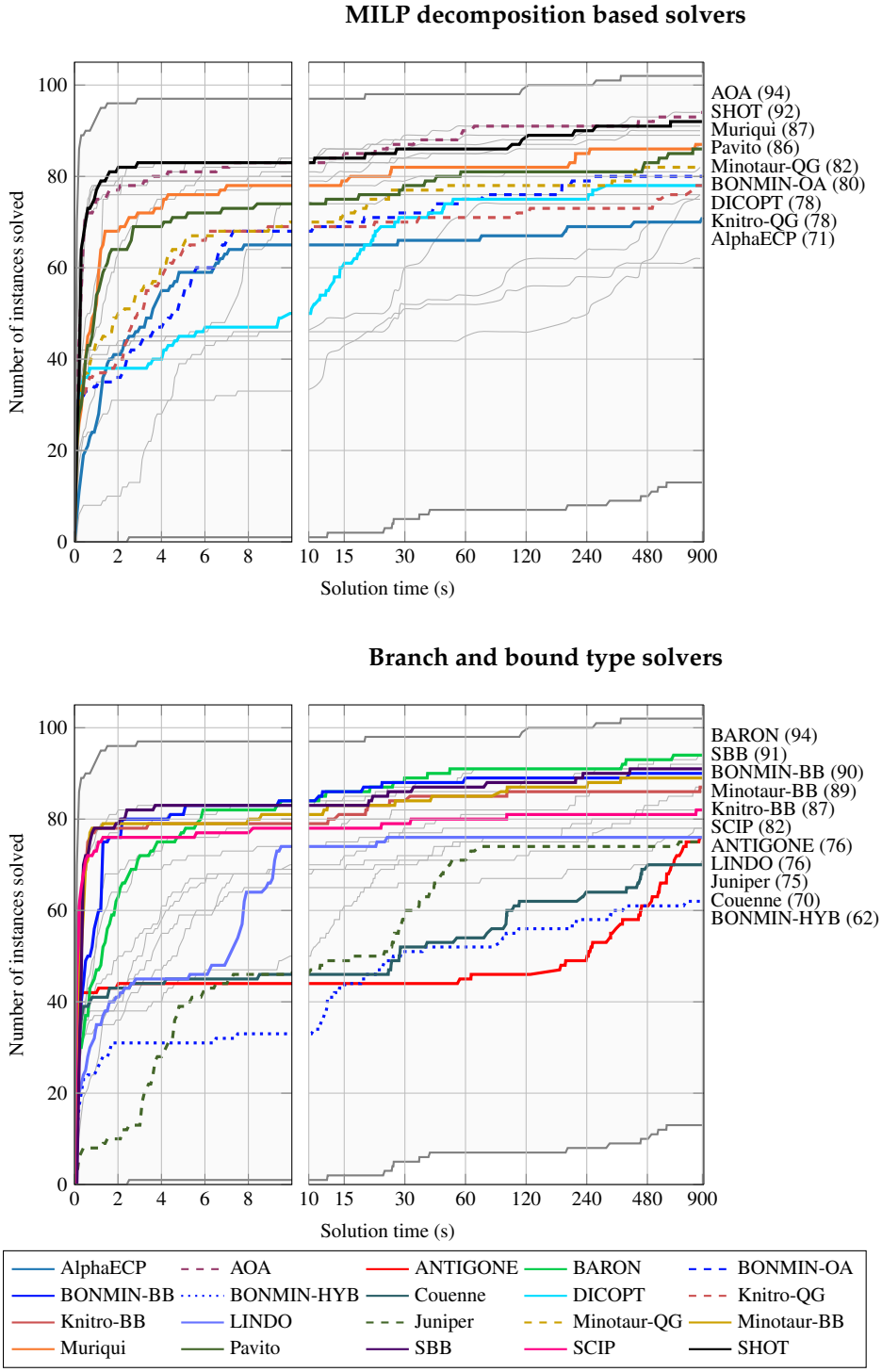


Figure 2.9: The solution profiles for problem instances with a high level of nonlinear variables as indicated in Appendix B.A.

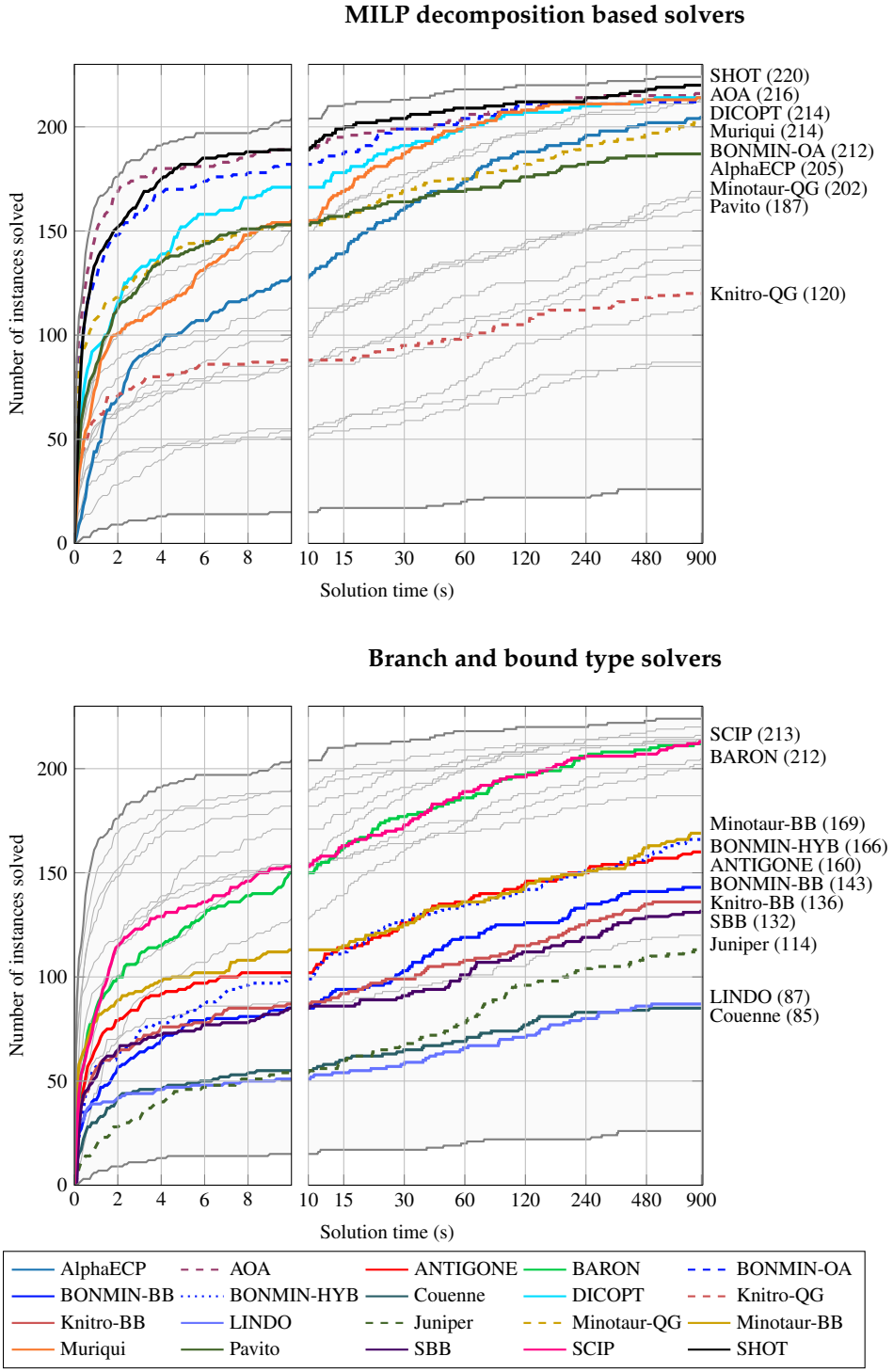


Figure 2.10: The solution profiles for problem instances with a low level of nonlinear variables as indicated in Appendix B.A.

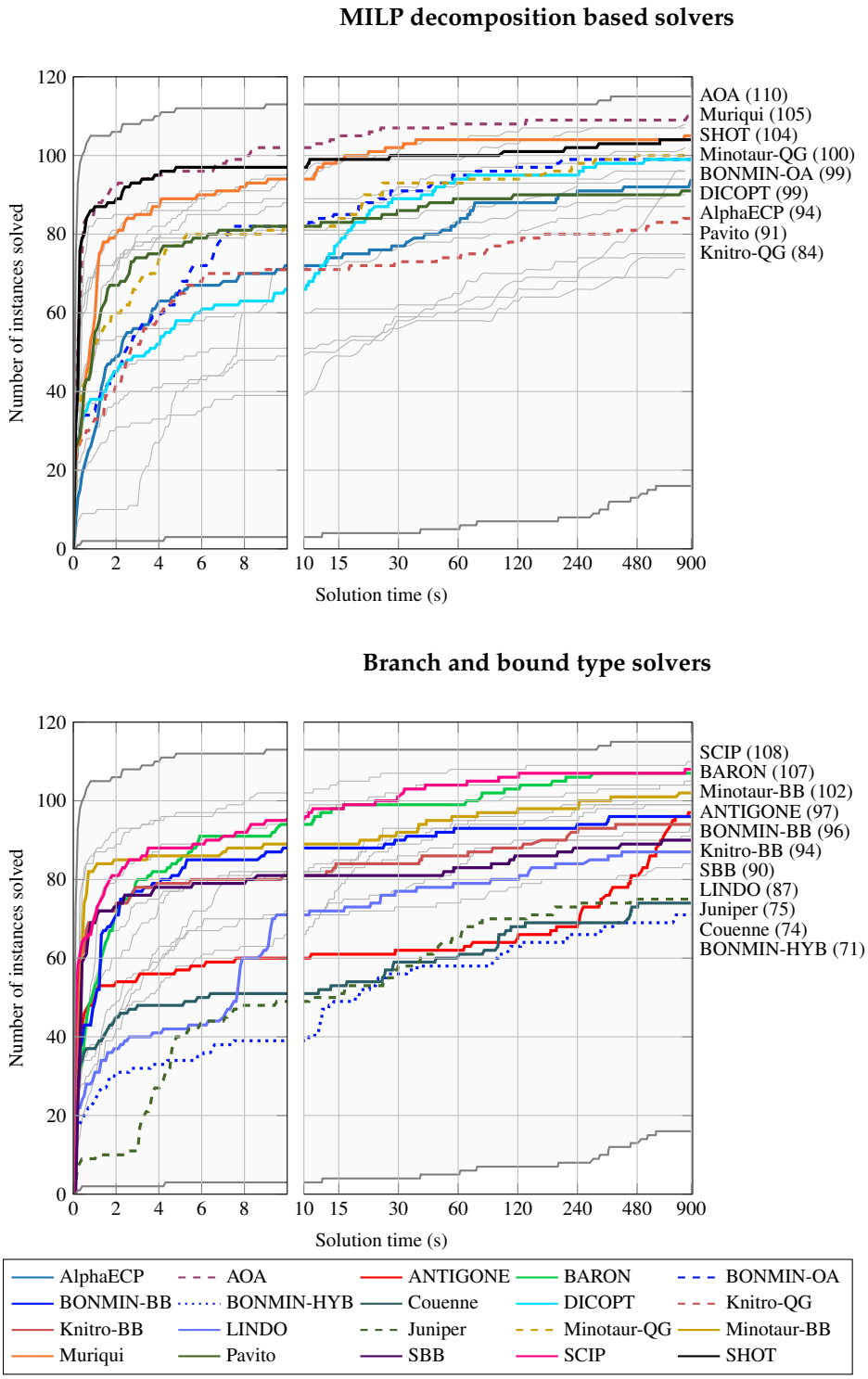


Figure 2.11: The solution profiles for problem instances with a high level of discrete variables as indicated in Appendix B.A.

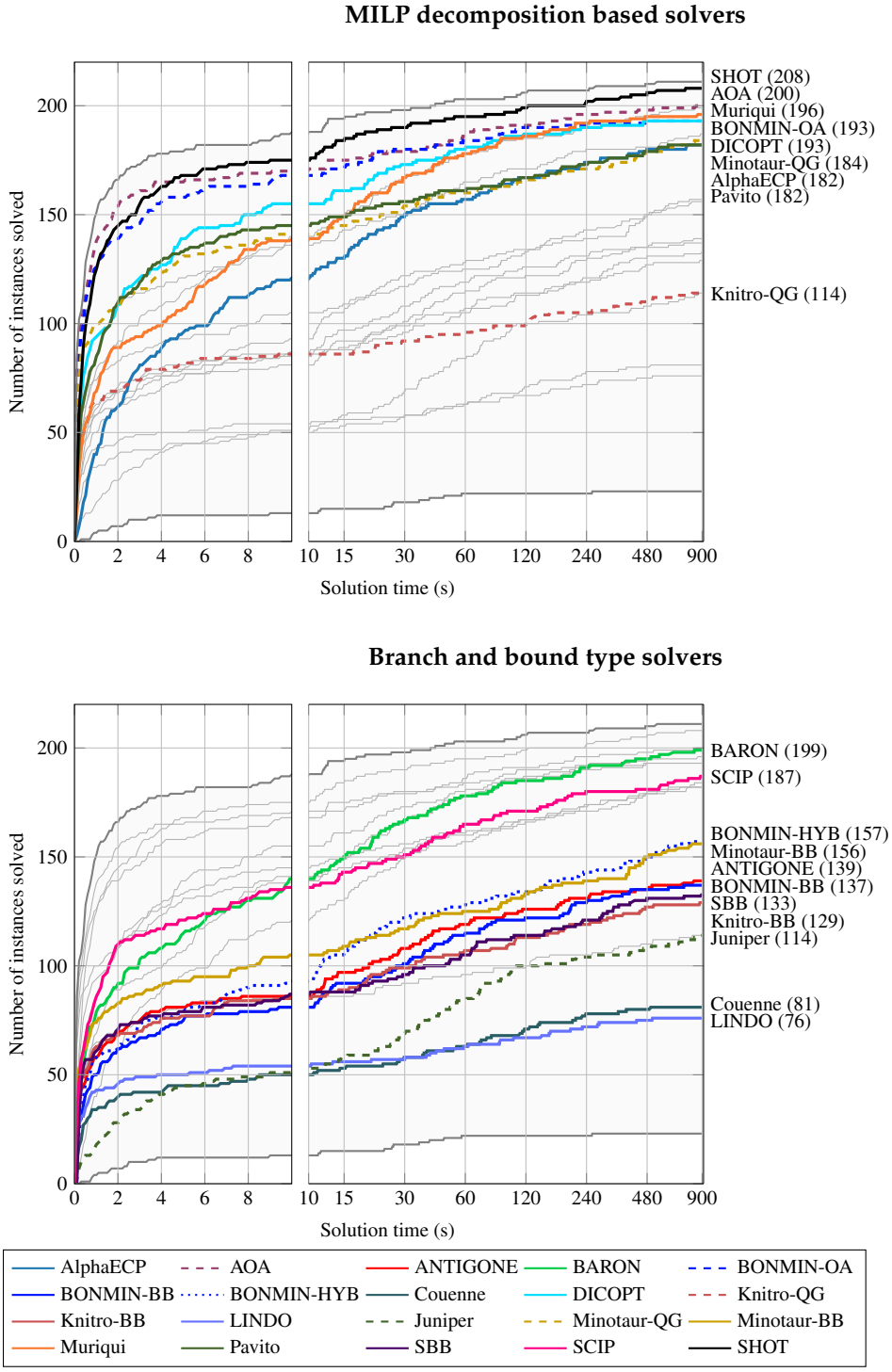


Figure 2.12: The solution profiles for problem instances with a low level of discrete variables as indicated in Appendix B.A.

2.6.4 Summary of the results

The solvers are affected by the continuous relaxation gap, degree of nonlinearity and discrete density summarized in Table 2.4. The table shows the number of problems solved within each category and an indicator of how the solvers' performances were affected by the specific properties. The performance indicator tries to show how the performance of a solver is affected by the problem properties with respect to the other solvers. Suppose a solver clearly performed better in a category with respect to speed and number of solved problems. In that case, it is indicated by '+', and similarly, '–' indicates that the solver performed worse for that category of problems. If the performance is similar within both categories, it is indicated by '∼'. Each performance indicator has been chosen by comparing how a specific solver performed in both the high and low categories with respect to the other solvers. Note that a '–' sign does not necessarily indicate that the solver performed poorly; it simply states that the solver did not perform as well as in the other category. For example, for problems with a low degree of nonlinearity, DICOPT is one of the fastest solvers. For the problems with a high degree of nonlinearity, DICOPT also performs well, but not in the other category. This is indicated by a '+' and '–' sign in Table 2.4. That there are no significant changes for the performance of AOA concerning both categories is indicated by '∼' sign.

These indicators were obtained by carefully analyzing the performance profiles. They are not intended as a grade of the solver but to show how it is affected by different problem properties. The results presented in Table 2.4 indicates that BB-based solvers seem to be more affected by the problem properties considered here compared to the MILP decomposition-based solvers. One possible explanation for the differences between the two types of solvers is that several MILP decomposition-based solvers rely heavily on the MILP subsolver and benefit from several advanced features from the MILP solver.

Comparing the global solvers (ANTIGONE, BARON, Couenne, LINDO, and SCIP) with the convex solvers is not entirely fair since the global solvers can solve a broader class of problems. In the numerical comparison, one should keep in mind that these global solvers are intended for different (more general) types of problems. Some of these solvers do not have a convex option, and thus, they have access to less information about the problem and might treat it as nonconvex. For example, the performance difference of ANTIGONE and Couenne compared to BARON and SCIP may be explained with the solvers treating some of the convex functions as nonconvex, and therefore, generate unnecessarily weak relaxations. BARON seems to be very efficient at identifying convex problems since it can deal with the problems in the benchmark set in such an efficient manner. Even if it is a global solver capable of handling various nonconvex problems, it is also one of the most efficient solvers for convex problems.

Furthermore, one should not draw any conclusions on how the solvers perform on nonconvex problems based solely on the results presented in this chapter. For example, the convex strategies in AOA and SHOT, the two most efficient solvers for this set of problems, do not necessarily work well for nonconvex problems. There is another strategy in AOA intended as a heuristic for nonconvex problems. Some of the nonglobal solvers may work quite well as heuristics for nonconvex problems, of course, without any guarantee of finding the global or even a feasible solution.

2.7 Conclusions

The comparisons presented in this chapter are mainly intended to help the readers make informed decisions about which tools to use when dealing with different types of convex MINLP problems. In the previous sections, we have shown how 16 different solvers

performed on a test set containing 335 MINLP instances. By comparing the solvers on MINLP instances with different properties, we noticed significant differences in the solvers' performance. For example, the solvers based on NLP-BB were strongly affected by both the continuous relaxation gap and the degree of nonlinearity. Several solvers are based on the same main algorithms. However, they differ significantly in terms of speed and number of problems solved. The differences are mainly due to different degrees of preprocessing, primal heuristics, cut generation procedures, and different strategies used by the solvers. The performance differences highlight the importance of such techniques for efficient solver implementation.

For the test set considered here, SHOT and AOA were the overall fastest solvers. Both solvers are based on a single-tree approach closely integrated with the MILP solver by utilizing callbacks to add the linearizations as lazy constraints. The results show the benefits of such a solution technique and support the firm belief in the single-tree approach by [128] and [104]. The close integration with the MILP solver allows AOA and SHOT to benefit from different techniques integrated within the MILP solver, such as branching heuristics, cut generation procedures, and bound tightening.

Overall, several of the solvers performed well on the test set and solved a large portion of the problems. The most instances any solver could solve within the time limit were 312 instances. By combining all the solvers, we solved 326 of the 335 MINLP problems to a 0.1% guaranteed optimality gap. However, it should be noted that many of the test instances are pretty small and straightforward compared to industry-relevant problems. Still today, real-world problems must often be simplified and reduced in size to obtain tractable formulations, in the process limiting the practical benefits of MINLP. Thus, to fully benefit from convex MINLP as a tool for design and decision-making, further algorithmic research and solver software development is required. We also hope that this chapter encourages

MINLP users to submit their problems to the instances libraries, *e.g.,* MINLPLib and www.minlp.org, to benefit both MINLP solver developers and end-users. Finally, we want to comment on mixed-integer disciplined convex programming (MIDCP). Pajarito is a tool mainly developed to deal with MICP problems using a disciplined convex programming (DCP) approach, a different modeling paradigm based on a different problem formulation. The MIDCP formulation enables Pajarito to utilize lifted problem formulations, resulting in tighter approximations. Here we have not used the MIDCP problem formulation but the standard convex MINLP formulation. Reformulating the problems as a MIDCP problem has been shown beneficial for Pajarito [42]. However, such formulations were not considered here. At the moment, the solver does not have the functionality to reformulate the problems automatically.

Finally, we want to share some ideas regarding future work within this field. Comparing the traditional convex MINLP problem formulations and a MIDCP formulation to investigate the advantages and disadvantages of this modeling framework with respect to traditional nonlinear modeling could be an exciting contribution to the existing literature. However, such comparisons did not fit into the scope of this paper.

Table 2.4: The table shows how the solvers are affected by the problem properties described in Section 5.1. Suppose a specific solver performs better for one of the categories. In that case, it is indicated by a '+' sign, and a '–' sign indicates that the solver performs worse on that specific category. If the solver performs similarly on both categories, it is indicated by '∼'. Furthermore, the number in each row shows the total number of problems that the solver was able to solve within a relative objective gap of 0.1% within 900 seconds.

	Integer relaxation gap				Nonlinearity				Discrete density			
MINLP solver	high		low		high		low		high		low	
AlphaECP	∼	120	∼	156	–	71	+	205	∼	94	∼	182
ANTIGONE	+	114	–	122	∼	76	∼	160	+	97	–	139
AOA	∼	133	∼	177	∼	94	∼	216	∼	110	∼	200
BARON	∼	133	∼	173	∼	94	∼	212	∼	107	∼	199
BONMIN-BB	–	60	+	173	+	90	–	143	+	96	–	137
BONMIN-OA	∼	125	∼	167	–	80	+	212	∼	99	∼	193
BONMIN-HYB	–	82	+	146	–	62	+	166	–	71	+	157
Couenne	∼	50	∼	105	∼	70	∼	85	∼	74	∼	81
DICOPT	∼	122	∼	170	–	78	+	214	∼	99	∼	193
Juniper	–	52	+	137	+	75	–	114	∼	75	∼	114
Knitro-BB	–	56	+	167	+	87	–	136	+	94	–	129
Knitro-QG	–	43	+	155	+	78	–	120	+	84	–	114
LINDO	–	36	+	127	+	76	–	87	+	87	–	76
Minotaur-QG	∼	120	∼	164	∼	82	∼	202	∼	100	∼	184
Minotaur-BB	–	99	+	159	+	89	–	169	+	102	–	156
Muriqui	∼	131	∼	170	∼	87	∼	214	∼	105	∼	196
Pavito	–	112	+	161	–	86	+	187	∼	91	∼	182
SBB	–	54	+	169	+	91	–	132	+	90	–	133
SCIP	∼	124	∼	171	–	82	+	213	+	108	–	187
SHOT	∼	138	∼	174	∼	92	∼	220	–	104	+	208
Number of problems		151		183		103		232		120		215

Chapter 3

Feasibility Pump implementation in DICOPT*

3.1 Introduction

The capabilities of the algorithms designed to solve mathematical programming problems are continuously improving. This allows solving increasingly larger and more complex problems. Efficient solutions of mixed-integer linear programs (MIP) and nonlinear programs (NLP) enable the solution of mixed-integer nonlinear programs (MINLP). These problems are of great interest in chemical engineering and many other areas as they combine integer variables (like discrete choices in superstructures or networks) with nonlinear constraints (for example, posynomial equations in resource allocation for scheduling and convex reformulations of horizon time constraints in the design of multiproduct batch processes) [**LeLe12**, 5, 129, 207, 208]. The general form of an MINLP is

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}), \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \\ & \mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{y} \in \mathbb{Z}^{n_y}, \end{aligned} \tag{MINLP}$$

where $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}$ is the objective function and at least one of the constraints $\mathbf{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \to \mathbb{R}^m$ or the objective function itself is nonlinear. MINLP models are generally nonconvex due to the discrete nature of $\mathbf{y}$ and possible nonconvexity of f and $\mathbf{g}$. Models in

*Published as: David E. Bernal, Stefan Vigerske, Francisco Trespalacios, and Ignacio E. Grossmann. "Improving the performance of DICOPT in convex MINLP problems using a feasibility pump". *Optimization Methods and Software* 0.0 (2019), pp. 1–20.

which f and g_i, $i = 1,\ldots,m$, are convex, are denoted as convex MINLP problems. We denote by $(\mathbf{x}^*, \mathbf{y}^*)$ an optimal solution of the MINLP if it exists.

DICOPT (Discrete Continuous Optimizer) is an MINLP solver that has been developed in 1988. It combines the outer-approximation method [122] with equality relaxation and augmented penalty [124]. The algorithm decomposes the MINLP into an NLP subproblem defined by fixing the discrete variables in the MINLP and a MIP approximation defined by linearizations of the nonlinear functions in the MINLP. The MIP and the NLP are solved alternately, whereby the MIP approximation provides values for fixing the discrete variables in the NLP, and the NLP subproblem provides feasible solutions to the MINLP and cutting planes to improve the MIP approximation. If the MINLP is convex, then this MIP approximation is a relaxation of the MINLP, thus providing a lower bound to its optimal value, and the NLP subproblems can be solved to global optimality yielding an upper bound. By adding additional inequalities to the MIP approximation, one can further ensure that any fixed values for the discrete variables are evaluated by an NLP at most once. Therefore, for a convex MINLP, a possible stopping criterion is that the bound defined by the last MIP approximation exceeds the objective value of the best-found solution [124].

For some problems, DICOPT has difficulty in finding a feasible solution. This mainly because, by default, and to address nonconvex problems, DICOPT omits linearizations of nonlinearities from infeasible NLPs into the MIP. Instead, it only excludes the infeasible fixed integer variables in the MIP and resolves them. Furthermore, even if linearizations are included for infeasible NLPs, which are valid for convex MINLPs, DICOPT still has difficulties finding a feasible solution for some problems. This results in slow progress compared to the case where feasible MINLP solutions are found early in the search.

To quickly find initial feasible solutions for convex MINLPs, an implementation of a feasibility pump [209, 210] has been incorporated into DICOPT as described in this paper.

The feasibility pump is similar to the outer-approximation algorithm, but its focus is on finding feasible solutions. As with outer-approximation, the main idea of the feasibility pump is to decompose the original MINLP problem into a MIP and an NLP. The MIP problem yields solutions that satisfy integrality requirements ($\mathbf{y} \in \mathbb{Z}^{n_y}$) but may violate nonlinear constraints, while the solutions of the NLP problems satisfy the constraints $\mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}$ but may violate integrality requirements. Contrary to outer-approximation, both MIP and NLP are defined over relaxations of the feasible area of the original MINLP. By alternately projecting onto the MIP and NLP relaxations, a solution is obtained that is feasible for both relaxations and thus for the MINLP itself if specific constraint qualifications are satisfied [210]. The feasibility pump can also be used as a standalone solver for convex MINLPs by including a bound (cutoff-value) to the objective function, which is set to the objective value of the best-known solution, reduced by a desired δ-improvement. Applying this modified feasibility pump repeatedly until the MIP becomes infeasible finds a δ-optimal solution of a convex MINLP [210]. The drawback of this algorithm is that it may require many iterations since only a δ-improvement of the objective function is enforced at each iteration.

In this work, a feasibility pump is added to DICOPT. It is used as an initialization of the outer-approximation method. In the feasibility pump, improvements in the objective function are enforced at each iteration. After the method finishes, the cuts that define the MIP relaxation of the feasibility pump and the best-found solution are passed on to the outer-approximation method to find better solutions, if any, and prove optimality. The described extension of DICOPT has been available in GAMS* since version 24.5. We present computational results of the new method on a set of convex MINLP problems and show that it outperforms the previous version of DICOPT.

*http://www.gams.com/latest

This paper is organized as follows. Section 3.2 provides an overview of the outer-approximation and the feasibility pump algorithms for convex MINLP problems. Section 3.3 describes the algorithm proposed in this work. This algorithm uses the feasibility pump as initialization for the outer-approximation algorithm. An illustrative example and computational results are presented in Section 3.4.

3.2 Background

In the following, we summarize the outer-approximation algorithm [122], the feasibility pump algorithm [210], and a hybrid of both algorithms. These algorithms are intended to solve problems of the form MINLP. The following assumptions are made [122, 211]:

(A1) The set of constraints $\mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}$ includes lower and upper bounds for every integer variable.

(A2) The constraint functions $\mathbf{g}(\mathbf{x},\mathbf{y})$ and the objective function $f(\mathbf{x},\mathbf{y})$ are continuously differentiable and convex on the set that is defined by the variable bounds (if present, otherwise $\mathbb{R}$).

(A3) The continuous relaxation of the MINLP obtained by removing the integrality requirement $\mathbf{y} \in \mathbb{Z}^{n_y}$ is bounded.

3.2.1 Outer-approximation algorithm

The outer-approximation algorithm was proposed by Duran and Grossmann in 1986 [122]. In the original version of the algorithm, the starting point was given by some fixed values for the binary variables $\mathbf{y}$. Viswanathan and Grossmann [124] proposed to solve the continuous relaxation of the MINLP in the first iteration, which is obtained by relaxing the integrality

requirement on $\mathbf{y}$,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}), \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \\ & \mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{y} \in \mathbb{R}^{n_y}. \end{aligned} \tag{rMINLP}$$

If rMINLP is infeasible, then MINLP is also infeasible. Otherwise, let $(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$ be a solution to rMINLP. If $\bar{\mathbf{y}}^0$ is integral, $(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$ is an optimal solution to MINLP and the algorithm stops.

If $\bar{\mathbf{y}}^0$ is not integral, a MIP relaxation of MINLP is constructed by linearizing the nonlinear functions in $\mathbf{g}(\mathbf{x},\mathbf{y})$ by first-order Taylor series approximations at $(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$, which, in the case of convex functions, provide supporting hyperplanes [124]. Given a set of solutions $\bar{\mathbf{x}}^k$, $k = 1,\ldots,i-1$, the i-th MIP problem generated by the outer-approximation algorithm is as follows:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & \alpha, \\ \text{s.t.} \quad & f(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k) + \nabla f(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k)^\top \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}}^k \\ \mathbf{y} - \bar{\mathbf{y}}^k \end{bmatrix} \leq \alpha, && k = 0,\ldots,i-1, \\ & g_l(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k) + \nabla g_l(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k)^\top \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}}^k \\ \mathbf{y} - \bar{\mathbf{y}}^k \end{bmatrix} \leq 0, && l \in L^k, k = 0,\ldots,i-1, \\ & \|\mathbf{y} - \bar{\mathbf{y}}^k\|_1 \geq 1, && k \in C^i, \\ & \mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{y} \in \mathbb{Z}^{n_y}, \alpha \in \mathbb{R}, \end{aligned} \tag{MIP^i}$$

where $L^k \subseteq \{1,\ldots,m\}$ is a subset of constraints for which linearizations are included ($L^0 = \{1,\ldots,m\}$, typically), and $C^i \subseteq \{1,\ldots,i-1\}$ is a subset of iterations in which the so-called *no-good cut* $\|\mathbf{y} - \bar{\mathbf{y}}^k\|_1$ is added [212] (discussed below). Note, that due to assumption (A1), the equation $\|\mathbf{y} - \bar{\mathbf{y}}^k\|_1 \geq 1$ can be written in an equivalent linear form, see Appendix 3.B. MIP^i is called the *master problem*. We denote by $(\widehat{\alpha}^i, \widehat{\mathbf{x}}^i, \widehat{\mathbf{y}}^i)$ a solution for MIP^i, if feasible. Due to assumption (A2), the optimal value of MIP^i yields a lower bound to the optimal value

of MINLP, if $C^i = \varnothing$ (for now).

The solution of MIP^i is used to define the following NLP subproblem of MINLP, obtained by fixing the integer variables to $\widehat{\mathbf{y}}^i$:

$$\begin{aligned} \min_{x} \quad & f(\mathbf{x}, \widehat{\mathbf{y}}^i), \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}, \widehat{\mathbf{y}}^i) \leq \mathbf{0}, \\ & \mathbf{x} \in \mathbb{R}^{n_x}. \end{aligned} \tag{NLP^i}$$

Let $\bar{\mathbf{y}}^i := \widehat{\mathbf{y}}^i$ and let $\bar{\mathbf{x}}^i$ be a solution to NLP^i, if feasible. Then $(\bar{\mathbf{x}}^i, \bar{\mathbf{y}}^i)$ is a feasible point to MINLP and provides an upper bound on its optimal value. If NLP^i is not feasible, then let $(\bar{\mathbf{x}}^i, \bar{\mathbf{s}}^i)$ be a minimal infeasible solution to NLP^i, that is, a solution to the NLP

$$\begin{aligned} \min_{\mathbf{x},\mathbf{s}} \quad & \sum_{j=1}^{m} s_j, \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}, \widehat{\mathbf{y}}^i) - \mathbf{s} \leq \mathbf{0}, \\ & \mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{s} \in \mathbb{R}^m_+. \end{aligned} \tag{NLP-feas^i}$$

Note that adding linearization of $g_j(\mathbf{x}, \mathbf{y})$ in $(\bar{\mathbf{x}}^i, \bar{\mathbf{y}}^i)$ for those $j \in \{1, \ldots, m\}$ with $g_j(\bar{\mathbf{x}}^i, \bar{\mathbf{y}}^i) > 0$ to MIP^i will eliminate $(\bar{\mathbf{x}}^i, \bar{\mathbf{y}}^i)$ from its feasible set. However, there may exist some other values of $\mathbf{x}$ for which $(\mathbf{x}, \bar{\mathbf{y}}^i)$ is still feasible for MIP^i. Therefore, one may, additionally or alternatively, add the no-good cut $\|\mathbf{y} - \widehat{\mathbf{y}}^i\|_1 \geq 1$ to MIP^i to cut off any point in $\mathbb{R}^{n_x} \times \{\widehat{\mathbf{y}}^i\}$. Therefore, if only those iterations are included into C^i for which NLP^i is infeasible, then the optimal value of MIP^i provides a lower bound to the optimal value of MINLP.

The outer-approximation algorithm is summarized in Algorithm 1. The NLP and MIP problems are solved alternately until the gap between the bounds given by NLP^i and MIP^i is less than the specified tolerance. It has been proved that this algorithm finds an ϵ-optimal solution of a convex MINLP or proves that none exist in a finite number of iterations [122].

3.2.2 Outer-approximation in DICOPT

Outer-approximation is the main algorithm behind the solver DICOPT [124, 213, 214], which has been developed in the late 1980s by the research group of I.E. Grossmann at the Engineering Research Design Center at Carnegie Mellon University. Since then, it has been available in the commercial algebraic modeling system GAMS. DICOPT solves NLP and MIP problems using other solvers that are available in GAMS and are specialized to these problem types.

As DICOPT is also intended as a heuristic for nonconvex MINLPs, the implementation of the outer-approximation algorithm deviates slightly from Algorithm 1 as described in Appendix 3.A.

3.2.3 Feasibility pump

The feasibility pump algorithm is a primal heuristic developed by Fischetti, Glover, and Lodi to quickly find feasible solutions for MIPs where all integer variables are binaries [209]. Extensions and variations of the algorithm have been proposed, including an extension to general integer variables [215]. Nowadays, many state-of-the-art commercial and non-commercial MIP solvers feature implementations of the feasibility pump [215]. The first extension of the feasibility pump algorithm to convex MINLP problems was introduced by Bonami et al. [210]. Contrary to the original feasibility pump for MIP [209], the convex MINLP feasibility pump is guaranteed to converge to a feasible solution, if any. Subsequently, several authors have proposed extensions to nonconvex MINLPs [216–218], where the handling of the nonconvex nonlinear constraints poses an additional challenge. The MINLP solvers BONMIN and Couenne have implemented feasibility pump algorithms as primal heuristics [210, 216].

The main idea of this algorithm is to decompose the original mixed-integer problem

into two parts: integer feasibility and constraint feasibility. For convex MINLPs, a MIP is solved to obtain a solution, which satisfies the integrality constraints on $\mathbf{y}$, but may violate some of the nonlinear constraints; next, by solving an NLP, a solution is computed that satisfies the constraints ($\mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}$) but might again violate the integrality constraints on $\mathbf{y}$. By minimizing the distance between these two types of solutions iteratively, a solution that is both constraint and integer feasible can be expected. The first iteration of the algorithm proposed in [210] is the same as in Algorithm 1, where the continuous relaxation rMINLP of the original MINLP problem is solved. Following this, the next iteration builds a MIP master problem with the outer-approximation linearization of the nonlinear constraints and a modified objective function called the Feasibility Outer-approximation:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & \|\mathbf{y} - \bar{\mathbf{y}}^{i-1}\|_1, \\ \text{s.t.} \quad & g_l(\bar{\mathbf{x}}^k,\bar{\mathbf{y}}^k) + \nabla g_l(\bar{\mathbf{x}}^k,\bar{\mathbf{y}}^k)^\top \begin{bmatrix} \mathbf{x}-\bar{\mathbf{x}}^k \\ \mathbf{y}-\bar{\mathbf{y}}^k \end{bmatrix} \leq 0, \quad l \in L^k, k = 0,\ldots,i-1, \\ & \mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{y} \in \mathbb{Z}^{n_y}, \end{aligned} \tag{FOA^i}$$

where $L^k \subseteq \{1,\ldots,m\}$ is chosen as in Algorithm 1. The solution to this problem is denoted as $(\widehat{\mathbf{x}}^i,\widehat{\mathbf{y}}^i)$. In FOA^i, the original objective function has been replaced by the ℓ_1-distance of $\mathbf{y}$ to $\bar{\mathbf{y}}^{i-1}$. In the first iteration, $\bar{\mathbf{y}}^0$ corresponds to the solution of the continuous relaxation rMINLP of MINLP. However, in the following iterations, $\bar{\mathbf{y}}^{i-1}$ is given by the solution of the following nonlinear program for the feasibility pump:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & \|\mathbf{y} - \widehat{\mathbf{y}}^{i-1}\|_2^2, \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \\ & \mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{y} \in \mathbb{R}^{n_y}. \end{aligned} \tag{FP-NLP^i}$$

The solution of this problem is denoted as $(\bar{\mathbf{x}}^i,\bar{\mathbf{y}}^i)$. If $\bar{\mathbf{y}}^i \in \mathbb{Z}^{n_y}$, a feasible solution for MINLP has been found.

Bonami et al. [210] have shown on an example that this basic algorithm can cycle ($(\bar{\mathbf{x}}^{i-1}, \bar{\mathbf{y}}^{i-1}) = (\bar{\mathbf{x}}^{i}, \bar{\mathbf{y}}^{i})$) if certain constraint qualifications are not satisfied. A possibility to avoid this cycling is to add the cut

$$(\bar{\mathbf{y}}^{i} - \widehat{\mathbf{y}}^{i-1})^{\top}(\mathbf{y} - \bar{\mathbf{y}}^{i}) \geq 0 \tag{3.1}$$

to FOA^i. Since FP-NLP^i projects the solution $\widehat{\mathbf{y}}^{i-1}$ onto the convex set $\{\mathbf{y} \in \mathbb{R}^{n_y} : \exists \mathbf{x} \in \mathbb{R}^{n_x} : \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$, the cut (3.1) outer-approximates the feasible region of MINLP and is violated by $\widehat{\mathbf{y}}^{i-1}$ (unless $\bar{\mathbf{y}}^{i} = \widehat{\mathbf{y}}^{i-1}$, in which case $(\bar{\mathbf{x}}^{i}, \bar{\mathbf{y}}^{i})$ is a feasible solution for MINLP). Thus, adding it to FOA^i avoids revisiting $\widehat{\mathbf{y}}^{i-1}$. This algorithm is denoted as *enhanced Feasibility Pump* in [210] and has been shown to find a feasible solution to MINLP or prove that none exist in a finite number of iterations, if assumptions (A1) and (A2) are satisfied.

To find further (and better) feasible solutions, the feasibility pump can be applied iteratively, thereby excluding solutions for which the (linearized) objective function has a worse value than the best-known value. This is achieved by the following modification to FOA^i:

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{y}} \quad & \|\mathbf{y} - \bar{\mathbf{y}}^{i-1}\|_1, \\
\text{s.t.} \quad & f(\bar{\mathbf{x}}^{k}, \bar{\mathbf{y}}^{k}) + \nabla f(\bar{\mathbf{x}}^{k}, \bar{\mathbf{y}}^{k})^{\top} \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}}^{k} \\ \mathbf{y} - \bar{\mathbf{y}}^{k} \end{bmatrix} \leq \alpha, \qquad k = 0, \ldots, i-1, \\
& g_l(\bar{\mathbf{x}}^{k}, \bar{\mathbf{y}}^{k}) + \nabla g_l(\bar{\mathbf{x}}^{k}, \bar{\mathbf{y}}^{k})^{\top} \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}}^{k} \\ \mathbf{y} - \bar{\mathbf{y}}^{k} \end{bmatrix} \leq 0, \quad l \in L^{k}, k = 0, \ldots, i-1, \\
& \alpha \leq Z^{U} - \delta, \\
& \mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{y} \in \mathbb{Z}^{n_y}, \alpha \in \mathbb{R}.
\end{aligned} \tag{FP-OA^i}$$

The variable α is initially unbounded ($Z^U = \infty$). When a new incumbent is found, Z^U is updated to the value of the original objective function in the incumbent. The small positive constant δ ensures that the incumbent becomes infeasible in FP-OA^i and enforces

the search for an improving solution. If MINLP is feasible, (A2) and (A3) are satisfied, and linear independence constraint qualificiations hold for FP-NLP^i at $(\bar{\mathbf{x}}^i, \bar{\mathbf{y}}^i)$, then this iterative algorithm finds a δ-optimal solution.

3.3 Proposed Algorithm

While the main focus of the outer-approximation algorithm is to find the best possible solution and proving its optimality, the feasibility pump algorithm mostly disregards the original objective function. It focuses primarily on simultaneously minimizing the violation of integrality and nonlinear constraints. Therefore, the outer-approximation algorithm may take longer to find feasible solutions on problems where feasible solutions are challenging to find. In contrast, the (iterative) feasibility pump algorithm may take longer to find a (proven) optimal solution on problems with many feasible points. To alleviate and explore the differences between these algorithms, hybrid algorithms have been designed, the first being in [210]. In [210], the feasibility pump algorithm is called when the NLP subproblem NLP^i is found to be infeasible.

For DICOPT, we implemented a variation of this hybrid algorithm. Instead of starting the feasibility pump one or several times within the outer-approximation algorithm, we run the iterative feasibility pump once before the main outer-approximation loop starts. Furthermore, we slightly modified the feasibility pump algorithm in the following way.

A drawback of neglecting the original objective function in the feasibility pump algorithm as stated in Section 3.2.3 is that although it may be successful in finding feasible solutions, the quality of solutions in terms of the objective function value can be poor [219]. Therefore, as in [210], after finding a feasible solution by solving FP-NLP^i, we try to improve it further by solving the NLP subproblem obtained from fixing all integer variables in MINLP to the

values in the solution of FP-NLP^i (that is, we solve NLP^i with $\widehat{\mathbf{y}}^i$ replaced by $\bar{\mathbf{y}}^i$).

Another problem arises from the possibility of repeating the same values in the integer variables ($\widehat{\mathbf{y}}^{i-1} = \widehat{\mathbf{y}}^i$), either due to cycling or when having several feasible solutions with the same values in the integer variables. The former is avoided by adding the cut (3.1), if $\bar{\mathbf{y}}^i \neq \widehat{\mathbf{y}}^{i-1}$, as proposed by [210]. If, however, $\bar{\mathbf{y}}^i = \widehat{\mathbf{y}}^{i-1}$, then a feasible solution for MINLP has been found and we can add the no-good cut $\|\mathbf{y} - \widehat{\mathbf{y}}^i\| \geq 1$. With $\delta > 0$, this would not be necessary to ensure progress in the search for an improving solution. However, we believe that it might accelerate the search to add this cut. Since the linearization of no-good cuts may require additional variables if general integer variables are present (see Appendix 3.B), no-good cuts are by default only added for mixed-binary problems. To summarize, the MIP projection problem that we solve is the following:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y}} \quad & \|\mathbf{y} - \bar{\mathbf{y}}^{i-1}\|_1, \\
\text{s.t.} \quad & f(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k) + \nabla f(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k)^\top \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}}^k \\ \mathbf{y} - \bar{\mathbf{y}}^k \end{bmatrix} \leq \alpha, && k = 0, \ldots, i-1, \\
& g_l(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k) + \nabla g_l(\bar{\mathbf{x}}^k, \bar{\mathbf{y}}^k)^\top \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}}^k \\ \mathbf{y} - \bar{\mathbf{y}}^k \end{bmatrix} \leq 0, && l \in L^k, k = 0, \ldots, i-1, \\
& \|\mathbf{y} - \bar{\mathbf{y}}^k\|_1 \geq 1, && k \in C^i, \\
& (\bar{\mathbf{y}}^k - \widehat{\mathbf{y}}^k)^\top (\mathbf{y}^k - \bar{\mathbf{y}}^k) \geq 0, && k = 1, \ldots, i-1, \\
& \alpha \leq Z^U - \delta \max(|Z^U|, 1), \\
& \mathbf{x} \in \mathbb{R}^{n_x}, \mathbf{y} \in \mathbb{Z}^{n_y}, \alpha \in \mathbb{R}.
\end{aligned}
\tag{FP-OA^i}
$$

Finally, similar to FP-OA^i, we add the constraint $f(\mathbf{x}, \mathbf{y}) \leq Z^U - \delta \max(|Z^U|, 1)$ to FP-NLP^i in order to avoid non-improving solutions.

When the feasibility pump terminates, the outer-approximation algorithm is initialized not only by the best solution that the feasibility pump may have found but also with the

linearizations, no-good cuts, and cuts (3.1) that have been added to FP-OA^i. However, regarding the cuts (3.1), only those generated before the last incumbent solution has been found can be used to initialize the outer-approximation algorithm, since later cuts were generated w.r.t. the additional constraint $f(\mathbf{x},\mathbf{y}) \leq Z^U - \delta \max(|Z^U|,1)$, which may cut off an optimal solution $(\mathbf{x}^*,\mathbf{y}^*)$ if $Z^U - \delta \max(|Z^U|,1) < f(\mathbf{x}^*,\mathbf{y}^*) < Z^U$.

A general outline of the proposed algorithm is given in Algorithm 2. Up until the Line 27 of Algorithm 2, the algorithm is similar to the Iterated Feasibility Pump (IFP) for MINLP proposed in [210]. The main differences with the IFP are the optional inclusion of the no-good cuts and the solution of problem NLP^i in Line 19. Therefore, we can argue that executing lines 1–27 of Algorithm 2 with $i_{\max} = \infty$ finds a δ-optimal ($\delta > 0$) solution to MINLP or proves that none exist if assumptions (A2) and (A3) are satisfied (see Theorem 2 in [210]). To find an optimal solution ($\delta = 0$), if any exists, no-good cuts need to be also generated if general integer variables are present. To add these in linear form, Assumption (A1) needs to be satisfied.

Contrary to the Enhanced Outer-approximation method presented by Bonami et al. [210], which runs the feasibility pump both as starting procedure and when the NLP subproblem NLP^i is infeasible; we employ the feasibility pump algorithm only once and before the actual Outer-approximation algorithm. This is motivated by the fact that the MIP FP-OA^i, built by the feasibility pump, provides a valid relaxation for the convex MINLP. Therefore, the feasibility pump does not only provide an initial feasible solution if successful but also, in any case, an initialization of the MIP relaxation MIP^i.

Algorithm 2 has been implemented as part of the solver DICOPT and is available in GAMS since version 25.1 (an earlier version without cuts (3.1) is available since GAMS 24.5). Several options were added to enable and adjust the algorithm, summarized in Table 3.1. As DICOPT is often used for nonconvex MINLPs, see also the discussion in Appendix 3.A,

Table 3.1: Feasibility pump options in DICOPT

Option	Description	Default
`convex`	If enabled, then the default values for the following options are changed to be more appropriate for convex MINLPs, see also Section 3.2.2: option `step` is set to 1, option `infeasder` is set to 1, and option `feaspump` is set to 1	0
`feaspump`	Whether to run the feasibility pump	0
`fp_iterlimit`	Major iteration limit ($i_{\max}$) in the feasibility pump	20
`fp_timelimit`	Time limit in the feasibility pump	∞
`fp_sollimit`	Limit on number of (improving) solutions found by the feasibility pump	∞
`fp_stalllimit`	Limit on the number of consecutive iterations where no improving solution is found. Only applies after a first solution has been found.	5
`fp_cutoffdecr`	Relative decrease of cutoff value for objective variable (δ)	0.1
`fp_acttol`	Tolerance on when a constraint is found active	10^{-6}
`fp_projzerotol`	Tolerance on when to consider the difference $\|\bar{\mathbf{y}}^i - \widehat{\mathbf{y}}^i\|$ as zero	10^{-4}
`fp_mipgap`	Optimality tolerance (relative gap) when solving FP-OA^i	0.01
`fp_transfercuts`	Whether to transfer cuts from the feasibility pump MIP to the DICOPT MIP	1
`fp_integercuts`	Whether to add no-good cuts to FP-OA^i when finding a new feasible solution	1
`fp_projcuts`	Whether to add cuts (3.1) to FP-OA^i after solving FP-NLP^i	1

the default values for options `convex` and `feaspump` are 0.

3.4 Computational results

In the following, we evaluate the benefits of adding the feasibility pump to DICOPT on a set of convex MINLPs selected from MINLPLib (version `c0f77612`, as of 13.3.2018)* [220].

*http://www.minlplib.org

First, we selected all instances marked as convex, not proven to be infeasible, and having at least one binary or general integer variable, no semicontinuous or semi-integer variables, and no special-ordered-sets. This gives a set of 359 instances. Second, we run DICOPT with the `convex` option enabled, and the feasibility pump disabled, and removed all instances for which DICOPT terminated in less than one second. In this remaining set, a strong dominance of some subsets of instances derived from the same model (substantial similarity in name) was observed. Therefore, we reduced these subsets to the four largest instances. This leaves a final set of 80 instances, which have their origin in a wide variety of applications, ranging from process synthesis flowsheets, facilities layout problems, batch design with storage, water treatment models, and investment portfolios. The Supplemental Material from the original paper* provides this list of instances.

For all the experiments, we used a time limit of 1800 seconds and set the GAMS gap tolerance `optcr` (relative distance of Z^L and Z^U) to 10^{-5}. GAMS 25.1.1 was run on a cluster of Dell PowerEdge M620 blades with 64 GB RAM, Intel Xeon E5-2680 CPUs running at 2.70 GHz, and Linux 4.4.0 (64bit). With this GAMS version, DICOPT uses CPLEX 12.8.0.0 for solving MIPs, and CONOPT 3.17I for solving NLPs. We used PAVER 2 [221] to help in the evaluation.

3.4.1 Illustrative example

Before evaluating the performance of the new feasibility pump on the complete test set, we discuss its behavior for a single instance. This instance corresponds to the block layout design problem with unequal areas. The original problem was proposed by Meller et al. [222] and was reformulated by Castillo et al. [223] as a convex MINLP. This type of problem may be applied in piping design problems and process plant layouts. The complete

*https://ndownloader.figstatic.com/files/17135045

formulation of this model is reported in [223]. The test case selected was the block layout design problem of 7 departments and an aspect ratio (the maximum permissible ratio between its longest and shortest dimensions) of 5. The problem involves 211 constraints, 14 of them nonlinear, specifically signomial, and 114 variables, 42 of them binary. This instance can be found in MINLPLib under the name `o7_2`*. The authors of the model used several MINLP solvers to find the optimal solution to this problem, among them DICOPT. DICOPT performed very poorly because of the linearizations in the initial outer-approximation MIP^i were not helpful, and many nonlinear subproblems NLP^i are infeasible [223].

The given instance was tested using different options for DICOPT. The stopping criterion for all the different options was closing the gap between the objective values of the MIP master problem and the incumbent solution. The default setting for option `infeasder` requires that if the nonlinear subproblem NLP^i is infeasible, only a corresponding no-good cut is added to MIP^i. Although rigorous for convex and non-convex MINLPs, this approach is not very efficient, particularly for this type of problem where "a significant amount of no-good cuts may be required before a feasible solution is obtained" [223]. For convex MINLPs, another rigorous approach is to add linearization cuts if the nonlinear subproblem is infeasible, using the solution of NLP-feas^i as a reference point. This can be enabled by using the option `infeasder`. Note that the setting of the `infeasder` option does not influence the handling of infeasible NLPs within the feasibility pump. A comparison of DICOPT on instance `o7_2` with the feasibility pump and the `infeasder` option enabled and disabled is given in Table 3.2.

We notice that DICOPT without feasibility pump and with `infeasder` disabled cannot find a feasible solution within 30 minutes. During this time, the solver performed 113 major iterations. That is, at each iteration, it solved a MIP master problem MIP^i and an NLP

*http://www.minlplib.org/o7_2.html

Table 3.2: Results of the solution of the illustrative example o7_2 for each setting of DICOPT.

DICOPT options	w/o FP w/o infeasder	w/o FP w/ infeasder	w/ FP w/o infeasder	w/ FP w/ infeasder
major iterations	113	7	2	2
feasible solutions found	0	1	5	5
FP iterations	0	0	12	12
FP time [s]	0	0	199.4	200.1
infeasible NLP	112	5	0	0
time to optimal sol. [s]	–	676.6	417.3	418.4
solution time [s]	> 1800*	915.5	620.9	621.2
final objective value	–	116.95	116.95	116.95

*Time limit reached.

subproblem NLP^i. All NLP subproblems were infeasible.

Enabling the `infeasder` option, the problem could be solved in 912 seconds. During this time, 5 out of the 6 solved NLP subproblems were infeasible. The only feasible solution found in the 7th iteration was also an optimal solution to the problem. It required another solution of the MIP to prove its optimality. The feasibility pump allowed the solver to find four feasible solutions in the first $\approx$ 200 seconds. After that, a single major iteration was required to find an optimal solution, which required 217 seconds in both cases with and without the `infeasder` option. In that same major iteration, the optimality of the solution was proven. The results when using the feasibility pump with and without the `infeasder` option are the same (except for variations in time measurement) since none of the NLP subproblems NLP^i in the outer-approximation algorithm was infeasible.

These results highlight that enabling the `infeasder` option can be essential to solving a problem or finding a feasible solution. Second, the feasibility pump can further improve the performance by finding feasible solutions early. We obtained a 38% reduction in the time needed to find an optimal solution to the problem and a 32% reduction in the full solution time by enabling the feasibility pump. It is also interesting to note that when this problem

is solved with AlphaECP, it required 897 seconds, with BONMIN 756 seconds, and SCIP 768 seconds.

3.4.2 Feasibility pump alone

In the following, we consider the complete test set of 80 instances. First, we run only our (iterative) feasibility pump implementation with various settings, that is, without continuing with the outer-approximation algorithm of DICOPT. In setting "default", the feasibility pump is run in its default settings, see Table 3.1, that is, a stall limit of 5 and a cutoff decrease of $\delta = 0.1$, except that the iteration limit has been disabled ($i_{\max} = \infty$). A stall limit of k iterations stops the feasibility pump if, after a first solution has been found, no improving solution is found within the subsequent k iterations. In setting "stall10", we increased the stall limit to 10. In setting "no cuts (3.1)", we disabled the addition of cuts (3.1) to FP-OA^i after having solved the NLP projection problem FP-NLP^i. Setting "no no-good cuts" completely disables the addition of no-good cuts when a new feasible solution has been found ($C^{i+1} = \varnothing$ in Line 22 of Algorithm 2). Finally, we evaluated three settings that target to find δ-optimal solutions of the MINLP. For this, setting "findopt" disables the stall limit and sets the cutoff decrease δ to 10^{-5}. Setting "findopt w/o no-good cuts" also disables the addition of no-good cuts when a new solution is found. Setting "findopt w/ all no-good cuts" enables the addition of no-good cuts to problems with general integer variables.

Table 3.3 summarizes the results for all settings. The mean time in Table 3.3 reports the shifted geometric mean of the runtimes $(t_1, \ldots, t_{80})$ of the feasibility pump on all instances, computed as $\prod_{i=1}^{80}(t_i + 1)^{\frac{1}{80}} - 1$. Figure 3.1 plots the primal gap of all runs for settings "default", "stall10", and "findopt". As primal gap, we compute the relative distance between the objective function value of the best solution found by the algorithm and the objective function value of the best known solution reported in MINLPLib. We can observe that the

Table 3.3: Results of running feasibility pump alone with different settings. For each setting, we show the number of instances in which the feasibility pump reaches the time limit, found a δ-optimal solution (without necessarily proving optimality), found a solution with primal gap $\leq 10\%$, found any feasible solution, and the time used, respectively.

setting	timeout	optimal	good sol.	feasible	mean time [s]
default	4	12	57	77	9.1
stall10	4	14	64	77	11.4
no cuts (3.1)	5	13	55	76	9.6
no no-good cuts	4	13	60	77	9.8
findopt	25	67	74	77	119.0
findopt w/o no-good cuts	42	44	52	77	238.1
findopt w/ all no-good cuts	29	69	74	77	197.1

feasibility pump in default settings finds an optimal solution for 12 instances, reasonable solutions ($< 10\%$ primal gap) for another 45 instances, and some feasible solutions ($\geq 10\%$ primal gap) for another 20 instances. Increasing the stall limit helps on seven of the instances where previously only bad solutions were found. On instances where good solutions were already found in default settings, increasing the stall limit affects one instance only, likely because the cutoff decrease δ cuts off solutions that are only slightly better or optimal. However, increasing the stall limit also leads to an $\approx 25\%$ increase in mean running time. By using the "findopt" setting, however, the feasibility pump can find optimal solutions for many instances where previously a small gap was remaining.

For the runs with stall limit ("default" and "stall10"), the feasibility pump usually terminates either when the MIP approximation FP-OA^i becomes infeasible, or the stall limit is reached. However, in the "findopt" setting, 25 instances terminated when the time limit of 1800 seconds was reached. Thus, the feasibility pump is not suited to prove the optimality of the found solutions. This justifies the choice of the stall limit as a stopping criterion.

Disabling cuts (3.1) has a slight negative impact on performance. Without this cut, the feasibility pump fails to find a solution for instance `tls7` within the allowed time, which

then also leads to an increase in the mean time. As noted by [210], cut (3.1) was not necessary in practice, though adding it is unlikely to have a negative effect. Also, disabling the no-good cuts has little impact on the performance. The number of instances with optimal and good solutions increases slightly, but the mean running time also increases slightly. However, disabling no-good cuts in the "findopt" setting has a severe impact on the performance, since, without these cuts, the cutoff decrease δ, which is only 10^{-5} in this setting, is the sole responsible for forcing the feasibility pump to look for better feasible solutions. On instances with general integer variables, no-good cuts are already disabled by default, which is why there is one instance (second instance in Figure 3.1) where the "findopt" setting produces a worse solution than "default". Hence, enabling no-good cuts also for instances with general integer variables improves solution quality at the cost of a considerably increased running time.

3.4.3 DICOPT with feasibility pump

We used DICOPT with the following settings: In the "DICOPT w/ FP" setting, the option `convex` was enabled, which also enables the feasibility pump. In the "DICOPT w/o FP" setting, the option `convex` was also enabled, but the feasibility pump was disabled. In the "DICOPT w/ FP w/o OA init" setting, option `convex` was again enabled, but the transfer of cuts from the feasibility pump MIP FP-OA^i to the outer-approximation MIP MIP^i has been disabled. Additionally, with "FP only" we consider the results from running only the feasibility pump without stall limit, and cutoff decreases $\delta = 10^{-5}$ ("findopt" setting in Section 3.4.2).

Table 3.4 summarizes for each setting the number of instances for which the time limit was reached, a δ-optimal solution was found, optimality was proven, a good solution was found (primal gap $\leq 10\%$), and mean running time. Detailed results are given in Tables B.3

Table 3.4: Results of running DICOPT with different settings.

setting	timeout	optimal	optimal w/ proof	good sol.	mean time
DICOPT w/ FP	31	58	48	67	137.9
DICOPT w/o FP	31	55	48	62	140.8
FP only	25	67	30	74	119.0
DICOPT w/ FP w/o OA init	31	59	49	68	170.5

and B.4 in the Supplemental Material from the paper*, and performance profiles are shown in Figures 3.2 and 3.3. The numbers show that adding the feasibility pump to DICOPT leads to finding optimal solutions to three more instances than before and slightly reducing the mean running time. Running the feasibility pump alone increases the number of instances where optimal or good solutions are found. It even decreases the mean running time but decreases the number of instances where optimality is proven. The best performance in finding proven optimal solutions can only be expected when combining the feasibility pump as primal heuristic and Outer-approximation to prove optimality. We also note that the outer-approximation algorithm in DICOPT is targeted for general MINLPs, which results in applying linearizations of nonlinear functions and convex, as soft constraints only cf. Section 3.A. That is, tuning the implementation of the outer-approximation in DICOPT to work better in the case of a convex MINLP might lead to achieving the best performance of DICOPT (with feasibility pump) also concerning running time or finding good solutions.

One of the original motivations to add the feasibility pump to DICOPT was to use cuts from the MIP projection problem FP-OA^i to warm-start the outer-approximation MIP MIP^i. As seen from Table 3.4, even though the solution quality does not decrease when disabling the initialization of MIP^i, one can observe that the additional time spent for running the feasibility pump pays off only if the cuts from FP-OA^i are transferred to MIP^i.

*https://ndownloader.figstatic.com/files/17135045

3.5 Conclusions and perspectives

This paper has addressed the solution of convex MINLPs using the commercial solver DICOPT. A modified iterative feasibility pump algorithm as a preprocessing for DICOPT has been proposed and implemented. As seen in the illustrative example, DICOPT in default settings performs poorly when many nonlinear subproblems are infeasible. Solving the illustrative example using DICOPT with the feasibility pump, better performance in solution time and solution quality could be achieved. As seen in the results from Section 3.4.2, the feasibility pump is not efficient in proving optimality, which validates the use of a stall limit as the criterion when to switch from the feasibility pump to the outer-approximation algorithm.

Suppose the feasibility pump is used as a primal heuristic only. In that case, the quality of the found solutions is improved, but the running time of DICOPT is increased considerably. The performance of DICOPT without the feasibility pump can only be improved with the cuts from the feasibility pump's MIP projection problem to initialize the MIP of the outer-approximation algorithm.

Further work to improve the feasibility pump implementation in DICOPT is motivated by the following observations. Achterberg and Bethold [219] proposed a modification to the original algorithm that includes some information about the original objective function in the objective function of the feasibility pump problems to mitigate the issue of finding poor feasible solutions in terms of the original objective. Further, currently, the feasibility pump is only run at the beginning of DICOPT before the main loop of the outer-approximation algorithm. Executing it only once has been sufficient to find good feasible solutions for many instances in our test set. However, for some instances, it may be worth investigating a more extensive integration of the feasibility pump into DICOPT, *e.g.*, allowing it to be also used when infeasible NLP subproblems are encountered similar to the approach proposed

by Bonami et al. [129]. Finally, the feasibility pump implementation should be generalized to nonconvex MINLP problems. Several authors have proposed such extensions [216, 217]. DICOPT itself already has heuristics to deal with nonconvex MINLPs, see Section 3.2.2, which could be carried over to the feasibility pump implementation.

3.A Implementation details of DICOPT

DICOPT is based on the Outer-approximation algorithm for solving MINLP problems. Although the Outer-approximation algorithm has a guaranteed convergence for convex MINLP problems [122], DICOPT implements the methods of equality relaxation and augmented penalty to make it a heuristic method for solving non-convex MINLP problems. The main differences between the original Outer-approximation and the implementation in DICOPT are:

- For convex MINLP, Z^L and Z^U yield valid lower and upper bounds on the optimal value of MINLP given that NLP^i is typically solved to global optimality. Therefore, closing the gap between these bounds is a stopping criterion that ensures finding a globally optimal solution in a finite number of iterations. This can be enabled in DICOPT by setting the option `stop` to 1. For nonconvex MINLPs, valid lower bounds and solving NLP^i to global optimality are not ensured. Therefore, by default, DICOPT stops as soon as the upper bound Z^U stops improving. Although it is a heuristic, this stopping criterion has shown that it yields optimal or near-optimal integer solutions in many cases.
- If the NLP subproblem NLP^i is infeasible, DICOPT by default adds only an no-good cut to eliminate the current fixing $\mathbf{y} = \widehat{\mathbf{y}}^i$ from MIP^i, but does not add the corresponding linearizations of nonlinear functions, *i.e.*, $L^i = \varnothing$ if NLP^i is infeasible in Line 29 of

Algorithm 1. This option is sufficient to avoid revisiting the same solution point while avoiding adding linearization that are not supporting hyperplanes for nonconvex MINLPs. However, it also yields slower progress as less information is made available to the master problem. Thus, when solving a convex MINLP, these valid linearizations should be added. This can be enabled in DICOPT by setting the option `infeasder`.

- If the NLP subproblem NLP^i is feasible linearizations of nonlinear functions are not added in their original form to MIP^i. Instead, they are added as soft constraints; that is, violation of these constraints is allowed but penalized in the objective function (by default, a weight of 1000 times the constraint marginal is used) [124]. Also, in the context of convex MINLP, the penalty relaxation of linearizations is applied. Note that the optimal value of the modified master problem still provides a valid lower bound on the optimal value of MINLP. If the penalty term's contribution is removed and termination is still ensured due to the finite number of integer points $\mathbf{y}$ to be enumerated.
- Finally, DICOPT relaxes nonlinear equality constraints to inequalities and adds corresponding linearizations to MIP^i. The dual multipliers in the solution of NLP^i are used to decide which direction to relax the inequalities [124]. For a convex MINLP, such constraints do not appear.

3.B Linearization of no-good cut

For given bounds $\mathbf{y}^L, \mathbf{y}^U \in \mathbb{Z}^{n_y}$, $\mathbf{y}^L \leq \mathbf{y}^U$, on the integer variables $\mathbf{y}$ and a point $\bar{\mathbf{y}} \in \mathbb{Z}^{n_y}$, $\mathbf{y}^L \leq \bar{\mathbf{y}} \leq \mathbf{y}^U$, consider the no-good cut

$$\|\mathbf{y} - \bar{\mathbf{y}}\|_1 \geq 1. \tag{3.2}$$

A linear formulation of (3.2) is easily found if $\bar{y}_j \in \{y_j^L, y_j^U\}$ for every $j \in J := \{1, \ldots, n_y\}$, since the absolute difference $|y_j - \bar{y}_j|$ is reduced to $y_j - y_j^L$, if $\bar{y}_j = y_j^L$, and $y_j^U - y_j$ otherwise. Thus, for the specific case of binary variables only, *i.e.*, $y_j^L = 0, y_j^U = 1, j \in J$, (3.2) simplifies to

$$\sum_{j \in J^L} y_j - \sum_{j \in J^U} (1 - y_j) \geq 1.$$

In the general case, we partition the set J into

$$J^L = \{j \in J \ : \ \bar{y}_j = y_j^L\},$$
$$J^U = \{j \in J \ : \ \bar{y}_j = y_j^U\},$$
$$J^M = J \setminus (J^L \cup J^U).$$

Using this set partition, the absolute difference of the variables to a given solution can be expressed as a sum of three terms. Thus, the no-good cut (3.2) can be written as

$$\sum_{j \in J^L} (y_j - y_j^L) + \sum_{j \in J^U} (y_j^U - y_j) + \sum_{j \in J^M} |y_j - \bar{y}_j| \geq 1.$$

For every $j \in J^M$, we introduce a binary variable v_j, which determines whether the variable y_j is greater than or less than $\bar{y}_j$, and a positive continuous variable w_j to represent the value $|y_j - \bar{y}_j|$. This can be expressed using the following disjunctions:

$$\begin{bmatrix} v_j = 0 \\ y_j \leq \bar{y}_j \\ w_j = \bar{y}_j - y_j \end{bmatrix} \vee \begin{bmatrix} v_j = 1 \\ y_j \geq \bar{y}_j \\ w_j = y_j - \bar{y}_j \end{bmatrix}, \quad j \in J^M.$$

This disjunction can be reformulated into mixed-integer linear form, which yields the

following reformulation of (3.2):

$$\begin{aligned}
&\sum_{j\in J^L}(y_j - y_j^L) + \sum_{j\in J^U}(y_j^U - y_j) + \sum_{j\in J^M} w_j \geq 1, && \\
&-w_j \leq y_j - \bar{y}_j \leq w_j, && j \in J^M, \\
&w_j \leq y_j - \bar{y}_j + M_j^1(1 - v_j), && j \in J^M, \\
&w_j \leq \bar{y}_j - y_j + M_j^2 v_j, && j \in J^M, \\
&w_j \geq 0, && j \in J^M, \\
&v_j \in \{0,1\}, && j \in J^M.
\end{aligned}$$

To avoid weak relaxations, the big-M constants M_j^1 and M_j^2 should be chosen as small as possible and such that

$$\begin{aligned}
&y_j - \bar{y}_j + M_j^1 \geq w_j = \bar{y}_j - y_j \quad \forall y_j \in [y_j^L, \bar{y}_j] && \text{(case } v_j = 0 \rightarrow y_j \leq \bar{y}_j\text{)}, \\
&\bar{y}_j - y_j + M_j^2 \geq w_j = y_j - \bar{y}_j \quad \forall y_j \in [\bar{y}_j, y_j^U] && \text{(case } v_j = 1 \rightarrow y_j \geq \bar{y}_j\text{)}.
\end{aligned}$$

Thus, $M_j^1 = 2(\bar{y}_j - y_j^L)$ and $M_j^2 = 2(y_j^U - \bar{y}_j)$.

Given the addition of extra variables for general no-good cuts, these are added by default if the original problem has only integer variables between 0 and 1 (binary). The DICOPT users can enforce the application of the no-good cuts even if the integer variables are not binary using the option `fp_integercuts`.

3.C Performance Profiles

Figure 3.2 shows performance profiles [204] comparing DICOPT with and without feasibility pump and the feasibility pump alone. Figure 3.3 shows a performance profile that illustrates the effect of disabling the initialization of MIP^i with the cuts from FP-OA^i.

Algorithm 1 Outer-approximation algorithm.

Set $Z^U = \infty$, $Z^L = -\infty$, $i = 0$ ▷ Initialization
Define gap tolerance $\epsilon \geq 0$
Solve rMINLP ▷ Solve initial relaxation
if rMINLP is infeasible **then**
    Set $Z^L = \infty$ ▷ MINLP is infeasible
else
    Let $(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$ be an optimal solution of rMINLP
    Set $Z^L = f(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$
    Set $L^0 = \{1, \ldots, m\}$, $C^0 = \emptyset$
    if $\bar{\mathbf{y}}^0 \in \mathbb{Z}^{n_y}$ **then**
        Set $Z^U = f(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$ and $\widehat{\mathbf{y}}^0 = \bar{\mathbf{y}}^0$
while $Z^U - Z^L > \epsilon$ **do**
    Set $i = i + 1$
    Solve MIP^i ▷ Solve master problem
    if MIP^i is infeasible **then**
        Set $Z^L = \infty$ ▷ MINLP is infeasible
    else
        Let $(\widehat{\alpha}^i, \widehat{\mathbf{x}}^i, \widehat{\mathbf{y}}^i)$ be an optimal solution of MIP^i
        Set $Z^L = \widehat{\alpha}^i$
        Set $\bar{\mathbf{y}}^i = \widehat{\mathbf{y}}^i$ and solve NLP^i ▷ Solve nonlinear subproblem
        if NLP^i is infeasible **then**
            Solve NLP-feas^i
            Let $(\bar{\mathbf{x}}^i, \bar{\mathbf{s}}^i)$ be an optimal solution of NLP-feas^i
            Set $C^{i+1} = C^i \cup \{i\}$
        else
            Let $\bar{\mathbf{x}}^i$ be an optimal solution of NLP^i
            Set $C^{i+1} = C^i$
            Set $Z^U = \min(Z^U, f(\bar{\mathbf{x}}^i, \widehat{\mathbf{y}}^i))$
        Set $L^i = \{j \in \{1, \ldots, m\} : g_j(\bar{\mathbf{x}}^i, \widehat{\mathbf{y}}^i) \geq 0 \text{ and } g_j \text{ is nonlinear}\}$
$(\bar{\mathbf{x}}^i, \widehat{\mathbf{y}}^i)$ is an optimal solution of MINLP, if $Z^U < \infty$, otherwise MINLP is infeasible

Algorithm 2 Proposed algorithm.

Set $Z^U = \infty$, $i = 0$ ▷ Initialization
Define cutoff decrease $\delta \geq 0$
Solve rMINLP ▷ Solve initial relaxation
if rMINLP is infeasible **then**
    Stop ▷ MINLP is infeasible
Let $(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$ be an optimal solution of rMINLP
Set $L^0 = \{1, \ldots, m\}$, $C^0 = \emptyset$
Set $Z^L = f(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$
if $\bar{\mathbf{y}}^0 \in \mathbb{Z}^{n_y}$ **then**
    Set $Z^U = f(\bar{\mathbf{x}}^0, \bar{\mathbf{y}}^0)$ ▷ Optimal solution found
    Stop
Set $i = 1$
Solve FP-OA^i ▷ Solve feasibility OA problem
while FP-OA^i is feasible and $i \leq i_{\max}$ **do**
    Let $(\widehat{\mathbf{x}}^i, \widehat{\mathbf{y}}^i)$ be an optimal solution of FP-OA^i
    Solve FP-NLP^i ▷ Solve nonlinear feasibility problem
    Let $(\bar{\mathbf{x}}^i, \bar{\mathbf{y}}^i)$ be an optimal solution of FP-NLP^i
    if $\|\bar{\mathbf{y}}^i - \widehat{\mathbf{y}}^i\| = 0$ **then**
        Solve NLP^i ▷ Solve nonlinear subproblem
        Let $\bar{\mathbf{x}}^i$ be an optimal solution of NLP^i
        Set $Z^U = \min(Z^U, f(\bar{\mathbf{x}}^i, \bar{\mathbf{y}}^i))$ ▷ New incumbent solution
        Set $C^{i+1} = C^i \cup \{i\}$ (if $\mathbf{y} \in \{0,1\}^{n_y}$ in MINLP)
    else
        Set $C^{i+1} = C^i$
    Set $L^i = \{j \in \{1, \ldots, m\} : g_j(\bar{\mathbf{x}}^i, \bar{\mathbf{y}}^i) \geq 0$ and g_j is nonlinear$\}$
    Set $i = i + 1$
    Solve FP-OA^i ▷ Solve feasibility OA problem
Solve MINLP using Alg. 1, initialized with incumbent solution $(\bar{\mathbf{x}}^i, \bar{\mathbf{y}}^i)$, if $Z^U < \infty$, and linearizations given by L^i, no-good cuts given by C^i, and cuts (3.1) in the relaxation MIP^i.

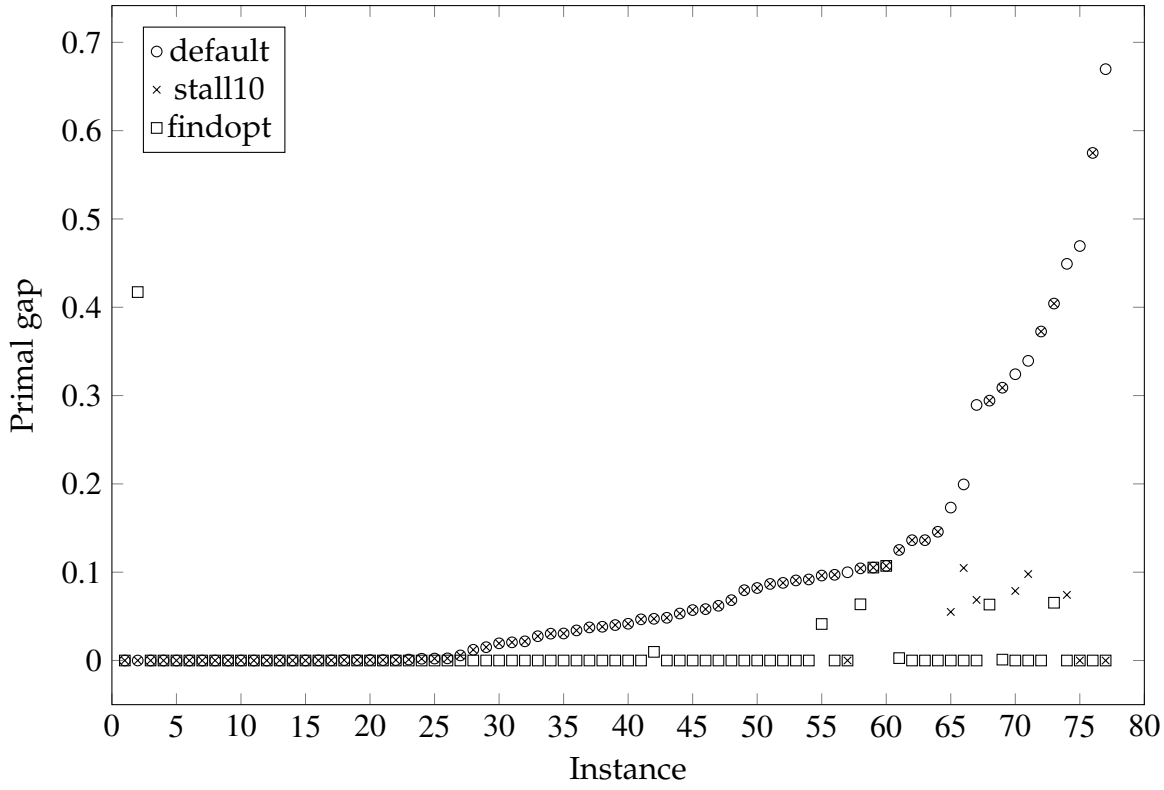


Figure 3.1: Primal gap of solutions found by feasibility pump (with different settings) for all instances in test set, sorted by primal gap of "default" setting.

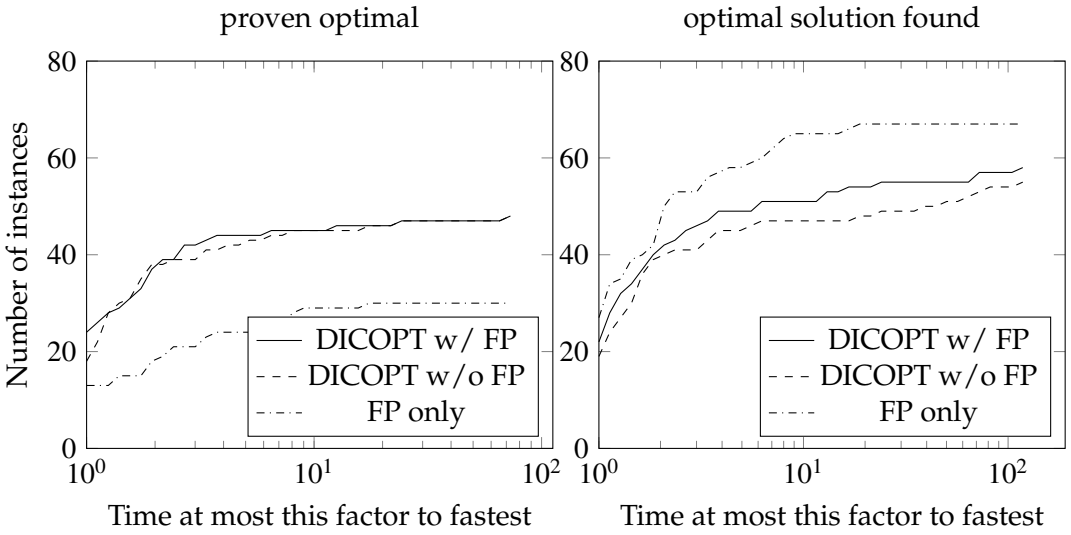


Figure 3.2: Performance profile showing the number of instances solved to proven optimality (left) and where an optimal solution has been found (right), respectively, with respect to solution time for various DICOPT settings.

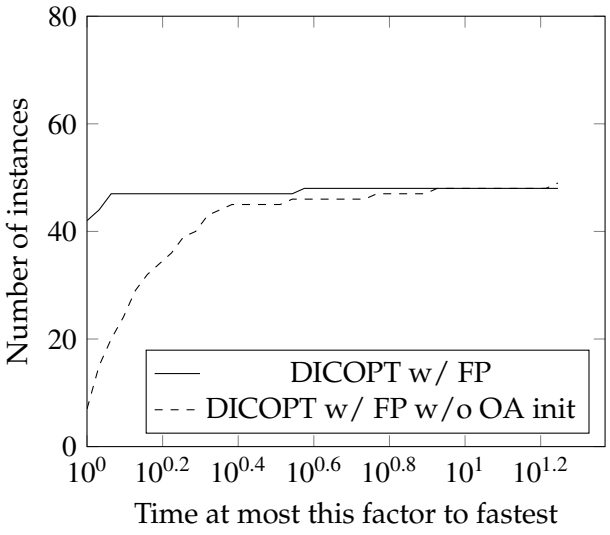


Figure 3.3: Performance profile showing the number of instances solved to proven optimality with respect to solving time, with and without the initialization of MIP^i with the cuts from the feasibility pump.

Chapter 4

Center-cut algorithm for convex MINLP*

4.1 Introduction

In this chapter, we present a Center-cut algorithm for convex Mixed-Integer Nonlinear Programming (MINLP) that can either be used as a primal heuristic or as a deterministic solution technique. Like several other algorithms for convex MINLP, the Center-cut algorithm constructs a linear approximation of the original problem. The main idea of the algorithm is to use the linear approximation differently to find feasible solutions within only a few iterations. The algorithm chooses trial solutions as the center of the current linear outer approximation of the nonlinear constraints, making the trial solutions more likely to satisfy the constraints. The ability to find feasible solutions within only a few iterations makes the algorithm well suited as a primal heuristic, and we prove that the algorithm finds the optimal solution within a finite number of iterations. Numerical results show that the algorithm obtains feasible solutions quickly and can obtain good solutions.

There has been a growing interest in so-called primal heuristics in recent years, *i.e.*, algorithms intended to obtain good feasible solutions to an optimization problem quickly. Such algorithms are helpful not only since they can provide a good feasible solution, but knowing a feasible solution can also significantly improve the performance of solvers, *e.g.*,

*Published as: Jan Kronqvist, David E Bernal, Andreas Lundell, and Tapio Westerlund. "A center-cut algorithm for quickly obtaining feasible solutions and solving convex MINLP problems". *Computers & Chemical Engineering* 122 (2019), pp. 105–113.

[141] claimed that primal heuristics were one of the most crucial improvements for MILP in the last decade. Several primal heuristics have also been proposed for MINLP problems, *e.g.*, undercover [142] and feasibility pump [143]. A review of several primal heuristics for MINLP is given by [146].

Primal heuristics can be a valuable tool, especially for complex MINLP problems, since it may be the only way to obtain a good solution, and for some applications such as real-time optimization, it may be of utter importance to quickly obtain a feasible solution. Knowing a feasible solution can also improve the performance of MINLP solvers as shown in [146] and [148]. A good feasible solution can significantly reduce the search tree in branch and bound, and in solvers based on ECP or ESH, it provides a bound on the objective enabling the use of optimality gap as a stopping criterion. It is also possible to solve pseudoconvex MINLP problems as a sequence of feasibility problems as in the GAECP algorithm [120].

This chapter describes a new Center-cut algorithm for convex MINLP problems that can either be used as a primal heuristic or as a deterministic solution technique. The algorithm was first presented briefly in a conference paper [224]. Here we continue with more details and rigorous proof that the algorithm will obtain the optimal solution in a finite number of iterations. Like OA, ECP, or ESH, the Center-cut algorithm also constructs a polyhedral approximation of the feasible region defined by the nonlinear constraints. However, here we use the polyhedral approximation differently to enable us to find feasible solutions within only a few iterations. The main idea of the algorithm is to choose the trial solutions in the polyhedral approximation center instead of choosing them on the boundary, as with ECP and ESH. A similar concept for solving NLP problems was proposed by [225]. The algorithm should be well suited as a primal heuristic. However, it can also be used as a stand-alone solution technique with guaranteed convergence.

4.2 Background

A convex MINLP problem can be defined compactly as

$$\min_{\mathbf{x},\mathbf{y}\in\mathcal{N}\cap\mathcal{L}\cap\mathcal{Y}} \mathbf{c}_1^\top\mathbf{x}+\mathbf{c}_2^\top\mathbf{y} \tag{P-MINLP}$$

where sets $\mathcal{N},\mathcal{L}$ and $\mathcal{Y}$ are given by

$$\begin{aligned}
\mathcal{N} &= \{(\mathbf{x},\mathbf{y})\in\mathbb{R}^n\times\mathbb{R}^m \mid g_k(\mathbf{x},\mathbf{y})\le 0 \quad \forall k=1,\dots,l\},\\
\mathcal{L} &= \{(\mathbf{x},\mathbf{y})\in\mathbb{R}^n\times\mathbb{R}^m \mid \mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{y}\le\mathbf{b}\},\\
\mathcal{Y} &= \{\mathbf{y}\in\mathbb{Z}^m\}.
\end{aligned} \tag{4.1}$$

In equation (4.1) A and B are matrices defining the linear constraints, including variable bounds. Throughout this chapter we consider the following assumptions to be true:

Assumption 1. The nonlinear functions $g_1,\dots,g_l$ are convex and continuously differentiable.

Assumption 2. The intersection $\mathcal{L}\cap\mathcal{Y}$ defines a compact non-empty set, *i.e.*, all variables must be bounded.

Assumption 3. By fixing the integer variables in the MINLP problem to a feasible integer combination $\mathbf{y}$, the resulting NLP problem satisfies Slater's condition, see [226].

These assumptions are needed to rigorously prove that the algorithm converges to the optimal solution in a finite number of iterations. Similar conditions are also needed to guarantee convergence of OA and ESH, see [25] and [121]. It should be possible to handle MINLP problems with nonsmooth convex functions with the algorithm. However, such problems are not considered here.

One of the critical elements in ECP, ESH, and OA is to construct an iteratively improving polyhedral approximation of set $\mathcal{N}$. The approximation is obtained by first-order Taylor series expansions of the nonlinear constraints generated at the trial solutions; at iteration i it

is given by

$$\widehat{\mathcal{N}}_i = \left\{ g_k(\mathbf{x}^j, \mathbf{y}^j) + \nabla g_k(\mathbf{x}^j, \mathbf{y}^j)^\top \begin{bmatrix} \mathbf{x} - \mathbf{x}^j \\ \mathbf{y} - \mathbf{y}^j \end{bmatrix} \leq 0, j = 1, \ldots, i, \ k \in K_j \right\}, \tag{4.2}$$

where K_j contains the indices of all nonlinear constraints active at the trial solution $(\mathbf{x}^j, \mathbf{y}^j)$, *i.e.,* all nonlinear constraints such that $g_k(\mathbf{x}^j, \mathbf{y}^j) \geq 0$. Due to convexity, we know that the polyhedral approximation will contain set $\mathcal{N}$ and every point within $\mathcal{N}$ is also a point within $\widehat{\mathcal{N}}_i$, *i.e.,* $\mathcal{N} \subset \widehat{\mathcal{N}}_i$.

The standard approach for using the polyhedral approximation is to simply replace set $\mathcal{N}$ by $\widehat{\mathcal{N}}_i$ in problem (P-MINLP). The next trial solution can then be obtained by solving the following MILP problem

$$\left(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}\right) \in \underset{(\mathbf{x},\mathbf{y}) \in \widehat{\mathcal{N}}_i \cap \mathcal{L} \cap \mathcal{Y}}{\arg\min} \ \mathbf{c}_1^\top \mathbf{x} + \mathbf{c}_2^\top \mathbf{y}. \tag{4.3}$$

Both ECP and ESH choose the trial solutions by solving problem (4.3), and OA selects the integer combination by the same approach. However, if we choose the trial solutions by solving problem (4.3), then we will not obtain a feasible solution before the very last iteration, see *e.g.,* [121]. By this approach, the trial solutions tend to be selected as points on the boundary of set $\widehat{\mathcal{N}}_i$.

In the Center-cut algorithm, we will use the polyhedral approximation differently; instead of choosing points on the boundary, we will select the trial solutions as points in the center of the polyhedral approximation. Since we know that the feasible set defined by the nonlinear constraints is contained somewhere in $\widehat{\mathcal{N}}_i$, it seems natural to search for a feasible solution in the center of the set.

4.3 The Center-cut algorithm

As previously mentioned, the main idea of the center cut algorithm is to choose the trial solutions as the center of the polyhedral approximation of the feasible set defined by the nonlinear constraints. There are several definitions of the center of a set, and here we will use the Chebyshev center. The Chebyshev center is defined as the point furthest from the boundary in all directions, which is also the center of the largest n-dimensional ball inscribed in the set [227]. Since set $\widehat{\mathcal{N}}_i$ is a polyhedral set defined by linear inequality constraints, we can find the Chebyshev center of the set simply by solving the following Linear Programming (LP) problem.

$$
\begin{aligned}
&\max_{\mathbf{x},\mathbf{y},r} \; r \\
&\text{s.t.} \quad g_k(\mathbf{x}^j,\mathbf{y}^j) + \nabla g_k(\mathbf{x}^j,\mathbf{y}^j)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^j \\ \mathbf{y}-\mathbf{y}^j \end{bmatrix} + r\left\|\nabla g_k(\mathbf{x}^j,\mathbf{y}^j)\right\|_2 \leq 0, \quad j=1,\ldots,i,\; k\in K_j, \\
&\mathbf{x}\in\mathbb{R}^n, \mathbf{y}\in\mathbb{R}^m, r\in\mathbb{R},
\end{aligned}
\tag{4.4}
$$

where r is the radius of the inscribed ball. To the authors' best knowledge, it was first mentioned in [228] that the Chebyshev center of a polyhedral set could be obtained by solving an LP problem. For more details on how to find the Chebyshev center of a polyhedral set see *e.g.,* [229]. To simplify notation we introduce a new set $\mathcal{B}_i$ defined as

$$
\mathcal{B}_i = \left\{ g_k(\mathbf{x}^j,\mathbf{y}^j) + \nabla g_k(\mathbf{x}^j,\mathbf{y}^j)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^j \\ \mathbf{y}-\mathbf{y}^j \end{bmatrix} + r\left\|\nabla g_k(\mathbf{x}^j,\mathbf{y}^j)\right\|_2 \leq 0, \quad j=1,\ldots,i,\; k\in K_j \right\}, \tag{4.5}
$$

which contains all constraints defining set $\widehat{\mathcal{N}}_i$. In order to obtain a feasible solution of the MINLP problem, we also have to take the linear constraints and integer requirements into consideration. Therefore, a new trial solution will be chosen as the center of the largest ball

inscribed in set $\widehat{\mathcal{N}}_i$, with the restrictions that the center has to satisfy all linear constraints and integer requirements. Hence, the linear constraints and integer restrictions only affect the center's location and not directly the radius of the ball. A new trial solution is, thus, obtained by solving the following MILP problem

$$(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}, r^{i+1}) \in \underset{(\mathbf{x},\mathbf{y},r)\in\mathcal{B}_i\cap\mathcal{L}\cap\mathcal{Y}}{\arg\max} \; r. \tag{MILP-i}$$

Since we are maximizing the radius of the ball inscribed in $\widehat{\mathcal{N}}_i$, it results in a trial solution minimizing the left-hand side of the linearized constraints in equation (4.2). Once we have obtain a new trial solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ there are two possibilities: either it violates some of the nonlinear constraints or it is a feasible solution.

In case the trial solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ violates some of the nonlinear constraints, then we can improve the polyhedral approximation by generating cutting planes according to

$$g_k\left(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}\right) + \nabla g_k\left(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}\right)^\top \begin{bmatrix} \mathbf{x} - \mathbf{x}^{i+1} \\ \mathbf{y} - \mathbf{y}^{i+1} \end{bmatrix} \leq 0 \; \forall k \in K_{i+1}, \tag{4.6}$$

where K_{i+1} is the index set of all active and violated constraints. The new cutting planes will exclude the solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ from the search space, see *e.g.*, [120]. The new cutting planes are added to the polyhedral approximation, and we denote the new approximation as $\widehat{\mathcal{N}}_{i+1}$. The accumulation of cutting planes improves the polyhedral approximation of set $\mathcal{N}$. In the next iteration, we solve subproblem (MILP-i) updated with new cuts to obtain a new trial solution.

Now, in case the trial solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ is feasible, it may still not be the best possible one with the integer combination given by $\mathbf{y}^{i+1}$. Therefore, we will fix the integer variables in the original MINLP problem to the values given by $\mathbf{y}^{i+1}$, resulting in the following convex

NLP problem

$$(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}) \in \underset{(\mathbf{x},\mathbf{y})\in\mathcal{N}\cap\mathcal{L}\cap\mathcal{Y}}{\arg\min} \; \mathbf{c}_1^\top \mathbf{x} + \mathbf{c}_2^\top \mathbf{y} \quad \text{s.t.} \quad \mathbf{y} = \mathbf{y}^{i+1}. \tag{NLP-fixed}$$

By solving problem (NLP-fixed) we obtain the optimal solution for this specific integer combination. However, the obtained solution may still not be the optimal one to the original MINLP problem. In order to obtain better solutions, we will therefore, generate an objective cut according to

$$\mathbf{c}_1^\top \mathbf{x} + \mathbf{c}_2^\top \mathbf{y} \leq \mathbf{c}_1^\top \mathbf{x}^{i+1} + \mathbf{c}_2^\top \mathbf{y}^{i+1}, \tag{4.7}$$

where $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ is the solution obtained by solving subproblem (NLP-fixed). The cut obtained by equation (4.7) will exclude all solutions that have a worse objective function value than the obtained feasible solution, and will thus reduce the search space. To obtain better solutions, we include the objective cut in the polyhedral approximation $\widehat{\mathcal{N}}_{i+1}$. Subproblem (MILP-$i$), by which we choose the new trial solutions, will then contain the objective cut given by equation (4.7) in the following form

$$\mathbf{c}_1^\top \mathbf{x} + \mathbf{c}_2^\top \mathbf{y} + r\|(\mathbf{c}_1; \mathbf{c}_2)\|_2 \leq \mathbf{c}_1^\top \mathbf{x}^{i+1} + \mathbf{c}_2^\top \mathbf{y}^{i+1}, \tag{4.8}$$

forcing the next inscribed ball to also take the objective cut into consideration. As long as we obtain solutions of subproblems (MILP-i) with $r^i > 0$, it is clear that the constraint given by equation (4.8) will force the trial solutions to have a strictly lower objective function value than the obtained feasible solution. Thus, the objective cut will force the algorithm to search for better solutions. Once an objective cut has been added to the polyhedral approximation $\widehat{\mathcal{N}}_i$ it will no longer be an outer approximation of set $\mathcal{N}$. However, we know that the optimal solution will not be excluded from the search space due to convexity. The search space can be further reduced by generating cutting planes for all nonlinear constraints active at the feasible solution $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1})$ according to equation (4.6).

In each iteration, the radius of the ball inscribed in $\widehat{\mathcal{N}}_i$ is reduced since sets $\widehat{\mathcal{N}}_i$ shrink due to adding cuts. Later, we prove that the cuts added in each iteration force the radius to converge to zero. If the radius of the largest ball inscribed in $\widehat{\mathcal{N}}_i$ is zero, set $\widehat{\mathcal{N}}_i$ has an empty interior, thus verifying that the optimal solution has been found. In case the original MINLP problem is infeasible, the radius will converge to zero without finding any feasible solution. The convergence properties are discussed in more detail in Section 4.5.

The Center-cut algorithm is summarized as a pseudo-code in Algorithm 3. In the algorithm, we use the radius as an optimality measure since a smaller radius will ensure better solutions. However, to guarantee that the best found solution is optimal, we must continue until the radius is reduced to zero.

Algorithm 3 Pseudo-code of the Center-cut algorithm

Specify a tolerance $\delta \geq 0$.

1. Initialization.
   - 1.1 Set $\mathcal{B}_1 = \mathbb{R}^{n+m}$, set iteration counter $i = 1$.
   - 1.2 Solve problem (MILP-i) to obtain $(\mathbf{x}^1, \mathbf{y}^1)$ and r^1.
2. While $r^i > \delta$.
   - 2.a If $(\mathbf{x}^i, \mathbf{y}^i)$ satisfies all nonlinear constraints:
     - Solve problem (NLP-fixed) to obtain the optimal solution with the given integer solution and store the solution as $(\mathbf{x}^i, \mathbf{y}^i)$.
     - Construct cutting planes for any active constraint according to equation (4.6) and the objective cut according to equation (4.7).
     - Generate set $\mathcal{B}_{i+1}$ by adding the new cuts to $\mathcal{B}_i$
   - 2.b If $(\mathbf{x}^i, \mathbf{y}^i)$ does not satisfies all nonlinear constraints:
     - Obtain cutting planes for all violated constraints according to equation (4.6).
     - Generate set $\mathcal{B}_{i+1}$ by adding all cutting planes to $\mathcal{B}_i$
   - 2.c Solve problem (MILP-i) to obtain $(\mathbf{x}^{i+1}, \mathbf{y}^{i+1}, r^{i+1})$. and set $i = i + 1$.
3. Return the best found feasible solution $(\mathbf{x}^i, \mathbf{y}^i)$.

Note that if the original MINLP problem has a convex nonlinear objective function, we can simply replace the left-hand side of the objective cut in equation (4.7) by a linearization

of the objective. There is, therefore, no need to reformulate the problem to obtain a linear objective function.

In the next section, we apply the Center-cut algorithm to an illustrative example with two variables to give a geometric interpretation of the algorithm.

4.4 Illustrative example

For MINLP problems with only two variables, the Center-cut algorithm chooses the trial solutions by inscribing the largest possible circle in $\widehat{\mathcal{N}}_i$, such that the center of the circle satisfies all linear constraints and integer requirements. To illustrate the basics of the Center-cut algorithm, consider the following simple MINLP problem,

$$\begin{aligned} \min \quad & -3x - y \\ \text{s.t.} \quad & x^2 + y^2 \leq 25, \\ & x^2 + (5-y)^2 \leq 36, \\ & (6-x)^2 + y^2 \leq 36, \\ & 0 \leq x \leq 10, \quad 0 \leq y \leq 10, \\ & x \in \mathbb{R},\ y \in \mathbb{Z}. \end{aligned} \tag{Ex 1}$$

The MINLP problem (Ex 1) is illustrated in Figure 4.1. Application of the Center-cut algorithm (Algorithm 3) to the illustrative example problem (Ex 1) gives the following iterations which are also depicted in Figure 4.2. In the first iteration, we have no cutting planes approximating set $\mathcal{N}$ and the radius is therefore not limited, and any solution satisfying the linear constraints and integer requirement can be chosen. In the second iteration, we obtain a solution where the circle center satisfies all constraints. The solution is improved by solving problem (NLP-fixed). In iteration two, we generate an objective cut

according to equation (4.7) and a cutting plane for the second nonlinear constraint. The optimal solution is obtained in iteration 4, but we need an additional iteration to verify optimality. In iteration 5, we find that the largest inscribed circle has a radius of zero, proving that the optimal solution has been found. As a comparison, it takes nine iterations with the basic ECP algorithm to find a feasible solution and three iterations with OA.

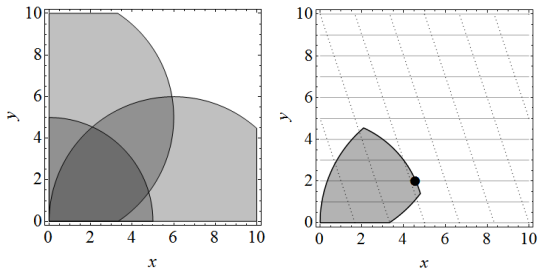
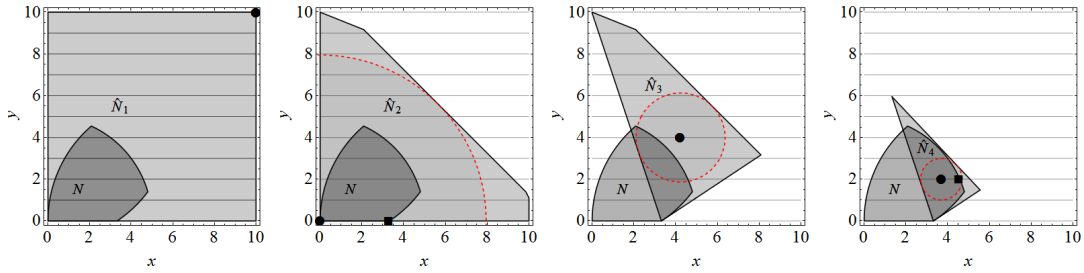


Figure 4.1: The figure to the left shows the feasible regions defined by the nonlinear constraint of problem (Ex 1). The second figure shows the feasible region defined by the constraints, contours of the objective function and the optimal solution. The horizontal lines correspond to integer values of the integer variable y.



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3 (d) Iteration 4

Figure 4.2: Applying the Center-cut algorithm to problem (Ex 1) results in five iterations, and the first four iterations are shown in the figures.The figures show the feasible region defined by the nonlinear constraints and the region defined by sets $\widehat{N}_i$. The circular dot represents the center of the inscribed circle and the dashed curves represent the circle. The solution obtained by solving subproblem (NLP-fixed) is shown by the squared dot. The horizontal lines correspond to integer values of the integer variable y.

In the next section, we describe why the radius of the inscribed ball can be used as

an optimality measure, and we prove that the Center-cut algorithm will find an optimal solution to the MINLP problem in a finite number of iterations.

4.5 Proof of convergence

Here we focus on the convergence properties of the Center-cut algorithm. We show that the radius of the inscribed ball converges to zero. From there, we can show that the algorithm will converge to the optimal solution of a convex MINLP problem.

To prove that the radius of the inscribed balls will converge to zero, we need some properties presented in Lemma 4.5.1.

Lemma 4.5.1. In the Center-cut algorithm, in iteration i the radius r^i of ball inscribed in $\widehat{\mathcal{N}}_i$ will be bounded by distance between the current center $\left(\mathbf{x}^i, \mathbf{y}^i\right)$ and any previously obtained center $\left(\mathbf{x}^{i-l}, \mathbf{y}^{i-l}\right)$ according to

$$r^i \leq \left\|(\mathbf{x}^i - \mathbf{x}^{i-l}; \mathbf{y}^i - \mathbf{y}^{i-l})\right\|_2,$$

where $0 < l < i$.

Proof. At iteration $i - l$, we generate cuts according to equation (4.6) and if we obtain a feasible solution we also add an objective cut according to equation (4.7). These cuts will either exclude the center $\left(\mathbf{x}^{i-l}, \mathbf{y}^{i-l}\right)$ from set $\widehat{\mathcal{N}}_{i-l+1}$ or result in a cut which passes through it. The center $\left(\mathbf{x}^{i-l}, \mathbf{y}^{i-l}\right)$ is, thus, either outside $\widehat{\mathcal{N}}_{i-l+1}$ or on the boundary of $\widehat{\mathcal{N}}_{i-l+1}$. Therefore, the radius r^i cannot be greater than the distance from the current center $\left(\mathbf{x}^i, \mathbf{y}^i\right)$ to the previously obtained center $\left(\mathbf{x}^{i-l}, \mathbf{y}^{i-l}\right)$, otherwise parts of the ball would be outside of $\widehat{\mathcal{N}}_{i-l+1}$. □

By using the bounds on the radius given by Lemma 4.5.1, it is possible to prove that the radius converges to zero as described by the following theorem.

Theorem 4.5.1. In the Center-cut algorithm, the radii r^i of the inscribed balls converge to zero when $i \to \infty$.

Proof. Assume that the algorithm does not stop and an infinite sequence of centers $\left\{\mathbf{x}^i, \mathbf{y}^i\right\}_{i=1}^{\infty}$ is obtained. Due to Assumption 2, all of the centers in the sequence belong to a compact subset of $\mathbb{R}^{n+m}$. According to the Bolzano-Weirstrass theorem, the sequence must contain at least one convergent subsequence $\left\{\mathbf{x}^{i_j}, \mathbf{y}^{i_j}\right\}_{i_j=1}^{\infty}$. The convergent subsequence also forms a Cauchy sequence with the following property

$$\lim_{j\to\infty} \left\|(\mathbf{x}^{i_j} - \mathbf{x}^{i_j-1}; \mathbf{y}^{i_j} - \mathbf{y}^{i_j-1})\right\|_2 = 0.$$

Note that Lemma 4.5.1 is true for any two centers and, thus, $\lim_{i\to\infty} r^i = 0$. □

To prove that the algorithm obtains the optimal solution in a finite number of iterations, we need some other properties presented in Lemma 4.5.2, regarding the geometry of the feasible region. The proof of Lemma 4.5.2 follows from Slater's condition, but for the sake of completeness, we have included the proof. Here we denote the optimal value of the objective function of the MINLP problem as z^*.

Lemma 4.5.2. For any $\epsilon > 0$, $\exists\, r > 0, (\mathbf{x}^c, \mathbf{y}^c) \in \mathcal{N} \cap \mathcal{L} \cap \mathcal{Y}$ such that $\mathcal{B}_r(\mathbf{x}^c, \mathbf{y}^c) \subset \mathcal{N}^*$, where

$$\mathcal{N}^* = \left\{(\mathbf{x}, \mathbf{y}) \mid \mathbf{c}_1^\top \mathbf{y} + \mathbf{c}_2^\top \mathbf{x} \le z^* + \epsilon,\ g_k(\mathbf{x}, \mathbf{y}) \le 0 \quad \forall\, k\right\}, \tag{4.9}$$

$$\mathcal{B}_r(\mathbf{x}^c, \mathbf{y}^c) = \left\{(\mathbf{x}, \mathbf{y}) \;\middle|\; \left\|(\mathbf{x}^c - \mathbf{x}; \mathbf{y}^c - \mathbf{y})\right\|_2 \le r\right\}. \tag{4.10}$$

Proof. Note that z^* is given by $z^* = \mathbf{c}_1^\top \mathbf{x}^* + \mathbf{c}_2^\top \mathbf{y}^*$, where $(\mathbf{x}^*, \mathbf{y}^*)$ is an optimal solution of the MINLP problem. The optimal solution strictly satisfies the restriction on the objective function

$$\mathbf{c}_1^\top \mathbf{x}^* + \mathbf{c}_2^\top \mathbf{y}^* < z^* + \epsilon.$$

However, the optimal solution might be located on the boundary of set $\mathcal{N}^*$. Therefore it cannot be used as the center of the ball $\mathcal{B}_r$. By Assumption 3, the nonlinear constraints

satisfy Slater's condition even if the integer variables are fixed to $\mathbf{y}^*$, *i.e.*, $\exists\, \bar{\mathbf{x}} : \mathbf{A}\bar{\mathbf{x}} + \mathbf{B}\mathbf{y}^* \leq \mathbf{b}$, $g_k(\bar{\mathbf{x}}, \mathbf{y}^*) < 0\ \forall\ k$. Next, a new point is define as

$$\widehat{\mathbf{x}} = \alpha \mathbf{x}^* + (1-\alpha)\bar{\mathbf{x}}, \tag{4.11}$$

where $\alpha \in [0,1)$ is an interpolation parameter. Now, a value for α is chosen such that $(\widehat{\mathbf{x}}, \mathbf{y}^*)$ strictly satisfies all constraints in set $\mathcal{N}^*$. If $\mathbf{c}_1^\top(\bar{\mathbf{x}} - \mathbf{x}^*) < \epsilon$, it is sufficient to choose $\alpha = 0$ to strictly satisfy the objective constraint in $\mathcal{N}^*$. Otherwise α can be chosen as $\alpha = \left(\frac{\epsilon}{2} + \mathbf{c}_1^\top(\mathbf{x}^* - \bar{\mathbf{x}})\right)/\mathbf{c}_1^\top(\mathbf{x}^* - \bar{\mathbf{x}}) < 1$, which results in

$$\mathbf{c}_1^\top \widehat{\mathbf{x}} + \mathbf{c}_2^\top \mathbf{y}^* = z^* + \frac{\epsilon}{2}.$$

Since $\widehat{\mathbf{x}}$ was chosen as an interpolation between two points, with the same integer combination and both satisfy all the constraints, it is clear that $(\widehat{\mathbf{x}}, \mathbf{y}^*)$ will satisfy all constraints. Furthermore, since $(\bar{\mathbf{x}}, \mathbf{y}^*)$ strictly satisfies the nonlinear constraints and $\alpha < 1$ we get

$$g_k(\widehat{\mathbf{x}}, \mathbf{y}^*) < 0\ \forall\ k.$$

The point $(\widehat{\mathbf{x}}, \mathbf{y}^*)$ is, thus, located within the interior of set $\mathcal{N}^*$, and therefore it is possible to put a ball with a nonzero radius at $(\widehat{\mathbf{x}}, \mathbf{y}^*)$ such that the entire ball is contained in set $\mathcal{N}^*$. □

Now, we have all the intermediate results needed for proving that the optimal solution is obtained in a finite number of iterations.

Theorem 4.5.2. The Center-cut algorithm obtains the optimal solution of problem (P-MINLP) in a finite number of iterations.

Proof. As before the optimal solution of the MINLP problem is denoted as $(\mathbf{x}^*, \mathbf{y}^*)$ and the optimal objective value as z^*. Then, $\exists \epsilon > 0$, such that $\mathcal{N}^* \cap \mathcal{L} \cap \mathcal{Y}$ only contains optimal values for the integer variables $\mathbf{y}$, where $\mathcal{N}^*$ is given by equation (4.9). Lemma 4.5.2 states

that a ball with radius $r^* > 0$ can be inscribed in set $\mathcal{N}^*$, such that the center satisfies all constraints.

As long as the algorithm has not obtained the optimal solution, the entire set $\mathcal{N}^*$ will be contained within $\widehat{\mathcal{N}}_i$, *i.e.*, $\mathcal{N}^* \subset \widehat{\mathcal{N}}_i$. This is true because all the cutting planes added according to equation (4.6) are overestimating the feasible region defined by the nonlinear constraints. As long as $\mathcal{N}^* \subset \widehat{\mathcal{N}}_i$, the radius of the inscribed balls will be greater or equal to r^*. Theorem 4.5.2 states that the radius of the inscribed balls converges to zero, and therefore there exists a finite integer p such that

$$\forall i \geq p \quad r^i < r^*.$$

The only way to reduce the radius below r^* is to generate an objective cut according to equation (4.7) stricter than the objective cut in $\mathcal{N}^*$. Such an objective cut must be generated in iteration p at a feasible solution $(\mathbf{x}^p, \mathbf{y}^p)$ such that $\mathbf{c}_1^\top \mathbf{x}^p + \mathbf{c}_2^\top \mathbf{y}^p < z^* + \epsilon$. In the beginning, ϵ was chosen such that $\mathcal{N}^* \cap \mathcal{L} \cap \mathcal{Y}$ only contains the optimal integer combination, *i.e.*, $\mathbf{y}^p = \mathbf{y}^*$. Furthermore, the variables in iteration p will be chosen by solving subproblem (NLP-fixed) with the integer variables fixed as $\mathbf{y}^*$, and the subproblem will then return an optimal solution for the continuous variables, *i.e.*, $(\mathbf{x}^p, \mathbf{y}^p) = (\mathbf{x}^*, \mathbf{y}^*)$. The optimal solution of the MINLP problem was, thus, obtained in iteration p. □

From the proof of Theorem 4.5.2, it follows that the optimal solution will be obtained once the radius of the inscribed ball is reduced below a particular value. Furthermore, if the radius is reduced to zero, it verifies the optimality of the best-found solution. In the algorithm, we, therefore, use the radius as an optimality measure and termination criterion. For rigorously verifying optimality, the radius needs to be reduced to zero; however, in practice, it is often sufficient to stop once the radius is reduced to a given tolerance δ.

This section has proven that the algorithm will find the optimal solution to any convex

MINLP problem satisfying Assumptions 1, 2, and 3. The following section deals with some details regarding the implementation of the algorithm.

4.6 Implementing the algorithm

In previous sections, we have described the basics of the Center-cut algorithm. To test the functional performance of the Center-cut algorithm, we implemented it in Matlab 2017a and used Ipopt 3.12.7 [158] and Gurobi 7.5.2 as subsolvers for the NLP and MILP subproblems, respectively. We have also used OPTI Toolbox [99] to read the test problems. When implementing the Center-cut algorithm, it is possible to incorporate some tricks and features from other algorithms and solvers; next, we describe some of these that can easily be exploited.

First, when solving an MINLP problem with the Center-cut algorithm, it is unnecessary to solve every single MILP subproblem to optimality. It is sufficient to obtain a feasible solution, such that the inscribed ball has a radius strictly greater than zero. This is an essential detail since solvers such as CPLEX or Gurobi can quickly find several feasible solutions to an MILP problem. Often, the majority of the solution time is spent on proving optimality. The MILP subproblems are also by far the most time-consuming part of the Center-cut algorithm. By stopping the MILP solver after a specific number of found feasible solutions, it is often possible to significantly reduce the solution time while still obtaining reasonable solutions of the mixed-integer subproblems. This can be done with Gurobi by using the solution limit parameter. In implementing the Center-cut algorithm, we simply start with the solution limit parameter set to 2 and increase the solution limit parameter by one each time the radius of the inscribed ball is less than 0.001 and the solution was not reported as optimal. We also use a second test for increasing the solution limit, where the

solution limit is increased if the radius is less than half the radius in the previous iteration and the solution was not optimal. When choosing the solution limit by this technique, we start with a low solution limit and gradually increase it during the iterations to obtain good solutions to the subproblems. When using this technique, one must be careful with the termination criterion. The search should not be terminated unless the MILP subproblem was solved to optimality in the last iteration. A similar approach for speeding up the MILP subproblems and consequently speeding up the MINLP solution procedure is also used with the GAECP algorithm, see [120].

It might also be possible to speed up the algorithm by solving additional NLP subproblems in some cases. Even if the trial solution $(\mathbf{x}^i, \mathbf{y}^i)$ obtained by solving subproblem (MILP-i) does not satisfy the nonlinear constraints, $\mathbf{y}^i$ may still be a feasible integer combination. It might, therefore, be possible to obtain a feasible solution by fixing the integer variables to $\mathbf{y}^i$ and solving subproblem (NLP-fixed). This situation is illustrated in iteration 3 in Figure 4.2, where it would have been possible to obtain a feasible solution by solving an NLP subproblem. By solving such NLP problems, it may be possible to obtain feasible solutions more frequently. However, the additional NLP problems may also take some time to solve. In implementing the Center-cut algorithm, we try to fix the integer variables and solve subproblem (NLP-fixed) in every third iteration. A similar technique is used in both the GAMS solver AlphaECP [230] and in the SHOT solver [121]. The NLP problems with fixed integers may be infeasible in some iterations; however, in this case, Ipopt returns a solution that minimizes the constraint violation with the specific integer combination. By adding cuts according to equation (4.6) at the infeasible solution returned by Ipopt, we can exclude the infeasible integer combination from the search space. For details on such cuts, see [25]. In implementing the Center-cut algorithm, we use this technique for generating cuts when the NLP solver cannot find a feasible solution.

In the implementation, we have used the Center-cut algorithm as described in Algorithm 3 together with the additional features described here. With the NLP solver Ipopt, we have used the default settings for all parameters. With Gurobi, we have used the solution limit strategy as described earlier. For the other parameters, we have used default settings.

4.7 Numerical results

To test the functional performance of the Center-cut algorithm, we considered some convex MINLP test problems taken from the library MINLPLib2 [231]. We have used a standard desktop computer with an Intel i7 processor and 16GB of RAM for the tests.

First, we have chosen 8 test problems from MINLPLib 2 that represent several types of MINLP problems, such as some facility layout problems [232], retrofit planning problems [233] and trim loss problems [234]. The largest of these problems contains 1500 binary variables, 1500 continuous variables, and 1821 constraints. These specific problems were chosen since they are known to be challenging to solve. We have applied the Center-cut implementation to the test problems, and the results are shown in Table 4.1. The table shows the time and number of iterations needed to find a feasible solution, a second feasible solution, a solution within 5% of the best-known solution, a solution within 1% of the best-known solution, and the quality of the best found solution. Besides the settings described in the previous section, we have used a time limit of 1800 seconds.

To get a reference point for evaluating the performance, we have used the feasibility pump (FP) available in the state-of-the-art solver DICOPT in GAMS [214] on the same problems. As previously mentioned, the feasibility pump is a primal heuristic intended for quickly finding good solutions. The results obtained with the feasibility pump are shown in Table 4.2. With the feasibility pump, we have used CONOPT and Gurobi as

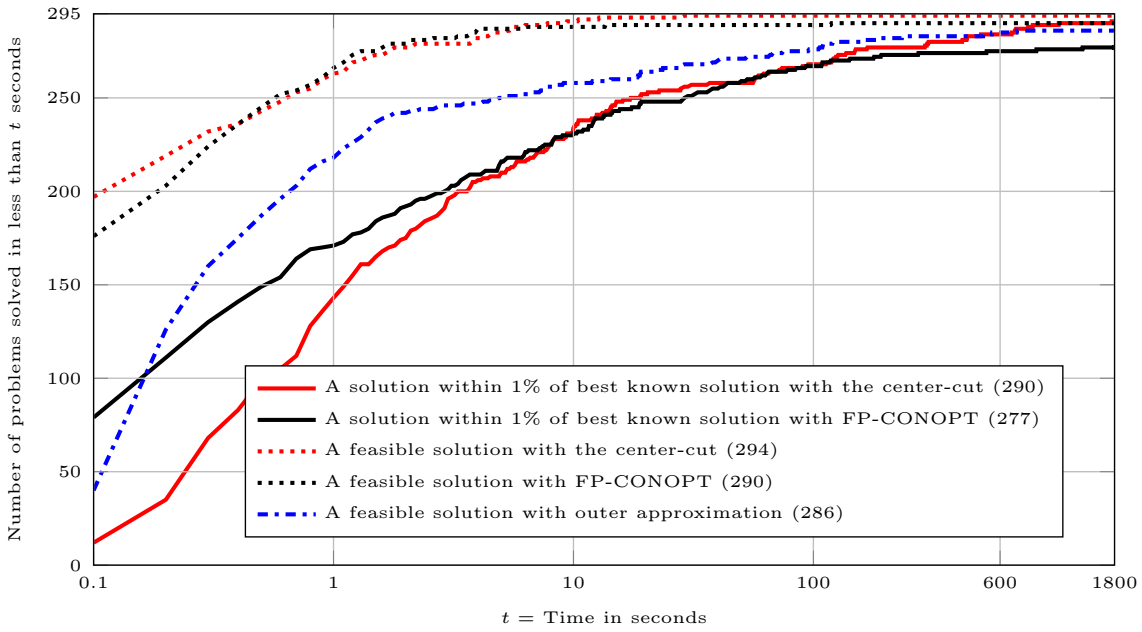


Figure 4.3: The lines show the number of problems that the Center-cut implementation, feasibility pump, and OA can find a solution to as a function of running time. The lines do not correspond to the cumulative solution time but show how many of the problems the algorithms can obtain a solution to within a specific time. The number in the parenthesis shows the total number of problems where a solution was obtained.

subsolvers, and we have used the following parameters to make sure we can obtain a solution within 1% of the optimal solution: `fp_cutoffdecr` = 0.01, `fp_timelimit` = 1800, `fp_stalllimit` = 10000 and `fp_sollimit` = 10000. We used Conopt as an NLP subsolver with the feasibility pump since it resulted in significantly better performance compared to using Ipopt, see Figure 4.4. Note that this is not intended directly as a comparison. The feasibility pump in DICOPT and the Matlab implementation of the Center-cut algorithm are difficult to compare in terms of solution time directly due to the different implementation environments. The Matlab implementation is quite simple and mainly intended as a proof of concept and shows the Center-cut algorithm's potential.

Table 4.1 shows that the Center-cut implementation can find a feasible solution to all of

the eight problems within less than 3 seconds. Furthermore, we can find good solutions to all of the problems except `tls7`, where the best-obtained solution is still far from the best-known solution. However, it should be noted that these are complex problems, and `tls7` is one of the few convex MINLP problems in MINLPLib2 that are still considered unsolved. The feasibility pump struggles with some of the problems. It is not able to find any solution for two of the problems. The feasibility pump is quicker at obtaining solutions of problems `gams01` and `o7_ar2_1`. However, for the other problems, the Center-cut implementation seems to be more efficient.

To further test the Center-cut implementation, we applied it to 295 test problems from MINLPLib2. In this test set, we chose all convex problems from MINLPLib2 containing at least one discrete variable. We removed the instances where the only nonlinearity was due to a quadratic objective function. Convex problems where the only nonlinear term is a quadratic objective can be solved efficiently directly with Gurobi. For such problems, the Center-cut algorithm is not necessarily a good choice, and therefore, we removed these test problems. The results obtained with the Center-cut implementation are presented in Figure 4.3. We have applied the feasibility pump to the test problems to get a reference point to evaluate the results. We have also used a basic implementation of OA, described in [89], to show the time needed to obtain a feasible solution with OA.

Figure 4.3 shows the number of problems that the Center-cut implementation can find a feasible solution to as a function of time. The figure is not based on the cumulative solution time. One of the lines represents the number of problems the algorithm can obtain a solution to within that specific time. The figure shows that the Center-cut implementation can find feasible solutions to these problems quickly. 250 test problems can find a feasible solution by running Center-cut implementation for less than 1 second. Furthermore, the implementation requires less than 10 seconds to find a feasible solution to 291 of the 295

test problems. There is only one problem (`tls12`) where the implementation fails to find a feasible solution. Compared with the feasibility pump, the Center-cut can find a feasible solution for more problems in 0.1 s, and the overall performance is similar. Within the given time limit, the Center-cut can find feasible solutions for all but one of the test instances. At the same time, the feasibility pump is not able to find any feasible solution to five of the instances. The figure also shows that OA is significantly slower at obtaining feasible solutions.

Finding a solution within 1% of the best-known solution requires more time. By running the Center-cut implementation for 10 seconds, finding a solution within the 1% tolerance for 236 of the test problems is possible. Within the given time limit, we managed to find a solution within the tolerance for 290 test problems. The feasibility pump is faster at obtaining solutions within the 1% tolerance for the easier test problems, partially due to a more efficient implementation. For the instances requiring more than 3 seconds, the feasibility pump and Center-cut performance are pretty similar. However, the feasibility pump fails to find a solution within the tolerance for 17 test problems in the end. In contrast, the Center-cut implementation only fails on 5 of the problems.

To show that the test problems considered here are not trivial, we have also solved them with the solvers BARON, DICOPT, and SBB in GAMS, and results are presented in the Appendix. To get a more comprehensive comparison between the feasibility pump and the Center-cut algorithm, we have also compared them in terms of the total number of iterations needed; the result is shown in Figure 4.4. However, the results are very similar to the comparison in solution time, and, therefore, the comparison based on iterations is only included in the Appendix.

The numerical results are mainly intended as a proof of concept and show the potential of the Center-cut algorithm. The results are promising and show that the simple Matlab

implementation of the Center-cut algorithm is actually on par with the feasibility pump in the state-of-the-art solver DICOPT.

The numerical results have shown that the Center-cut algorithm may be well suited as a primal heuristic. For the test problems, we quickly found feasible solutions to almost all of the 295 test problems. Furthermore, we are also able to obtain solutions of good quality.

4.8 Conclusions

In this chapter, we have given a detailed presentation of the Center-cut algorithm for convex MINLP. We have proven that the algorithm finds the optimal solution in a finite number of iterations. The algorithm uses a different approach to obtain trial solutions that should quickly obtain feasible solutions, and the numerical results verified this. The ability to quickly obtain feasible solutions makes the algorithm well suited as a primal heuristic. However, it can also be used as a deterministic solution technique. With the Center-cut algorithm, we do not directly obtain a lower bound as in ESH or ECP. Therefore it could be efficient to combine these algorithms in a solver to obtain both a lower and upper bound.

4.A Detailed Performance Results

Section 4.7 mentions that the feasibility pump in DICOPT performed better when using CONOPT instead of Ipopt as NLP subsolver. Figure 4.4 shows a clear difference between using the two subsolvers. However, we have been informed that an issue in the implementation could partially cause the difference, and it should be fixed in the next GAMS release.

To show that the test problems are not trivial to solve, we have also tried to solve them using the solvers BARON, DICOPT, and SBB in GAMS. With these solvers, we have used

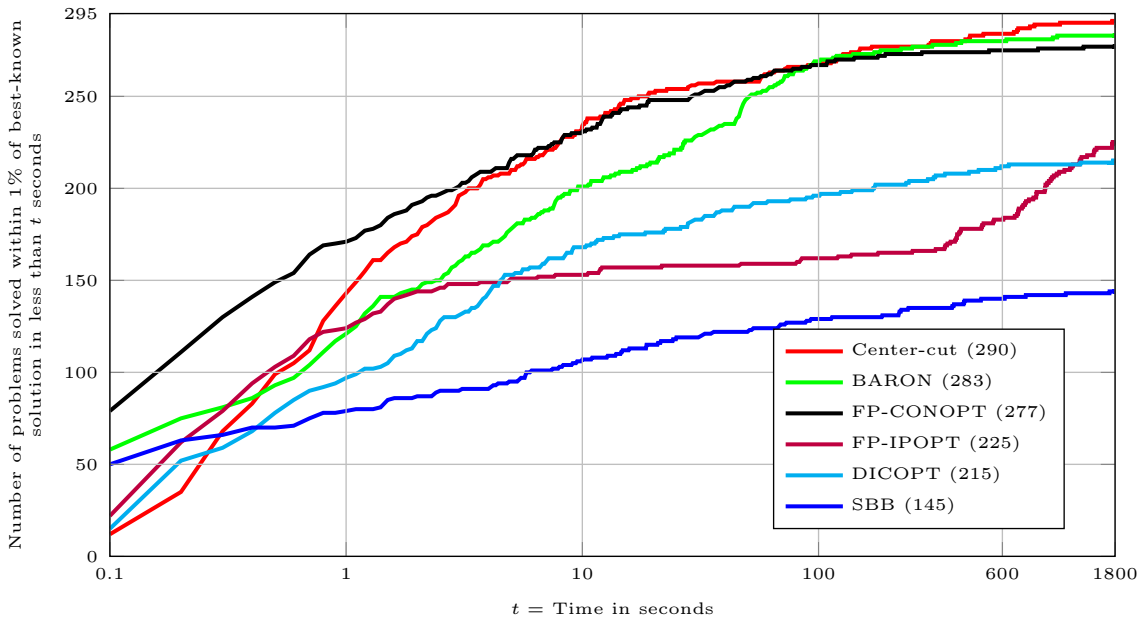


Figure 4.4: The lines show the number of problems that the Center-cut implementation, feasibility pump, and the solvers BARON, DICOPT, and SBB can find a solution to as a function of running time. The lines do not correspond to the cumulative solution time. However, they show how many individual problems the solvers can solve within the given time. The number in the parenthesis shows the total number of problems where a solution was obtained.

Gurobi and CONOPT as subsolvers. We have used default settings, except that we have changed the iteration limit to prevent the solvers from terminating prematurely. Also, if the solver has a convex strategy, then the strategy was activated. The results obtained with these solvers are shown in Figure 4.4, together with the results obtained by the Center-cut implementation and the feasibility pump. The Center-cut implementation can obtain a solution within 1% of the optimum for eight problems more than BARON. However, one should keep in mind that BARON, DICOPT, and SBB also obtain bounds on the optimum. Such optimality guarantees are provided by neither the Center-cut algorithm nor by the feasibility pump.

Section 4.7 mentions that it is difficult to compare the Center-cut implementation and the

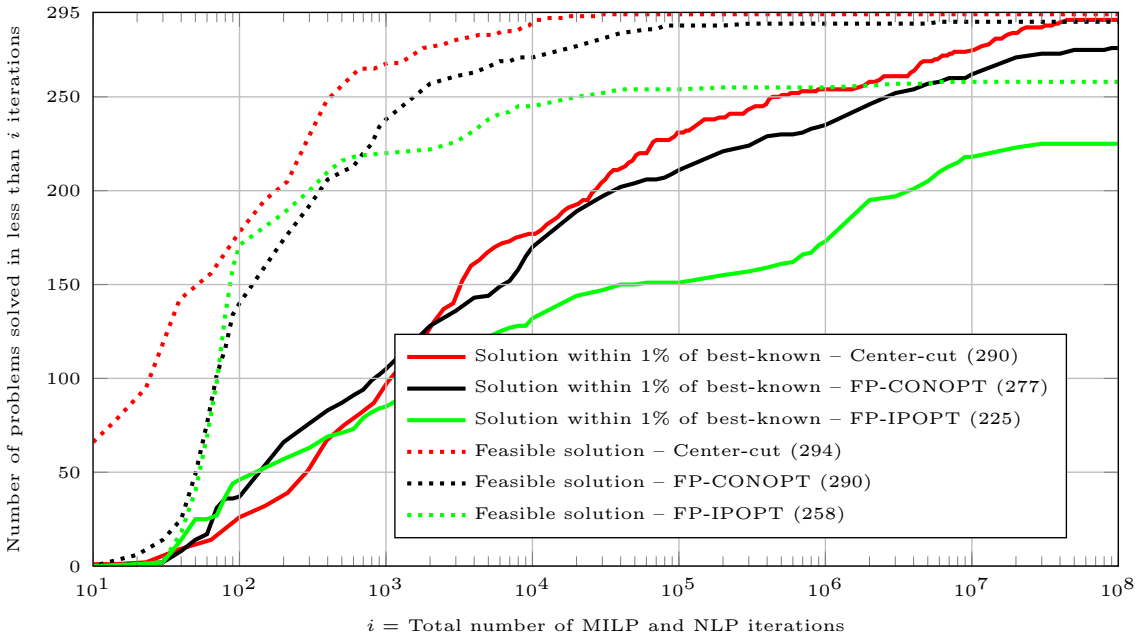


Figure 4.5: The lines show the number of problems that the feasibility pump and Center-cut algorithm can find a solution to as a function of the total number of iterations. The total number of iterations refers to all iterations, including the iterations performed by the MILP and NLP solver for each individual problem.

feasibility pump in terms of solution time. This is partially true as they are implemented in different environments that can affect the solution time. To avoid differences due to the implementations, the algorithms are also compared based on the total number of iterations in Figure 4.5. The total number of iterations includes all iterations performed by the MILP and NLP subsolver. The total number of iterations is comparable between the algorithms since both solve similar MILP and NLP subproblems. However, the MILP subproblems in the feasibility pump will, in general, contain more variables and constraints than those in the Center-cut algorithm. An iteration in the feasibility pump may, therefore, be computationally more demanding than a Center-cut iteration.

Table 4.1: The table shows the results obtained with the Center-cut implementation. The sign * indicates that no such solution was obtained within the time limit of 1800 seconds.

Results obtained by the Center-cut implementation

Name of MINLP problem	flay06h	gams01	ibs2	o7_ar2_1
Time / iterations to find a a feasible solution.	0.2 s / 2	0.3 s / 2	0.9 s / 4	0.6 s / 4
Time / iterations to find a second feas. sol.	*	0.7 s / 4	5.9 s / 10	2.0 s / 6
Time / iterations to find a sol. within 5% of opt.	0.2 s / 2	12 s / 23	91 s / 124	34 s / 12
Time / iterations to find a sol. within 1% of opt.	0.2 s / 2	139 s / 23	143 s / 146	60 s / 15
Sub-optimality of best-found solution.	0%	0.5%	1%	0%
Number of variables / discrete variables / constraints	567 / 60 / 694	146 / 110 / 1269	3011 / 1500 / 1822	113 / 42 / 270

Name of MINLP problem	rsyn0815m04h	stockcycle	tls6	tls7
Time / iterations to find a feasible solution.	0.2 s / 2	0.1 s / 1	0.7 s / 9	2.5 s / 14
Time / iterations to find a second feas. sol.	1.3 s / 3	0.7 s / 2	13 s / 36	5.0 s / 26
Time / iterations to find a sol. within 5% of opt.	8.8 s / 17	35 s / 102	830 s / 159	*
Time / iterations to find a sol. within 1% of opt.	9.4 s / 19	122 s / 184	*	*
Sub-optimality of best-found solution.	0.5%	0.5%	3%	33%
Number of variables / discrete variables / constraints	1797 / 367 / 3191	481 / 432 / 98	216 / 179 / 155	346 / 296 / 155

Table 4.2: The table shows the results obtained with the feasibility pump in DICOPT. The sign * indicates that no such solution was obtained within the time limit of 1800 seconds.

Results obtained by the feasibility pump in DICOPT

Name of MINLP problem	`flay06h`	`gams01`	`ibs2`	`o7_ar2_1`
Time / iterations to find a feasible solution.	0.2 s / 1	0.3 s / 1	*	0.8 s / 4
Time / iterations to find a second feas. sol.	0.3 s / 2	0.7 s / 2	*	0.9 s / 5
Time / iterations to find a sol. within 5%.	0.4 s / 3	8.3 s / 21	*	5.5 s / 11
Time / iterations to find a sol. within 1% of opt.	0.6 s / 5	37 s / 45	*	28 s / 13
Sub-optimality of best-found solution.	0%	0%	∞	0%
Number of variables / discrete variables / constraints	567 / 60 / 694	146 / 110 / 1269	3011 / 1500 / 1822	113 / 42 / 270

Name of MINLP problem	rsyn0815m04h	stockcycle	tls6	`tls7`
Time / iterations to find a feasible solution.	2.7 s / 1	0.1 s / 1	166 s / 48	*
Time / iterations to find a second feas. sol.	3.9 s / 3	0.2 s / 2	*	*
Time / iterations to find a sol. within 5% of opt.	5.7 s / 10	4.7 s / 51	*	*
Time / iterations to find a sol. within 1% of opt.	9.1 s / 14	226 s / 295	*	*
Sub-optimality of best-found solution.	0%	1%	12%	∞
Number of variables / discrete variables / constraints	1797 / 367 / 3191	481 / 432 / 98	216 / 179 / 155	346 / 296 / 155

Chapter 5

Outer-approximation with quadratic cuts*

5.1 Introduction

The applications of Mixed-Integer Nonlinear Programming (MINLP) models in Process Systems Engineering (PSE) are pervasive. The MINLP models are usually used to formulate problems such as synthesis, production planning, batch scheduling, operations optimization, and optimal control, among others [60, 236, 237]. The nature of the functions involved in optimization in PSE motivated the development of solution methods for generalized and specialized MINLP models. Convex MINLP models are of particular interest since their continuous relaxation gives rise to convex problems with a unique optimum value.

The most common methods for solving convex MINLP problems include decomposition methods, such as Outer-approximation (OA), Generalized Benders Decomposition (GBD), and Extended Cutting Planes (ECP), and Branch and Bound (B&B) methods. Methods for nonconvex MINLP problems include primarily spatial B&B methods [238]. Reviews at MINLP methods can be found in [104, 105, 200].

*Published as: Lijie Su, Lixin Tang, David E Bernal, and Ignacio E Grossmann. "Improved quadratic cuts for convex mixed-integer nonlinear programs". *Computers & Chemical Engineering* 109 (2018), pp. 77–95.

A general form of an MINLP is as follows,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & Z = f(\mathbf{x},\mathbf{y}), \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \\ & \mathbf{A}\mathbf{x} + \mathbf{E}\mathbf{y} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \mathbf{y} \in \mathcal{Y} \subseteq \mathbb{Z}^p, \end{aligned} \tag{MINLP}$$

where $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ is the objective function which may be nonlinear, and $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$ are the nonlinear inequality constraints. The matrices $\mathbf{A}$ and $\mathbf{E}$ are the coefficient matrices of the continuous and discrete variables in the linear constraints, respectively, and the vector $\mathbf{b}$ is the right-hand side of the linear inequalities. The sets $\mathcal{X}$ and $\mathcal{Y}$ are compact and bounded subsets of the n-dimensional Euclidean space and the p-dimensional integer lattice, respectively. The optimal solution to problem MINLP, if it exists, is denoted by $(\mathbf{x},\mathbf{y})$.

The discrete nature of the $\mathbf{y}$ variables makes the MINLP problems nonconvex, but a usual classification of MINLP models is based on the convexity of the objective function $f(\mathbf{x},\mathbf{y})$ and constraints $\mathbf{g}(\mathbf{x},\mathbf{y})$. If the objective function and the constraints are all convex, the MINLP can be classified as convex; otherwise, it would be classified as nonconvex.

This chapter addresses the solution of convex MINLP models based on OA and PSC methods using scaled quadratic cuts that underestimate the convex nonlinear functions. The paper is organized as follows. In Section 5.2, we present a brief background in MINLP solution methods, with a special focus on the OA and the PSC methods and the multi-generation cut strategy. Section 5.3 presents a motivation for this work, where a quadratic approximation to nonlinear functions can approximate more accurately the MINLP. Section 5.4 presents the quadratic approximation cuts and introduces the scaled quadratic cuts, which are proven to be valid underestimators for general convex nonlinear functions that appear in the objective function and the constraints. In Section 5.5 we present six improved

MINLP methods based on the scaled quadratic cuts, including the multi-generation strategies and PSC methods. Section 5.6 presents the numerical experiments of applying the proposed improvements to 44 benchmark MINLP problems and comparing their performances among them and with existing MINLP solvers. Finally, we give some conclusions for the improved OA and PSC methods with the proposed quadratic cuts.

5.2 Background

In the following section, we summarize different solution methods for MINLP, namely, Outer-approximation (OA) method, Partial Surrogate Cut (PCS) method, and multi-generation cuts strategy. The following assumptions are made:

1. The function $f : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}$ and vector functions $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^m$ are twice continuously differentiable convex functions, in which there is at least one nonlinear function.
2. $\mathbf{x}$ and $\mathbf{y}$ represent the continuous and discrete variables, respectively. The set $\mathcal{X}$ is a nonempty compact convex set, and the set $\mathcal{Y}$ is finite.
3. All NLP subproblems with fixed discrete variables of MINLP satisfy a constraint qualification [159].

These assumptions are required for the OA method and the nonlinear B&B to guarantee that problem MINLP and all its subproblems and relaxations are solved correctly [25, 30].

Here, the discrete variables $\mathbf{y}$ can be regarded as $\{0, 1\}$ binary variables. General bounded discrete variables can always be expressed in terms of binary variables.

5.2.1 MINLP solution methods

The general solution methods for convex MINLP problems are classified into two categories according to the structure of the algorithmic procedure. The first category of MINLP methods are the decomposition methods, which includes OA [30], GBD [28], and PSC [90]. The second category of MINLP methods are based on B&B [108] including LP/NLP based B&B [90, 114, 169]. The convex MINLP solution methods are used within global optimization algorithms for finding the globally optimal solution to nonconvex MINLP problems [238–241]. These methods rely on the convex relaxation of the nonconvex functions present in the problem, as in the αBB method [242]. In addition, ECP methods avoid the solution of NLP subproblems by solving a master MILP problem generated similarly as in the OA method but using the solution of the relaxed MILP problem as the following linearization point [29].

The idea of decomposition methods for convex MINLP is to decompose an MINLP into an NLP with fixed discrete variables and an MILP based on constructing cumulative relaxation cuts of nonlinear functions, which are the underestimators of the convex nonlinear functions. The NLP subproblem of an MINLP in minimization problems yields an upper bound assuming a feasible solution exists. The problem for fixed integer variables $\mathbf{y}^k$ is as follows,

$$\begin{aligned} \min_{\mathbf{x}} \quad & Z_{UB}^k = f(\mathbf{x}, \mathbf{y}^k), \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x}, \mathbf{y}^k) \leq \mathbf{0}, \\ & \mathbf{A}\mathbf{x} + \mathbf{E}\mathbf{y}^k \leq \mathbf{b}, \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n. \end{aligned} \tag{NLP($\mathbf{y}^k$)}$$

Let $\mathbf{x}^k$ be the solution to the NLP($\mathbf{y}^k$) problem, if feasible. Note that the solution $(\mathbf{x}^k, \mathbf{y}^k)$ is a feasible solution to problem MINLP. The MILP problem based on relaxation cuts of the nonlinear functions is called the master problem M^k shown later in this chapter and yields

a lower bound for minimization problems. By successively solving the NLP and the MILP in an iterative cycle, upper and lower bounds of the objective function are obtained, and the procedure is stopped when the bounds lie within a given tolerance. The differences in the decomposition methods lie in the MILP master problem, especially in the process of generating the relaxation cuts, which also determine the problem size of MILP and the predicted lower bounds for the MINLP. The detailed information of the master problem for OA and PSC is given in the following sections referred to as M^k_{OA} and M^k_{PSC} respectively.

5.2.2 Outer-approximation method

The OA method is a decomposition method proposed by Duran and Grossmann [30] for solving MINLP problems. In this method, the master problem is constructed with the accumulated linear cuts on the optimal solutions of NLP subproblems through the first-order Taylor series expansion of nonlinear functions. The master problem is given by the following MILP:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\eta} \quad & Z^k_{LB,OA} = \eta \\
\text{s.t.} \quad & \eta \geq f(\mathbf{x}^k,\mathbf{y}^k) + \nabla f(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix}, \quad k \in K, \\
& \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k) + \nabla \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix} \leq \mathbf{0}, \quad k \in K \qquad (M^k_{OA}) \\
& \mathbf{A}\mathbf{x} + \mathbf{E}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \mathbf{y} \in \mathcal{Y} \subseteq \mathbb{Z}^p, \eta \in \mathbb{R}.
\end{aligned}
$$

The solutions of the MILP are non-decreasing lower bounds of the MINLP. From a geometric point of view, an OA cut is a tangent line passing through one active nonlinear function in the MINLP problem. For the first iteration of the OA method, Viswanathan and Grossmann [124] proposed that the first linearizations of the problem be generated

based on the solution of the relaxed MINLP, in which the integer variables are treated as continuous,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & Z_{LB} = f(\mathbf{x},\mathbf{y}), \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \\ & \mathbf{A}\mathbf{x} + \mathbf{E}\mathbf{y} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \mathbf{y} \in \mathcal{Y} \subseteq \mathbb{R}^p. \end{aligned} \tag{rMINLP}$$

In case that the optimal solution of this problem satisfies the integrality constraints, no iterations are required as then the solution is found. In the case that problem rMINLP is infeasible and unbounded, the same can be assured for the original MINLP problem [124]. When fixing the integer variables $\mathbf{y}^k$, there is a possibility that the NLP($\mathbf{y}^k$) problem becomes infeasible. However, the values of the continuous variables $\mathbf{x}$ can still be used to generate valid linearizations if the functions are convex [90]. Furthermore, Fletcher and Leyffer [25] proposed to solve the following feasibility NLP subproblem after finding an infeasible NLP($\mathbf{y}^k$) to minimize the violation of the constraints,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{s}} \quad & \sum_{i=1}^{m} s_i, \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{y}^k) \leq \mathbf{s}, \\ & \mathbf{A}\mathbf{x} + \mathbf{E}\mathbf{y}^k \leq \mathbf{b}, \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \mathbf{s} \in \mathbb{R}^m_+. \end{aligned} \tag{F($\mathbf{y}^k$)}$$

5.2.3 Partial Surrogate Cuts method

Another decomposition method is the PSC method, proposed by Quesada and Grossmann [90]. In this method, the main idea is to partition the continuous variables into linear and nonlinear according to their presence in linear and nonlinear terms, respectively. This

allows us to rewrite problem MINLP as follows,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & Z = f(\mathbf{v},\mathbf{w},\mathbf{y}), \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{v},\mathbf{w},\mathbf{y}) \leq \mathbf{0}, \\ & \mathbf{A}_1\mathbf{v} + \mathbf{A}_2\mathbf{w} + \mathbf{E}\mathbf{y} \leq \mathbf{b}, \\ & \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \mathbf{y} \in \mathcal{Y} \subseteq \mathbb{Z}^p, \end{aligned} \tag{P}$$

where $\mathbf{v}$ are the nonlinear and $\mathbf{w}$ are the linear continuous variables, respectively. The coefficient matrix of the continuous variables in the linear constraints is also partitioned into $\mathbf{A}_1$ and $\mathbf{A}_2$ according to the linear and nonlinear continuous variables, respectively. This partition allows building a Lagrangean cut by projecting out the nonlinear terms. Introducing a real variable η representing the linearized objective, the cut is as follows,

$$\eta \geq f(\mathbf{v}^k,\mathbf{w},\mathbf{y}) + \begin{bmatrix} \lambda^k \\ -\mu^k \end{bmatrix}^\top \begin{bmatrix} \mathbf{g}(\mathbf{v},\mathbf{w},\mathbf{y}) & 0 \\ 0 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{v}-\mathbf{v}^k \end{bmatrix}, \tag{5.1}$$

where λ^k are the Lagrange multipliers of the nonlinear constraints including the nonlinear continuous variables $\mathbf{v}$, μ^k are the Lagrange multipliers of the linear constraints including nonlinear continuous variables $\mathbf{v}$, both obtained from the solution of problem rMINLP at the first iteration $k = 1$ and problem NLP($\mathbf{y}^k$) at iteration $k \geq 2$.

Contrary to OA, the linearizations based on the projection onto the nonlinear terms are not valid when the solution of problem NLP($\mathbf{y}^k$) is infeasible, in which case the following infeasibility cut is applied from the dual information of the nonlinear subproblem,

$$\begin{bmatrix} \lambda^k \\ -\mu^k \end{bmatrix}^\top \begin{bmatrix} \mathbf{g}(\mathbf{v},\mathbf{w},\mathbf{y}) & 0 \\ 0 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{v}-\mathbf{v}^k \end{bmatrix} \leq \mathbf{0}. \tag{5.2}$$

With these cuts, the master problem of the PSC method is as follows,

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\eta} \quad & Z^k_{LB,PSC} = \eta \\
\text{s.t.} \quad & \eta \geq f(\mathbf{v}^k,\mathbf{w},\mathbf{y}) + \begin{bmatrix} \lambda^k \\ -\mu^k \end{bmatrix}^\top \begin{bmatrix} \mathbf{g}(\mathbf{v}^k,\mathbf{w},\mathbf{y}) & 0 \\ 0 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{v}-\mathbf{v}^k \end{bmatrix}, \quad k \in K_F, \\
& \begin{bmatrix} \lambda^k \\ -\mu^k \end{bmatrix}^\top \begin{bmatrix} \mathbf{g}(\mathbf{v}^k,\mathbf{w},\mathbf{y}) & 0 \\ 0 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{v}-\mathbf{v}^k \end{bmatrix} \leq \mathbf{0}, \quad k \in K_I \qquad (M^k_{PSC}) \\
& \mathbf{A}\mathbf{x} + \mathbf{E}\mathbf{y} \leq \mathbf{b}, \\
& \begin{bmatrix} \mathbf{v} \\ \mathbf{w} \end{bmatrix} = \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \mathbf{y} \in \mathcal{Y} \subseteq \mathbb{Z}^p, \eta \in \mathbb{R},
\end{aligned}
$$

where K_F and K_I are the set of iterations where problem NLP($\mathbf{y}^k$) is feasible and infeasible, respectively.

The PSC method combines the OA and the GBD methods, where cuts are derived from gradient-based linearizations like in OA and the KKT conditions like in GBD. The relationship between the lower bounds in minimization MINLP problems for each method is as follows

$$Z^k_{LB,GBD} \leq Z^k_{LB,PSC} \leq Z^k_{LB,OA} \tag{5.3}$$

The size of the master problem from OA is larger than the one in the PSC method, although it provides stronger lower bounds. This tradeoff between the methods allows to choose among them depending on the problem, *e.g.*, if problem M^k_{OA} becomes too large to handle, the PSC method can become advantageous at the cost of having to perform more iterations to find the optimal solution of the MINLP problem [243].

5.2.4 Multi-generation cuts

The original OA, GBD, and PSC methods generate a single cut per iteration for the corresponding master problem M^k. Su, Tang, and Grossmann [243] propose adding several cuts at every iteration. This can be achieved using the solution pool of the MILP solver, such as CPLEX[52] while solving the master problem M^k and solving a subproblem NLP($\mathbf{y}^k$) for each feasible solution in the solution pool. Given $|S|$ feasible solutions of the master problem M^k, we solve for each $s \in \{1,\ldots,|S|\}$ a subproblem NLP($\mathbf{y}^k$). The solution to each problem, $(\mathbf{x}^{k,s},\mathbf{y}^{k,s})$, is a valid point for generating linearizations for a convex MINLP.

The OA cuts generated at the kth iteration are as follows,

$$\begin{aligned}
&\eta \geq f(\mathbf{x}^{k,s},\mathbf{y}^{k,s}) + \nabla f(\mathbf{x}^{k,s},\mathbf{y}^{k,s})^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^{k,s} \\ \mathbf{y}-\mathbf{y}^{k,s} \end{bmatrix}, \quad k \in K, s \in S, \\
&\mathbf{g}(\mathbf{x}^{k,s},\mathbf{y}^{k,s}) + \nabla \mathbf{g}(\mathbf{x}^{k,s},\mathbf{y}^{k,s})^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^{k,s} \\ \mathbf{y}-\mathbf{y}^{k,s} \end{bmatrix} \leq \mathbf{0}, \quad k \in K, s \in S.
\end{aligned} \tag{5.4}$$

In a similar fashion, multiple cuts can be generated for the PSC method as follows,

$$\begin{aligned}
&\eta \geq f(\mathbf{v}^{k,s},\mathbf{w},\mathbf{y}) + \begin{bmatrix} \lambda^{k,s} \\ -\mu^{k,s} \end{bmatrix}^\top \begin{bmatrix} \mathbf{g}(\mathbf{v}^{k,s},\mathbf{w},\mathbf{y}) & 0 \\ 0 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{v}-\mathbf{v}^{k,s} \end{bmatrix}, \quad k \in K_F, s \in S, \\
&\begin{bmatrix} \lambda^{k,s} \\ -\mu^{k,s} \end{bmatrix}^\top \begin{bmatrix} \mathbf{g}(\mathbf{v}^{k,s},\mathbf{w},\mathbf{y}) & 0 \\ 0 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{v}-\mathbf{v}^{k,s} \end{bmatrix} \leq \mathbf{0}, \quad k \in K_I, s \in S.
\end{aligned} \tag{5.5}$$

The multi-generation cuts (MC) procedure strengthens the approximation of the master problem, and therefore the lower bound in minimization MINLP problems can be improved. This approach can reduce the number of iterations of decomposition algorithms such as OA and PSC. Even though the iterations can be more time-consuming, the total computational time can decrease using this procedure [243].

5.3 Motivation

Quadratic approximations have been proposed for MINLP decomposition methods and B&B methods. **grossmann2002review** and Fletcher and Leyffer [25] discussed the quadratic master problem of decomposition methods. Based on the Hessian of the Lagrange function with respect to continuous and discrete variables, the master problem is constructed with a quadratic objective function and linear constraints, which results in a Mixed-Integer Quadratic Programming (MIQP) master problem.

Recently, MILP solvers like CPLEX and Gurobi have expanded their capabilities to solve MIQP and Mixed-Integer Quadratically Constrained Programming (MIQCP) problems [244]. The crucial factor of MIQCP solvability is the Hessian matrix. Convex MIQCP problems, which have a positive semi-definite (PSD) Hessian matrix, can be solved to optimality as one special convex MINLP with traditionally MILP solvers, *e.g.*, CPLEX.

Buchheim and Trieu [245] presented a quadratic Outer-approximation scheme for convex integer nonlinear programming problems, which are problems with an unconstrained convex nonlinear objective function with only integer variables. Leyffer [114] integrated the Sequential Quadratic Programming (SQP) algorithm in the branch and bound method.

In the MINLP decomposition methods with quadratic approximation cuts, the master problem is a MIQCP. If the objective function and constraints in MIQCP are convex, the MIQCP optimization problem is convex. If the MIQCP can be transformed into Mixed Integer Second Order Cone Programming (MISOCP), it can be efficiently solved using Second Order Cone Programming (SOCP) solution techniques [92]. Alternatively, Berthold, Heinz, and Vigerske [246] presented a global solution framework based on constraint integer programming used as a basis for the SCIP solver, which is aimed at the solution of MIQCP problems with general Hessian matrix [180].

If the quadratic cuts based on Taylor's second-order expansion term are introduced into

the decomposition of MINLP methods, the objective function can be better approximated than the general linear master problem. However, the quadratic Taylor expansion does not provide a rigorous lower bound for general nonlinear functions. Hence, the OA and PSC with an MIQCP master problem are not guaranteed to converge to the optimal solution of the MINLP. In this chapter, we show that the quadratic approach for the OA and PSC method can be made rigorous by scaling the quadratic terms in scaled quadratic cuts, and in that way, help to accelerate the solution process.

We replace the tangent cuts of the master problem with scaled quadratic cuts that underestimate the convex nonlinear functions. These scaled quadratic cuts allow us to use second-order derivative information and still ensure that we underestimate the nonlinear functions with an approximation at most as nonlinear as the second-order expansion. In combination with the strategy of multi-generation cut, we present the strategy of multi-generation quadratic cuts and hybrid cuts for OA and PSC for solving convex MINLP problems. Numerical experiments demonstrate the performance of the improved OA and PSC method.

5.4 Quadratic approximation cuts

In this section, we show the Taylor series truncated at the second-order and a scalar α can be used to build quadratic cuts (QCUT) that are tight valid underestimators for the nonlinear functions.

5.4.1 Quadratic approximation of nonlinear functions

According to Taylor's theorem, any function $f : \mathbb{R}^n \to \mathbb{R}$ of the class C^{n+1} on the open convex set $\mathcal{B}$ containing the point $\mathbf{x}_0 \in \mathbb{R}^n$ can be written as an expansion around $\mathbf{x}_0$ using a

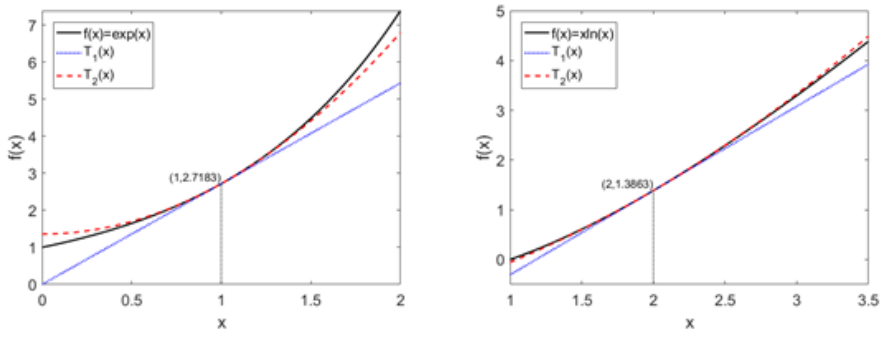
polynomial whose coefficients depend in the derivatives of the function in that point [247, Section 2.7]. The first-order expansion can be defined using the gradient of the function at the expansion point $\nabla f(\mathbf{x}_0)$ as follows,

$$T_1(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0]. \tag{5.6}$$

When we expand a nonlinear function with second-order terms of the Taylor series, we obtain the quadratic approximation of the function,

$$T_2(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0] + \frac{1}{2}[\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0], \tag{5.7}$$

where $\nabla^2 f(\mathbf{x}_0)$ denotes denotes the Hessian matrix of the function f evaluated at the point $\mathbf{x}_0$.



(a) $f(x) = e^x, 0 \leq x \leq 2, x_0 = 1$ (b) $f(x) = x \ln x, 1 \leq x \leq 3.5, x_0 = 2$

Figure 5.1: Comparisons of the first- and second-order Taylor series expansions of convex one dimensional functions

The deviation between the nonlinear function and the second-order expansion is smaller than the deviation between the nonlinear function and its first-order expansion. We can then state the following proposition, which can be trivially proved.

Proposition 5.4.1. If a convex nonlinear function is expanded in one fixed point, the second-order Taylor expansion of the function must be greater than or equal to the first-order Taylor expansion at any point.

From the geometric point of view, the curve of second-order Taylor expansion must lie over the tangent line of the first-order expansion and provides a better approximation. However, the second-order expansion is not always a valid underestimation of a nonlinear function, as is the case of the first-order expansion, as shown in Figure 5.1.

5.4.2 Scaled quadratic approximation

In order to globally underestimate a convex nonlinear function by a quadratic cut, we expand the second-order expansion with a scaled coefficient $\alpha \in \mathbb{R}_+$ such that

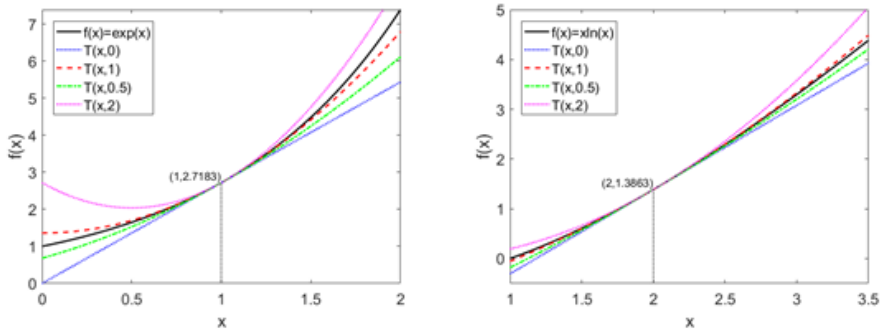
$$T(\mathbf{x}, \alpha) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0] + \frac{\alpha}{2} [\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0) [\mathbf{x} - \mathbf{x}_0]. \tag{5.8}$$

$T(\mathbf{x}, \alpha)$ with $0 \leq \alpha \leq 1$ corresponds to an approximation that lies between the supporting hyperplane and the quadratic expansion, which still is a quadratic approximation with a larger radius of curvature at $\mathbf{x}_0$. On the other hand, for the range of $\alpha \geq 1$, $T(\mathbf{x}, \alpha)$ is an overestimation of the second-order Taylor expansion.

For the convex functions considered in Figure 5.1, we add scaled quadratic cuts as shown in Figure 5.2. We can notice how the scaled second-order expansion lies between the first- and second-order expansions when α is between 0 and 1, while when $\alpha \geq 1$, it lies above the second-order approximation.

Therefore, if the coefficient α satisfies certain conditions, it can yield quadratic approximations that underestimate the convex nonlinear functions. The following propositions provide theoretical guarantees for valid underestimation of scaled quadratic cut, assuming that the variables are bounded.

Proposition 5.4.2. Let $f : \mathbb{R}^n \to \mathbb{R}$ define a general convex twice continuously differentiable nonlinear function $f(\mathbf{x})$, in terms of the n-dimensional variable $\mathbf{x}$ bounded as $\mathbf{x} \in [\mathbf{x}^l, \mathbf{x}^u]$, where $\mathbf{x}^l$, $\mathbf{x}^u$ are finite bounds, and α be scaled coefficient for a quadratic approximation as



(a) $f(x) = e^x, 0 \le x \le 2, x_0 = 1$ (b) $f(x) = x \ln x, 1 \le x \le 3.5, x_0 = 2$

Figure 5.2: Comparisons of the scaled second-order Taylor series expansions with different values of α of convex one dimensional functions

in Eq. (5.8). If we calculate the scalar α in Eq. (5.8) as follows,

$$\frac{\alpha}{2} = \min_{\mathbf{x}\in\mathcal{F}} \frac{f(\mathbf{x}) - f(\mathbf{x}_0) - \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0]}{[\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0]}, \tag{5.9}$$

where $\mathcal{F} = \{\mathbf{x} : \mathbf{x}^l \le \mathbf{x} \le \mathbf{x}^u\} \setminus \{\mathbf{x}_0\}$ and that no element of the Hessian $\nabla^2 f(\mathbf{x}_0)_{ij}$ is equal to zero, then the scaled second-order approximation $T(\mathbf{x}, \alpha)$ is a valid underestimator of the function $f(\mathbf{x})$.

Proof. The necessary condition to prove in this proposition is the following.

$$f(\mathbf{x}) \ge T(\mathbf{x}, \alpha), \forall \mathbf{x} \in \mathcal{F} = \{\mathbf{x} : \mathbf{x}^l \le \mathbf{x} \le \mathbf{x}^u\}. \tag{5.10}$$

If we define the function $g(\mathbf{x})$ as the difference between the original function and the scaled approximation, we have the following condition to prove.

$$g(\mathbf{x}) = f(\mathbf{x}) - T(\mathbf{x}, \alpha) \ge \mathbf{0}, \forall \mathbf{x} \in \mathcal{F} = \{\mathbf{x} : \mathbf{x}^l \ge \mathbf{x} \le \mathbf{x}^u\}. \tag{5.11}$$

Substituting $T(\mathbf{x}, \alpha)$ as in Eq.(5.8) we obtain the following condition:

$$f(\mathbf{x}) - f(\mathbf{x}_0) - \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0] - \frac{\alpha}{2} [\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0] \ge \mathbf{0}. \tag{5.12}$$

Given the fact that $f(\mathbf{0})$ is convex, we can rearrange (5.12) as follows:

$$\frac{\alpha}{2} \leq \frac{f(\mathbf{x}) - f(\mathbf{x}_0) - \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0]}{[\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0]}, \forall \mathbf{x} \neq \mathbf{x}_0, \nabla^2 f(\mathbf{x}_0)_{ij} \neq 0, i, j \in 1, \ldots, n. \tag{5.13}$$

Since the term in the denominator is greater or equal than zero it does not change the inequality sign and we can exclude the expansion point from the domain of the inequality.

Then, minimizing the obtained expression over the selected domain we obtain

$$\min_{\substack{\mathbf{x} \in \mathcal{F} = \{\mathbf{x}: \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u\} \setminus \{\mathbf{x}_0\} \\ \nabla^2 f(\mathbf{x}_0)_{ij} \neq 0,\ i,j \in 1,\ldots,n}} \left\{ \frac{f(\mathbf{x}) - f(\mathbf{x}_0) - \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0]}{[\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0]} \right\} \leq \frac{f(\mathbf{x}) - f(\mathbf{x}_0) - \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0]}{[\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0]} \tag{5.14}$$

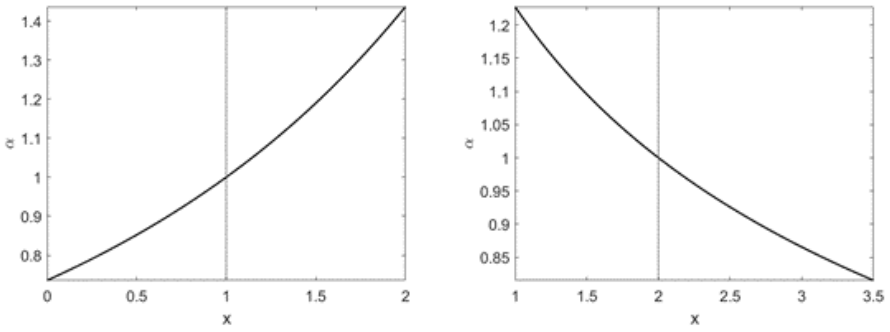
which yields the expression in Eq. (5.9). □

Given the fact that the function $f(\mathbf{x})$ is convex, the term $[\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0]$ is greater or equal than zero, and therefore the function $g(\mathbf{x})$ is monotonic decreasing with respect to α. This means that minimizing the value of the scalar α maximizes the difference between $f(\mathbf{x})$ and $T(\mathbf{x}, \alpha)$.

Having a convex $f(\mathbf{x})$ does not mean that the function defining α is convex itself. Therefore, finding its minimum is not trivial. However, the following observation can avoid solving this nonconvex optimization problem for a particular class of convex MINLP problems. In particular, if we assume that the maximum of the difference between these functions $g(\mathbf{x})$ as in Eq. (5.11) lies at one of the vertices, the function defining α is coordinate-wise monotonic, *i.e.*, nondecreasing or nonincreasing in each variable independently. This special case implies that the minimum α can be found using the following equation,

$$\frac{\alpha}{2} = \min_{\mathbf{x}^v \in \mathcal{V}} \frac{f(\mathbf{x}^v) - f(\mathbf{x}_0) - \nabla f(\mathbf{x}_0)^\top (\mathbf{x}^v - \mathbf{x}_0)}{(\mathbf{x}^v - \mathbf{x}_0)^\top \nabla^2 f(\mathbf{x}_0)(\mathbf{x}^v - \mathbf{x}_0)}, \forall \mathbf{x}^v \neq \mathbf{x}_0, \nabla^2 f(\mathbf{x}_0)_{ij} \neq 0,\ i, j \in 1, \ldots, n, \tag{5.15}$$

where $\mathbf{x}^v$ denotes the vertices in set $\mathcal{V} = (x_1^l, x_1^u) \times \cdots \times (x_n^l, x_n^u)$. The minimum for the expression can then be found by enumeration instead of solving the nonconvex optimization problem in Eq. (5.9) as seen in Figure 5.3.



(a) $f(x) = e^x, 0 \le x \le 2, x_0 = 1$ (b) $f(x) = x \ln x, 1 \le x \le 3.5, x_0 = 2$

Figure 5.3: Function calculating the scalar α for convex one dimensional functions

Although this condition of coordinate-wise monotonicity in α is stronger than the condition of function $f(\mathbf{x})$ being convex, the problems used in this work satisfy that the minimum α is indeed at a vertex, see Appendix 5.A for more details.

In summary, based on Proposition 5.4.2 we generate the quadratic underestimations of convex nonlinear functions for MINLP problems to accelerate the convergence of the OA and PSC algorithms.

5.4.3 Scaled quadratic cuts for Outer-approximation

We generate the scaled quadratic cuts in the master problem of OA as follows.

$$\begin{aligned}
&\eta \ge f(\mathbf{x}^k,\mathbf{y}^k) + \nabla f(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix}\mathbf{x}-\mathbf{x}^k\\ \mathbf{y}-\mathbf{y}^k\end{bmatrix} + \frac{\alpha^k}{2}\begin{bmatrix}\mathbf{x}-\mathbf{x}^k\\ \mathbf{y}-\mathbf{y}^k\end{bmatrix}^\top \nabla^2 f(\mathbf{x}^k,\mathbf{y}^k)\begin{bmatrix}\mathbf{x}-\mathbf{x}^k\\ \mathbf{y}-\mathbf{y}^k\end{bmatrix}, \quad k \in K,\\
&\mathbf{g}(\mathbf{x}^k,\mathbf{y}^k) + \nabla \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix}\mathbf{x}-\mathbf{x}^k\\ \mathbf{y}-\mathbf{y}^k\end{bmatrix} + \frac{\beta^k}{2}\begin{bmatrix}\mathbf{x}-\mathbf{x}^k\\ \mathbf{y}-\mathbf{y}^k\end{bmatrix}^\top \nabla^2 \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k)\begin{bmatrix}\mathbf{x}-\mathbf{x}^k\\ \mathbf{y}-\mathbf{y}^k\end{bmatrix}, \le \mathbf{0}, \quad k \in K,
\end{aligned} \tag{5.16}$$

where α^k is the scale coefficient for the objective function, and β^k is a the scale vector for the constraint functions vector at the kth iteration of OA. The scalar α^k and vector β^k are

calculated based on Eq. (5.9).

The master problem then becomes the following MIQCP problem:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\eta} \quad & Z^k_{LB,OA-QCUT} = \eta \\
\text{s.t.} \quad & \eta \geq f(\mathbf{x}^k,\mathbf{y}^k) + \nabla f(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix} + \frac{\alpha^k}{2} \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix}^\top \nabla^2 f(\mathbf{x}^k,\mathbf{y}^k) \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix}, \quad k \in K, \\
& \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k) + \nabla \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix} + \frac{\beta^k}{2} \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix}^\top \nabla^2 \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k) \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix}, \leq \mathbf{0}, \quad k \in K, \\
& \mathbf{A}\mathbf{x} + \mathbf{E}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \mathbf{y} \in \mathcal{Y} \subseteq \mathbb{Z}^p, \eta \in \mathbb{R}.
\end{aligned}
\tag{$M^k_{OA-QCUT}$}
$$

To illustrate our approach we tackle the solution of the following example problem,

$$
\begin{aligned}
\min_{x_1,x_2} \quad & f(x_1,x_2) = x_1^4 + 3, \\
\text{s.t.} \quad & g_1(x_1,x_2) = 20 - x_1^2 - x_2^2 \geq 0, \\
& g_2(x_1,x_2) = 5 + (x_1-1)^2 - x_2 \leq 0, \\
& g_3(x_1,x_2) = 15\sqrt{x_1} - x_2 \leq 0, \\
& 0 \leq x_1 \leq 3, 0 \leq x_2 \leq 30, \\
& x_1 \in \mathbb{R}, x_2 \in \mathbb{Z}.
\end{aligned}
\tag{Ex}
$$

The master problems for OA are derived at the point $\mathbf{x}_0 = (1, 15)$. Figure 5.4 provides the geometric illustration of the nonlinear objective function and feasible region with scaled quadratic cuts.

Problem M_{OA}^{k} for this example becomes:

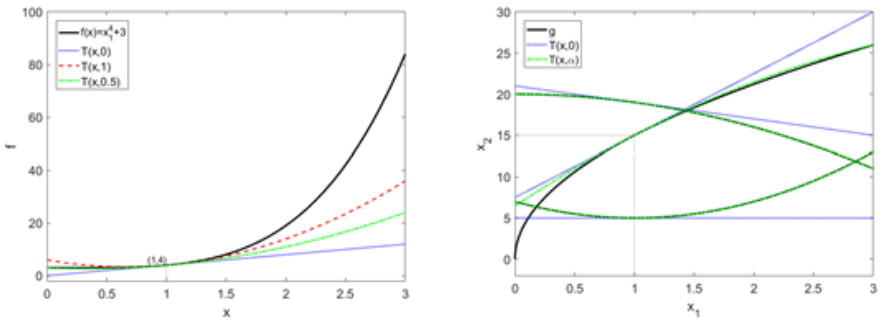
$$\begin{aligned}
\min_{x_1,x_2,\eta} \quad & \eta \\
\text{s.t.} \quad & \eta \geq T_f(x_1,x_2,0) = 4x_1, \\
& T_{g_1}(x_1,x_2,0) = 21 - 2x_1 - x_2 \geq 0, \\
& T_{g_2}(x_1,x_2,0) = 5 - x_2 \leq 0, \\
& T_{g_3}(x_1,x_2,0) = 15/2(x_1+1) - x_2 \leq 0, \\
& 0 \leq x_1 \leq 3, 0 \leq x_2 \leq 30, \\
& x_1 \in \mathbb{R}, x_2 \in \mathbb{Z}, \eta \in \mathbb{R}.
\end{aligned} \tag{5.17}$$

In Figure 5.4(a) we can see that the scaled quadratic cut with $\alpha = 0.5$ is an underestimation of the nonlinear function. Figure 5.4(b) is the illustration of the scaled quadratic cuts for each of the nonlinear constraints. The scaled quadratic problem is as follows,

$$\begin{aligned}
\min_{x_1,x_2,\eta} \quad & \eta \\
\text{s.t.} \quad & \eta \geq T_f(x_1,x_2,0.5) = 4x_1 + 0.5\frac{12}{2}(x_1-1)^2, \\
& T_{g_1}(x_1,x_2,1) = 21 - 2x_1 - x_2 - (x_1-1)^2 \geq 0, \\
& T_{g_2}(x_1,x_2,1) = 5 - x_2 + 2(x_1-1)^2 \leq 0, \\
& T_{g_3}(x_1,x_2,0.536) = 15/2(x_1+1) - x_2 + 0.536\frac{15}{8}(x_1-2)^2 \leq 0, \\
& 0 \leq x_1 \leq 3, 0 \leq x_2 \leq 30, \\
& x_1 \in \mathbb{R}, x_2 \in \mathbb{Z}, \eta \in \mathbb{R}.
\end{aligned} \tag{5.18}$$

Note that for every nonlinear constraint, the scalar α is calculated via Eq. (5.15). Therefore a different value is obtained for each constraint. For the quadratic constraints g_1 and g_2, the value is one, and for the third constraint, it is 0.536. The scaled quadratic cuts underestimate the objective function and overestimate the convex feasible region. Based on Propositions 5.4.1, the scaled second-order expansion of a general convex nonlinear

function is closer to the nonlinear function than its first-order Taylor expansion for a fixed expanded point, while Proposition 5.4.2 guarantees the validity of the relaxation. Therefore, the lower bound of the master problem with scaled quadratic cuts is tighter than one with linear cuts.



(a) Objective function (b) Constraints

Figure 5.4: Geometrical representation of scaled quadratic cuts of the master problem of OA for problem Ex

With scaled quadratic cuts, the master problem of OA becomes a MIQCP with linear objective function and quadratic constraints.

Proposition 5.4.3. The convex MINLP problem MINLP has the same optimal solution $(\mathbf{x}^*, \mathbf{y}^*)$ as the following problem

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\eta} \quad & Z^k_{LB,OA-QCUT} = \eta \\
\text{s.t.} \quad & \eta \geq f(\mathbf{x}^k,\mathbf{y}^k) + \nabla f(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix} + \frac{\alpha^k}{2} \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix}^\top \nabla^2 f(\mathbf{x}^k,\mathbf{y}^k) \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix}, \quad k \in K^*, \\
& \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k) + \nabla \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix} + \frac{\beta^k}{2} \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix}^\top \nabla^2 \mathbf{g}(\mathbf{x}^k,\mathbf{y}^k) \begin{bmatrix} \mathbf{x}-\mathbf{x}^k \\ \mathbf{y}-\mathbf{y}^k \end{bmatrix}, \leq \mathbf{0}, \quad k \in K^*, \\
& \mathbf{A}\mathbf{x} + \mathbf{E}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n, \mathbf{y} \in \mathcal{Y} \subseteq \mathbb{Z}^p, \eta \in \mathbb{R},
\end{aligned}
\tag{MIQCP}
$$

where K^* is the index set of optimal solutions to the problems NLP($\mathbf{y}^k$) for all $\mathbf{y}^k$ which are feasible for the original problem MINLP. All the previous optimal solutions to NLP($\mathbf{y}^k$) are considered and its corresponding cuts are accumulated.

Proof. Since the generated scaled cuts do not cut off the feasible solutions of the convex nonlinear discrete problem, from Proposition 5.4.2, using the convergence theorem for OA [25, 30] this Proposition can be trivially proved. □

Since the Hessian matrices of functions f and $\mathbf{g}$ are positive semi-definite in all points $(\mathbf{x}^k, \mathbf{y}^k)$, the master problem MIQCP is convex.

In the implementation of the algorithm, we solve a relaxation of problem MIQCP, problem $M^k_{OA-QCUT}$ with $k \in K \subseteq K^*$. The solution of problem $M^k_{OA-QCUT}$ corresponds to a lower bound of the optimal solution of MINLP.

Since the scaled quadratic cuts in $M^k_{OA-QCUT}$ are accumulated as the OA main iterations proceed. The master problem can be solved to obtain a nondecreasing sequence of lower bounds. Therefore, we develop an expansion of OA with scaled quadratic cuts, OA-QCUT. Note that if the original MINLP problem MINLP has quadratic objective and/or constraints, problem $M^k_{OA-QCUT}$ will represent exactly the original problem in the first iteration. Therefore it will be solved by solving once the master problem.

The OA-QCUT method is also a decomposition method for convex MINLP problems, whose difference with the general OA is that the master problem is a MIQCP.

5.5 Improved MINLP methods with quadratic cuts

In Su, Tang, and Grossmann [243], effective computational strategies for MINLP methods are presented, including multi-generation cuts and hybrid cuts for OA, GBD, and PSC. Several improved MINLP methods are presented below for convex MINLP problems based

on integrating the scaled quadratic cuts with the strategies of multi-generation cuts and PSC.

5.5.1 Multi-generation quadratic cuts for OA (OA-MQCUT)

The OA method with scaled quadratic cuts inherits the computational difficulties of the master problem in the original OA method regarding the accumulation of generated cuts, which can be even more challenging given that the problem now is an MIQCP. The strategy of multi-generation cuts for OA aims to decrease the computational effort of the MILP master problem by merging solution of multiple MILP problems. Therefore, we try to integrate multi-generation cuts (MCUT) with scaled quadratic cuts for OA, generating multiple quadratic cuts by parallel solutions of NLP subproblems in one main iteration. Suppose the number of multi-generation cuts is $|S|$. The equations of multi-generation scaled quadratic cuts are as follows,

$$\begin{aligned}
&\eta \geq f(\mathbf{x}^{k,s},\mathbf{y}^{k,s}) + \nabla f(\mathbf{x}^{k,s},\mathbf{y}^{k,s})^\top \begin{bmatrix}\mathbf{x}-\mathbf{x}^{k,s}\\ \mathbf{y}-\mathbf{y}^{k,s}\end{bmatrix} + \frac{\alpha^k}{2}\begin{bmatrix}\mathbf{x}-\mathbf{x}^{k,s}\\ \mathbf{y}-\mathbf{y}^{k,s}\end{bmatrix}^\top \nabla^2 f(\mathbf{x}^{k,s},\mathbf{y}^{k,s})\begin{bmatrix}\mathbf{x}-\mathbf{x}^{k,s}\\ \mathbf{y}-\mathbf{y}^{k,s}\end{bmatrix}, \quad k\in K, s\in S,\\
&\mathbf{g}(\mathbf{x}^{k,s},\mathbf{y}^{k,s}) + \nabla \mathbf{g}(\mathbf{x}^{k,s},\mathbf{y}^{k,s})^\top \begin{bmatrix}\mathbf{x}-\mathbf{x}^{k,s}\\ \mathbf{y}-\mathbf{y}^{k,s}\end{bmatrix} + \frac{\beta^k}{2}\begin{bmatrix}\mathbf{x}-\mathbf{x}^{k,s}\\ \mathbf{y}-\mathbf{y}^{k,s}\end{bmatrix}^\top \nabla^2 \mathbf{g}(\mathbf{x}^{k,s},\mathbf{y}^{k,s})\begin{bmatrix}\mathbf{x}-\mathbf{x}^{k,s}\\ \mathbf{y}-\mathbf{y}^{k,s}\end{bmatrix}, \leq \mathbf{0}, \quad k\in K, s\in S.
\end{aligned} \tag{5.19}$$

Here, we can simplify the calculation of the scaling coefficients. Assume at the kth iteration, we obtain the scaling coefficients vector $(\alpha^{k,s},\beta^{k,s})$. The scaling coefficients of iteration $k+1$ are initialized as,

$$\begin{aligned}
\alpha^{k+1} &= \min_{s\in S}\{\alpha^{k,s}\}\\
\beta^{k+1} &= \min_{s\in S}\{\beta^{k,s}\}.
\end{aligned} \tag{5.20}$$

We can then use Proposition 5.4.2 to compute the scaling coefficients or use this suboptimal value for the coefficients as long as the cuts using them are valid. This leads to the

scaling coefficients vector α^{k+1},β^{k+1} to yield a monotonic decreasing sequence.

The main steps of multi-generation quadratic cuts are described in the following algorithm:

Algorithm 4 Outer-approximation multi-quadratic cut (OA-MQCUT) algorithm.

Set $Z_{UB} = \infty$, $Z_{LB} = -\infty$, $k = 0$, specify $|S|$ ▷ Initialization
Define gap tolerance $\epsilon \geq 0$
Solve rMINLP ▷ Solve continuous relaxation
if rMINLP is infeasible **then**
    Set $Z_{LB} = \infty$ ▷ rMINLP and MINLP are infeasible
else
    Let $(\mathbf{x}^0, \mathbf{y}^0)$ be an optimal solution of rMINLP
    Set $Z_{LB} = f(\mathbf{x}^0, \mathbf{y}^0)$
    Derive scaled quadratic cuts at $(\mathbf{x}^0, \mathbf{y}^0)$ following Eq.5.16 to set up problem $M^k_{OA-QCUT}$.
while $Z_{UB} - Z_{LB} > \epsilon$ **do**
    Solve problem $M^k_{OA-QCUT}$ ▷ Solve master problem
    if $M^k_{OA-QCUT}$ is infeasible **then**
        Set $Z^L = \infty$ ▷ $M^k_{OA-QCUT}$ is infeasible
    else
        Let $(\eta^{k,s}, \mathbf{x}^{k,s}, \mathbf{y}^{k,s})$ be the solutions of $M^k_{OA-QCUT}$
        Set $Z_{LB} = \min_{s \in S} \eta^{k,s}$
        Set $()\mathbf{x}^k, \mathbf{y}^k) = \mathbf{x}^{k,s'}, \mathbf{y}^{k,s'}$ with $s' = \arg\min_{s \in S} \eta^{k,s}$
        Solve NLP($\mathbf{y}^k$) for every $\mathbf{y}^{k,s}$ ▷ Solve nonlinear subproblems
        if NLP($\mathbf{y}^k$) is infeasible **then**
            Solve F($\mathbf{y}^k$)
        else
            Let $\mathbf{x}^{k,s}$ be an optimal solution of NLP($\mathbf{y}^k$) with $\mathbf{y}^{k,s}$
            Set $Z_{UB} = \min(Z_{UB}, f(\mathbf{x}^{k,s}, \mathbf{y}^{k,s})$
        Update scaling coefficients $\alpha^{k,s}$, $\beta^{k,s}$ according to Eq. (5.9) and Eq. (5.20)
    Set $k = k + 1$
$(\mathbf{x}^k, \mathbf{y}^k)$ is an optimal solution of problem MINLP, if $Z_{UB} < \infty$, otherwise problem MINLP is infeasible

5.5.2 Hybrid linear and quadratic cuts for OA (OA-HCUT) with multi-generation strategy (OA-MHCUT)

Considering the computational complexity of MIQCP, we develop hybrid linear and scaled quadratic cuts (HCUT) for OA based on a multi-generation strategy. We separate the OA solution process into two stages by iteration number. The main steps are as follows.

(Stage 1) In the first stage, multi-generation linear supporting cuts are constructed and accumulated into the MILP master problem, which is MCUT-OA [243] until the specified limit of iterations is reached.

(Stage 2) In the second stage, multi-generation scaled quadratic cuts are constructed and accumulated in the master problem, which changes the master problem from MILP into MIQCP. This is OA-MQCUT proposed in Section 5.5.1. The solution process terminates when the termination conditions of the OA method are satisfied.

The specified iteration number of the first stage is problem-dependent, which is influenced by the computational complexities of the generated MILP and MIQCP. Suppose the generated MIQCP requires more solution time than the generated MILP. In that case, we can set larger values for the specified iteration number in the first stage. A single first linear approximation can provide a good point for generating strong quadratic cuts; therefore, in this approach, we set the value of iterations in the first stage as 1.

We can extend the hybrid cuts strategy, like generating multi-generation scaled linear cuts in the first stage, generating multi-generation quadratic cuts in the second stage. The cuts generated at the first iterations, since they are distant from the optimal solution, are dominated by those generated in the following iterations. To avoid generating quadratic cuts that are inefficient, we start with linear approximations for the first iterations, and we use the quadratic cuts as we get closer to the optimal solution. Moreover, we can divide the solution process into multiple stages and generate different kinds of cuts in any given stage.

5.5.3 Partial surrogate quadratic and multi-generation cuts

Partial surrogate cuts are one type of cuts between OA cuts and GBD cuts, which are derived from projecting on the nonlinear variables and applying KKT conditions [90, 243]. Since the PSC includes continuous nonlinear variables, we can extend it to scaled quadratic cuts as follows,

$$\eta \geq f(\mathbf{v}^k,\mathbf{w}^k,\mathbf{y}^k)+\begin{bmatrix}\lambda^k\\ -\mu^k\end{bmatrix}^\top\begin{bmatrix}\mathbf{g}(\mathbf{v}^k,\mathbf{w}^k,\mathbf{y}^k) & 0\\ 0 & \mathbf{A}_1\end{bmatrix}\begin{bmatrix}1\\ \mathbf{v}-\mathbf{v}^k\end{bmatrix}+\frac{\alpha^k}{2}\begin{bmatrix}1\\ \mathbf{v}-\mathbf{v}^k\end{bmatrix}^\top\nabla^2 L(\mathbf{v}^k)\begin{bmatrix}1\\ \mathbf{v}-\mathbf{v}^k\end{bmatrix},\ k\in K, \quad (5.21)$$

where L is the projection of the Lagrange function in the nonlinear continuous variable space $\mathbf{v}$, *i.e.*,

$$\begin{aligned}L(\mathbf{v}) &= \text{proj}_{\mathbf{v}}\mathcal{L}(\mathbf{v},\mathbf{w},\mathbf{y})\\ &= \text{proj}_{\mathbf{v}}\left(f(\mathbf{v},\mathbf{w},\mathbf{y})+\lambda^\top\mathbf{g}(\mathbf{v},\mathbf{w},\mathbf{y})+\mu^\top(\mathbf{A}_1\mathbf{v}+\mathbf{A}_2\mathbf{w}+\mathbf{E}\mathbf{y}-\mathbf{b})\right)\end{aligned} \quad (5.22)$$

Eq. (5.21) can be derived from the procedure of PSC cuts with the second-order expansion of nonlinear functions [90].

The scalar are computed in a equivalent manner as in Eq. (5.15), as follows,

$$\begin{aligned}&\frac{\alpha}{2}=\min_{(\mathbf{v}^v,\mathbf{w}^v,\mathbf{y}^v)\in\mathcal{V}}\frac{f(\mathbf{v}^v,\mathbf{w}^v,\mathbf{y}^v)-f(\mathbf{v}^k,\mathbf{w}^k,\mathbf{y}^k)-\lambda^k\mathbf{g}(\mathbf{v}^k,\mathbf{w}^k,\mathbf{y}^k)+\mu^k\mathbf{A}_1(\mathbf{v}^v-\mathbf{v}^k)}{(\mathbf{v}^v-\mathbf{v}^k)^\top\nabla^2 L(\mathbf{v}^k)(\mathbf{v}^v-\mathbf{v}^k)},\\ &\mathbf{v}^v\neq\mathbf{v}^k,\nabla^2 L(\mathbf{v}^k)_{ij}\neq 0,\ i,j\in 1,\ldots,|\mathbf{v}|,\end{aligned} \quad (5.23)$$

Eqs. (5.21) and (5.23) define the Partial Surrogate Quadratic Cuts (PSQC), which are an underestimation for convex nonlinear functions based on Proposition 5.4.2.

Furthermore, if the calculated scalar α based on Eq. (5.23) is greater than 1, we set it to 1. This since we want to avoid overestimating the second-order approximation.

We develop the PSC-QCUT method based on the PSC method using scaled quadratic cuts in Eq. (5.21). Compared to the OA cuts, PSC cuts are not as tight for the relaxation of nonlinear constraints [243]. The advantage of PSC is generating fewer cuts than OA for an

iteration. These advantages generally are inherited from PSC to PSC-QCUT, which may reduce the computational time of solving the master problem MIQCP.

We integrate the multi-generation strategy with PSQC, leading to the Partial Surrogate Multi-generation Quadratic Cuts (PSC-MQCUT) generating multiple cuts in a single iteration as follows,

$$\begin{aligned}\eta \geq f(\mathbf{v}^{k,s},\mathbf{w}^{k,s},\mathbf{y}^{k,s}) + \begin{bmatrix}\lambda^{k,s}\\ -\mu^{k,s}\end{bmatrix}^{\top}\begin{bmatrix}\mathbf{g}(\mathbf{v}^{k,s},\mathbf{w}^{k,s},\mathbf{y}^{k,s}) & 0\\ 0 & \mathbf{A}_1\end{bmatrix}\begin{bmatrix}1\\ \mathbf{v}-\mathbf{v}^{k,s}\end{bmatrix}\\ + \frac{\alpha^{k,s}}{2}\begin{bmatrix}1\\ \mathbf{v}-\mathbf{v}^{k,s}\end{bmatrix}^{\top}\nabla^2 L(\mathbf{v}^{k,s})\begin{bmatrix}1\\ \mathbf{v}-\mathbf{v}^{k,s}\end{bmatrix},\ k\in K, s\in S,\end{aligned} \tag{5.24}$$

5.5.4 Proposed MINLP solution methods

In summary, we develop six improved MINLP methods, OA-QCUT, OA-MQCUT, OA-HCUT, OA-MHCUT, PSC-QCUT, and PSC-MQCUT, integrating the proposed scaled quadratic cuts with multi-generation of linear and quadratic cuts for OA and PSC.

5.6 Numerical Experiments

We solve 44 convex MINLP benchmark problems with the proposed methods of OA-QCUT, OA-MQCUT, OA-HCUT, OA-MHCUT, PSC-QCUT, PSC-MQCUT. These problems vary in size and number of nonlinear functions. Furthermore, the computational performance is compared with OA and PSC and with the MINLP solvers DICOPT, BONMIN, SBB, and SCIP.

We implement the benchmark MINLP test problems in GAMS 24.8.2 [248]. The operating system is Windows 7, with an Intel Core 2 Duo processor, a CPU of 3.4 GHz, and 16 GB of RAM. The solutions of the NLP problems are obtained using the solvers CONOPT

3.17C [154] and IPOPT 3.12 [158], and the solutions of the MILP/MIQCP problems are obtained using CPLEX 12.7.0.0 [52]. Three stopping criteria for the proposed methods and solvers are used. The first one is if the objective of the last MIP master problem is greater than the best NLP solution found, namely a "crossover" occurred due to the use of integer cuts, see Duran and Grossmann [30] and Viswanathan and Grossmann [124]; the second one is the maximum number of iterations; the third one is the maximum solution CPU time. The major iteration limit is 500, and the maximum CPU solution time is 3600 seconds. The CPU solution time only includes the time used to solve the master problem and the subproblems. It does not include the time calculating the values of the scale parameters α and β. These parameters were calculated using Eq. (5.15), which resulted in the same values as solving the problem stated in Eq. (5.9). None of the methods required considerable time compared to the overall MILP and NLP solution times.

The MINLP test problems are classified into three groups: 7 simple problems, 15 problems with special structure, and 22 medium/large scale problems [60].

5.6.1 Simple MINLP problems

The seven simple MINLP problems are tested taken from the MINLPLib problem library⋆. Five problems were partly also used by Duran and Grossmann [30]. In the problems `Synthes*`, the nonlinear terms are exponential or logarithmic functions of only continuous variables, which are separable from the discrete variables. Problem `Gkocis` has a nonlinear objective function, and the problem `Alan` has only one nonlinear constraint. The problem `Ex1223b` has discrete variables involved in the nonlinear functions, while the problem `St_e14` is an equivalent transformation of the `Ex1223b` problem. Here, we expand the nonlinear terms of `Ex1223b` with discrete variables to first-order and all nonlinear terms to

⋆http://www.gamsworld.org/minlp/minlplib.htm accessed in January 2017

Table 5.1: Computational results of DICOPT, OA, OA-QCUT, and OA-MQCUT for small MINLP problems

	DICOPT		OA*		OA-QCUT		OA-MQCUT		
Problems	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	α_{min}
`Synthes1`	4	0.8	5	2.0	3	1.5	3	1.3	0.41
`Synthes2`	7	0.5	3	0.9	3	0.8	3	1.1	0.46
`Synthes3`	11	0.6	7	3.0	3	1.2	2	0.8	0.58
`Gkocis`	3	0.4	3	1.2	2	0.7	2	0.8	1.0
`Alan`	5	0.4	5	2.6	3	1.1	2	0.9	1.0
`Ex1223b`	7	0.4	4	4.5	3	1.0	3	1.0	1.0
`St_e14`	6	0.5	6	5.3	2	0.8	2	0.7	0.99

* This denotes an implementation of OA directly in GAMS

second-order using the Taylor series expansion.

The number of variables and constraints for the seven problems are shown in Table 8 of Appendix 2 in [249]. The computational results of DICOPT and implementations of OA, OA-QCUT, and OA-MQCUT are listed in Table 5.1. The initial scalar vector α is computed based on Eq.(5.15), and updated by Eq. (5.20). We find that the scalars α are near 0.5 for the first three problems and almost 1 for the last four problems. All solution methods were able to obtain the optimal solutions for all seven simple problems. For the simpler problems, fewer iterations and shorter CPU times were required by OA-QCUT and OA-MQCUT than OA with linear cuts. OA-QCUT requires fewer iterations than DICOPT, although it generally requires slightly longer CPU times.

For the same problem set, the computational results of OA with hybrid cuts (OA-HCUT) and multi-generation hybrid cuts (OA-MHCUT) and GAMS implementations of PSC, PSC-QCUT, PSC-MQCUT are shown in Table 5.2. For OA-HCUT, we obtain linear cuts in the first iteration and then quadratic cuts in the following iterations.

Table 5.2: Computational results of OA-HCUT, OA-MHCUT, PSC, PSC-QCUT, PSC-MQCUT for small MINLPs

	OA-HCUT		OA-MHCUT		PSC		PSC-QCUT		PSC-MQCUT	
Problems	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)
`Synthes1`	3	1.0	2	0.9	4	2.2	3	1.7	3	1.3
`Synthes2`	3	1.0	2	0.8	6	2.8	5	3.0	3	1.3
`Synthes3`	3	1.3	4	2.0	12	6.4	5	2.5	2	1.0
`Gkocis`	2	1.1	2	1.0	3	1.3	3	1.6	3	1.7
`Alan`	3	1.0	2	0.8	5	2.7	4	2.2	3	2.1
`Ex1223b`	4	1.4	3	1.5	6	2.8	6	3.2	3	1.5
`St_e14`	2	1.0	2	1.1	6	3.0	4	2.9	3	1.9

Although the scaled quadratic cuts are generated at the second iteration of the OA-HCUT method, OA-HCUT requires either fewer iterations or shorter solution times than OA for six problems in this comparison. OA-MHCUT was even more efficient than OA-HCUT, except for the `Synthes3` and `St_e14` problems.

PSC-QCUT performs better than PSC, with fewer iterations and smaller CPU times on five problems, and same iterations on the other two problems. PSC-MQCUT generally requires fewer iterations and shorter CPU times than PSC-QCUT in the simple MINLP instances, except for the `Gkocis` problem.

5.6.2 MINLP with special structure

The set of Constrained Layout (`CLay*`) and Security Layout (`SLay*`) problems[⋆] is special since all the MINLPs in it have quadratic nonlinear constraints. The chosen instances were the ones that reformulated the disjunctions using the Big-M method [60]. Since the constraints are quadratic, the scaling coefficients (α,β) are set to 1. The computational results for these instances are shown in Table 5.3.

[⋆] `http://egon.cheme.cmu.edu/ibm/page.htm` accessed in January 2017

Since the quadratic cut is equal to the quadratic function in this problem set, OA-QCUT and OA-MQCUT terminate at the first normal main iteration after the initial one. As expected, OA and DICOPT require more iterations to obtain the optimal solutions than OA-QCUT and OA-MQCUT. OA-QCUT performs better than OA-MQCUT on CPU times for all `CLay*` problems, but OA-MQCUT performs better than OA-QCUT on CPU times for 71% problems of `SLay*`. `SLay*` instances are MIQPs, which makes the generated master problem for OA-MQCUT one with a single quadratic constraint, less than the quadratic constraints of the `CLay*` instances. That is the reason that OA-MQCUT performs better on these instances. We also solve these instances directly using CPLEX, which results with smallest CPU times among all methods.

Note that we obtain the OA cuts for the first master problems of PSCs and the following iteration when the NLP subproblem is infeasible. Although the iteration numbers of OA-HCUT and OA-MHCUT are the same as with OA-QCUT, the CPU times of OA-MHCUT are generally longer than OA-HCUT except for the problems `CLay43`, `Slay08`, and `SLay10`. Because of the structure of the `CLay*` and `SLay*` instances, the master problems of OA-MHCUT generates more quadratic cuts than the master problems of OA-HCUT. OA-HCUT requires longer solution time than OA-QCUT for 80% of the problems in reported in Table 5.3.

There are infeasible subproblems for the `CLay*` and `SLay*` instances, which generate the infeasible cuts in the PSC-QCUT method. These cuts are similar to those generated by OA-QCUT. This fact allows the PSC-QCUT and PSC-MQCUT to solve the compared instances in only 3 or 2 iterations. Since PSC-MQCUT generated more scaled quadratic cuts in the master problems than PSC-QCUT, PSC-MQCUT requires longer solution times than PSC-QCUT, except for 3 `SLay*` instances.

The nonlinear constraints or objective function of `CLay*` and `SLay*` instances are all

Table 5.3: Complutational results for Special structure MINLP problems (MIQCP).

	CPLEX	DICOPT		OA		OA-QCUT		OA-MQCUT		OA-HCUT		OA-MHCUT		PSC		PSC-QCUT		PSC-MQCUT	
Problems	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)
CLay42	0.8	10	3.2	10	7.8	2	2.9	2	4.6	3	5.9	3	8.8	8	4.3	3	2.5	3	3.3
CLay43	2.8	12	4.3	9	6.3	2	5.4	2	13.4	3	15.0	3	14.6	9	4.6	3	5.0	3	7.2
CLay44	2.2	20	14.2	17	17.6	2	10.5	2	19.7	3	20.4	3	70.1	13	8.8	3	14.9	3	23.7
CLay45	16.5	25	36.7	19	62.4	2	33.4	2	60.5	3	91.7	3	144.7	17	16.9	3	43.3	3	78.5
CLay52	4.8	10	7.4	12	18.5	2	9.7	2	11.8	3	16.6	3	41.5	8	8.9	3	8.7	3	12.3
CLay53	5.7	11	6.9	7	7.6	2	11.0	2	33.5	3	20.7	3	134.1	8	9.1	3	8.6	3	27.9
CLay54	17.4	21	101.4	19	216.5	2	50.2	2	56.5	3	115.5	3	308.6	12	79.0	3	57.2	3	135.2
CLay55	58.0	40	498.3	33	601.8	2	115.2	2	207.6	3	225.8	3	421.2	28	640.9	3	169.9	3	348.8
SLay04	0.4	9	0.8	10	4.8	2	0.7	2	0.9	3	1.8	3	3.1	3	2.0	2	0.9	2	1.1
SLay05	0.2	26	1.7	11	3.7	2	1.1	2	1.2	3	2.1	3	2.9	3	3.0	2	1.5	2	1.9
SLay06	0.3	50	8.4	20	19.3	2	4.9	2	2.5	3	2.4	3	3.3	3	11.2	2	1.9	2	1.7
SLay07	0.3	500	-	24	17.1	2	5.5	2	1.5	3	2.2	3	2.3	3	102.0	3	3.1	2	3.0
SLay08	0.4	500	-	25	30.2	2	9.8	2	4.2	3	12.0	3	9.0	3	1003.1	2	2.7	2	3.8
SLay09	0.4	79	94.4	39	448.7	3	22.1	2	17.3	3	5.6	3	9.2	3	1019.2	2	2.9	2	44.1
SLay10	1.7	500	-	51	2433.8	3	8.7	2	5.6	3	1005.5	3	534.9	3	1006.6	2	16.6	2	4.8

quadratic, which would mean that a quadratic approximation would be exact. The nonlinear constraints in the `CLay*` instances ensured that the feasible region between two variables was a circle. One specific case was a circle of radius 6 and centered on the point $(x_1, x_2) = (15, 10)$. A constraint of the form,

$$6^2 \leq (x_1 - 15)^2 + (x_2 - 10)^2, \tag{5.25}$$

would be exactly approximable using a secon-order Taylor series expansion around any point of the domain.

To illustrate the effect of scaling, we present the previous constraint as two separate constraints with respecto to x_2 and with squared-root terms,

$$\begin{aligned} g_1(x_1, x_2) &: x_2 \leq 10 + \sqrt{(x_1 - 15)^2 - 6^2}, \\ g_2(x_1, x_2) &: x_2 \geq 10 - \sqrt{(x_1 - 15)^2 - 6^2}, \end{aligned} \tag{5.26}$$

Assuming that we have an expansion point at $x_0 = 11$ and $x_0 = 20$, the linear supporting planes derived from the first-order approximation will be as follows,

$$\begin{aligned} T_{g_1,1}(x_1, x_2, 0) &: x_2 \leq 10 + \sqrt{20} + \frac{\sqrt{20}(x_1 - 11)}{5}, \\ T_{g_2,1}(x_1, x_2, 0) &: x_2 \geq 10 - \sqrt{20} - \frac{\sqrt{20}(x_1 - 11)}{5}, \\ T_{g_1,2}(x_1, x_2, 0) &: x_2 \leq 10 + \sqrt{11} - \frac{5\sqrt{11}(x_1 - 20)}{11}, \\ T_{g_2,2}(x_1, x_2, 0) &: x_2 \geq 10 - \sqrt{11} + \frac{5\sqrt{11}(x_1 - 20)}{11}, \end{aligned} \tag{5.27}$$

On the other hand, after determining the minimum for every constraint such that every

cut is a valid underestimator we obtain the following quadratic cuts.

$$
\begin{aligned}
T_{g_1,1}(x_1,x_2,0.556) &: x_2 \le 10 + \sqrt{20} + \frac{\sqrt{20}(x_1-11)}{5} - \frac{0.556}{2}\frac{9\sqrt{20}(x_1-1)^2}{100},\\
T_{g_2,1}(x_1,x_2,0.556) &: x_2 \ge 10 - \sqrt{20} - \frac{\sqrt{20}(x_1-11)}{5} + \frac{0.556}{2}\frac{9\sqrt{20}(x_1-1)^2}{100},\\
T_{g_1,2}(x_1,x_2,0.394) &: x_2 \le 10 + \sqrt{11} - \frac{5\sqrt{11}(x_1-20)}{11} - \frac{0.394}{2}\frac{36\sqrt{11}(x_1-1)^2}{121},\\
T_{g_2,2}(x_1,x_2,0.394) &: x_2 \ge 10 - \sqrt{11} + \frac{5\sqrt{11}(x_1-20)}{11} + \frac{0.394}{2}\frac{36\sqrt{11}(x_1-1)^2}{121},
\end{aligned}
\tag{5.28}
$$

We show in Figure 5.5 the difference between the scaled quadratic cuts and the linear cuts for this problem.

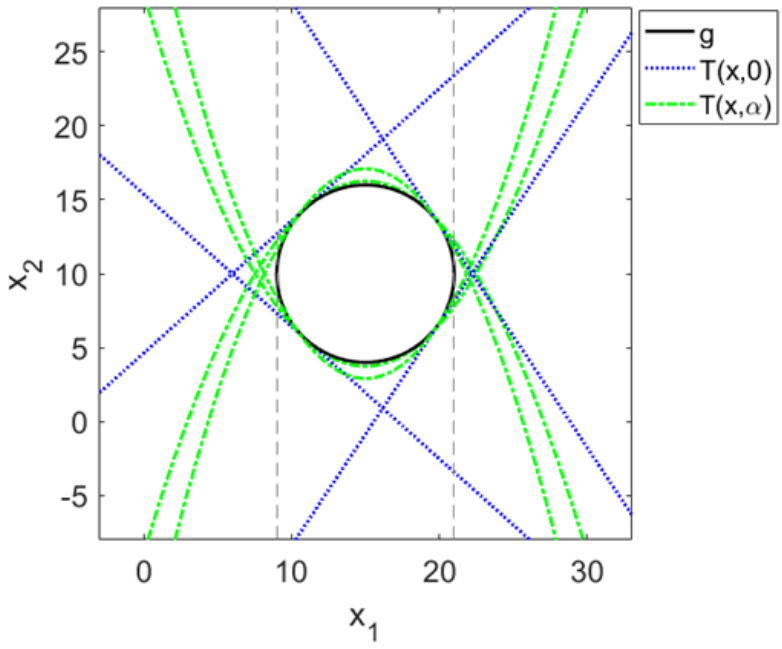


Figure 5.5: Illustration of feasible region of a circular constraints with linear cuts and scaled quadratic cuts

The quadratic approximation of the feasible region in this example is tighter compared with the linear approximation as seen in Figure 5.5. This observation gives intuition around why OA-QCUT performs better than the original OA in the `CLay*` and `SLay*` instances.

5.6.3 Medium scale MINLP problems

22 MINLP problems with medium scale are tested in this subsection, obtained from [60]. Aiming at general convex MINLP problems with all kinds of nonlinear functions, by definition if convex the Hessian matrices expanding the nonlinear functions of MINLP satisfy Positive Semi-Definiteness (PSD). Based on Eq. (5.9), the value of α is computed for each problem in order to guarantee the optimal solutions. For the `Batch*` instances, we constructed the first master problems of PSC, PSC-QCUT, and PSC-MQCUT using PSC and PSC-QCUT cuts, respectively. For other problems in this subsection, we used the OA cuts for the first master problems of PSC, PSC-QCUT, and PSC-MQCUT. The initial PSC cuts of the `Batch*` problems are valid in the hybrid solution method, but the initial PSC cuts of other problems in this subsection lead to many infeasible NLP subproblems. This behavior can be explained based on the `Batch*` models, which are mostly linear except for a single constraint and the objective function, conditions exploited by the PSC cuts. The computational results of all approaches considered in this work for these instances appear in Table 5.4

Note that OA-QCUT and OA-MQCUT require only two to four iterations. However, the CPU solution times are not shorter than DICOPT for `Batch` problems since these involve mostly linear constraints. This trend is reversed for the Csched problems except for Csched1a. DICOPT failed to solve two of the Farm Layout (`FLay*`), and OA fails on one `FLay*` instance, but OA-QCUT and OA-MQCUT solved all `FLay*` instances. In all the instances except from `Batch08`, where the value of the α_{min} is equal to 0.85, α_{min} is equal to 1. This means that the nonlinear functions were underestimated by the second-order expansions for this set of test problems.

We know that the general PSC cuts are not tighter than the general OA cuts. The scaled quadratic PSC cuts are also looser than the scaled quadratic OA cut. When α of OA-QCUT

Table 5.4: Complutational results for Medium Scale MINLP problems.

	DICOPT		OA		OA-QCUT		OA-MQCUT		OA-HCUT		OA-MHCUT		PSC		PSC-QCUT		PSC-MQCUT	
Problems	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)	Iter.	CPU(s)
Batch03	2	0.4	2	0.6	2	0.6	2	0.7	2	0.8	2	0.8	14	5.4	9	6.7	8	8.2
Batch06	4	0.3	5	1.7	3	1.3	3	2.2	3	7.2	2	1.3	280	264.7	47	287.8	18	63.8
Batch08	5	0.5	5	2.0	4	1.4	4	2.7	4	1.4	4	2.4	2	0.6	4	5.4	2	1.4
Batch10	10	4.7	12	6.3	2	7.7	2	11.2	3	10.6	2	6.1	4	1.4	3	8.2	3	6.7
Batch12	5	4.2	6	9.3	2	13.3	2	17.7	3	20.5	4	36.8	4	2.0	3	27.6	3	15.2
Batch15	6	6.5	7	12.3	3	42.2	3	46.6	3	26.7	2	18.8	3	1.2	3	1.5	3	1.8
Batch20	4	6.2	10	30.4	2	33.8	2	76.1	3	42.4	3	56.4	3	1.3	3	1.6	3	2.0
Csched1a	2	0.3	4	1.2	2	0.6	2	0.7	3	1.0	2	0.8	4	1.2	2	0.7	2	0.8
Csched1	20	1.2	8	2.3	2	1.1	2	0.8	5	3.2	2	0.8	8	2.3	2	0.7	2	0.8
Csched2a	90	16.5	55	34.1	3	1.8	3	2.1	4	1.9	3	3.2	100	95.2	3	1.8	3	2.6
Csched2	160	146.5	117	254.3	4	2.7	3	3.1	4	2.0	3	3.9	117	217.4	3	1.9	3	2.2
FLay02	4	0.4	3	1.0	3	0.9	2	0.6	3	1.1	3	1.4	500	-	500	-	500	-
FLay03	34	1.3	9	3.7	2	1.0	2	1.0	3	1.4	3	2.2	500	-	500	-	500	-
FLay04	300	100.9	18	15.8	3	3.0	3	4.2	3	2.6	3	3.0	500	-	500	-	500	-
FLay05	500	-	25	306.0	2	4.2	2	4.6	4	5.8	2	3.5	500	-	500	-	500	-
FLay06	-	3600	-	3600	2	34.7	2	68.1	3	19.3	3	27.8	500	-	500	-	500	-
Proc_21a	25	4.0	8	4.3	4	9.2	3	15.0	5	12.4	4	13.0	7	3.5	4	5.6	3	4.1
Proc_21b	25	3.6	6	3.0	4	4.2	3	5.0	4	4.9	3	5.8	10	5.4	4	3.6	3	3.8
Proc_31a	30	4.6	5	2.2	3	2.6	2	2.9	3	2.4	2	1.6	4	2.0	4	2.4	3	1.9
Proc_31b	28	4.5	6	4.1	2	2.3	2	2.9	3	3.3	3	4.8	40	21.9	3	3.4	3	3.8
Proc_36a	32	8.3	6	4.2	3	5.4	3	9.5	3	5.8	3	9.1	139	108.9	3	5.9	3	9.1
Proc_36b	35	8.7	6	5.1	3	5.7	3	8.6	3	5.0	3	7.5	500	-	5	4.1	4	4.7

is equal to 1 for the problems in Table 5.4, it should also be 1 for PSC-QCUT. For this problem set, OA-HCUT performs better on iterations than OA except for two problems with the same iterations but worse than OA-QCUT for 50% of the instances. The number of iterations for OA-MHCUT is smaller than that of OA-HCUT for 50% instances, but longer solution times are required for 64% of the problems. The PSC with quadratic cuts is better on iterations than PSC with linear cuts for 59% of the problems, especially for the csched2a and csched2 instances. PSC-MQCUT needs a longer CPU time than PSC-QCUT for 50% instances. One issue noticed was that the PSC cuts for the `FLay` instances are very weak, which lead the three PSC methods to fail after surpassing the given maximum iteration limit.

We can observe the behavior of the proposed approaches in the performance profile in Figure [**fig:qcoa.performance˙proposed**]. One can notice that improved solution techniques, *e.g.*, scaled quadratic cuts and multi-generation of cuts, yields a benefit vs. the traditional OA and PSC methods. The best performant proposed alternative for the evaluated instances is OA-QCUT, followed closely by OA-HCUT and OA-MQCUT. These methods, together with OA, were able to solve all the problems in the testes set of instances in less than one hour. In general, we observe that the PSC approaches dominate across the instances solvable in less than 10 seconds, mainly since these instances are primarily linear that satisfy the problem structure that motivates the PSC methods with a portion of continuous variables interacting linearly in the constraints. This difference reverses for larger and more challenging instances, where OA can solve more instances than PSC in general.

5.6.4 Comparison with MINLP solvers

We also solve the 44 test problems with the BONMIN implementation of Outer-approximation, B-OA 1.8 [169], SBB [250] and SCIP 3.2 [180], which are OA, Branch

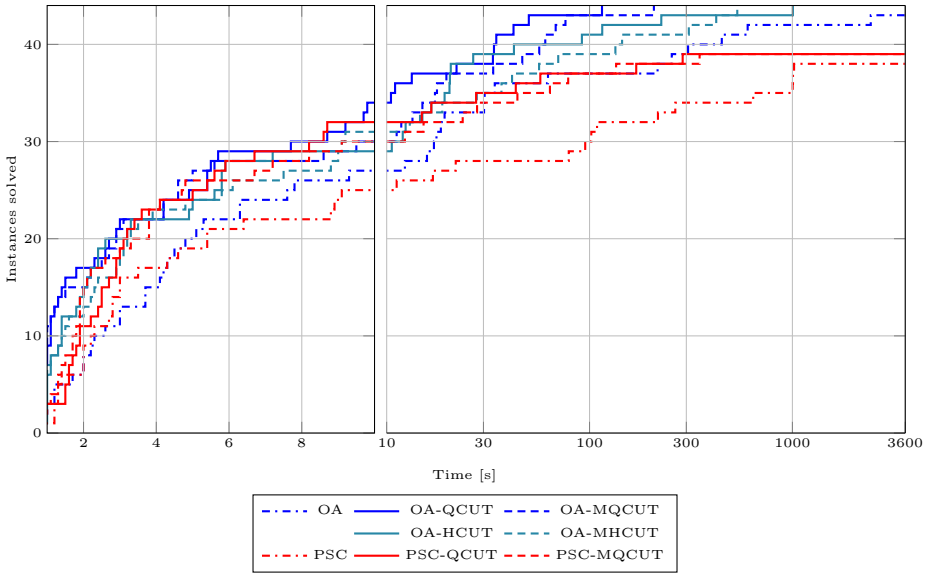


Figure 5.6: Absolute performance profiles for scaled quadratic approaches presented in this chapter.

& Bound or Constraint Programming-based MINLP solvers. This comparison gives an idea of how the proposed methods perform compared to available MINLP solvers. Although CPLEX was only able to solve the MINLP with special structure, we included CPLEX compared to other solvers. The performance profile, including the best and worst performant alternatives of OA and PSC, together with the commercial solvers, is shown in Figure 5.7.

We note that OA-QCUT and OA-MQCUT solve all the tested instances in the time limit, with OA-QCUT being more efficient for the section above 50% of the solved instances than OA-MQCUT, while the rest is comparable. Both OA-QCUT and OA-MQCUT are better than OA according to this performance profile. SCIP continuously outperforms all methods until about 75% of the total problems. The performance profile shows that although the first solved problems are solved more efficiently by DICOPT, SBB, and SCIP, the proposed methods could solve more instances given the time limit. Finally, CPLEX

has outstanding performance for those problems that are MIQCP instances, solving these instances as efficiently as SCIP, but fails to solve all the other instances solving only 15 of the total 44 test problems.

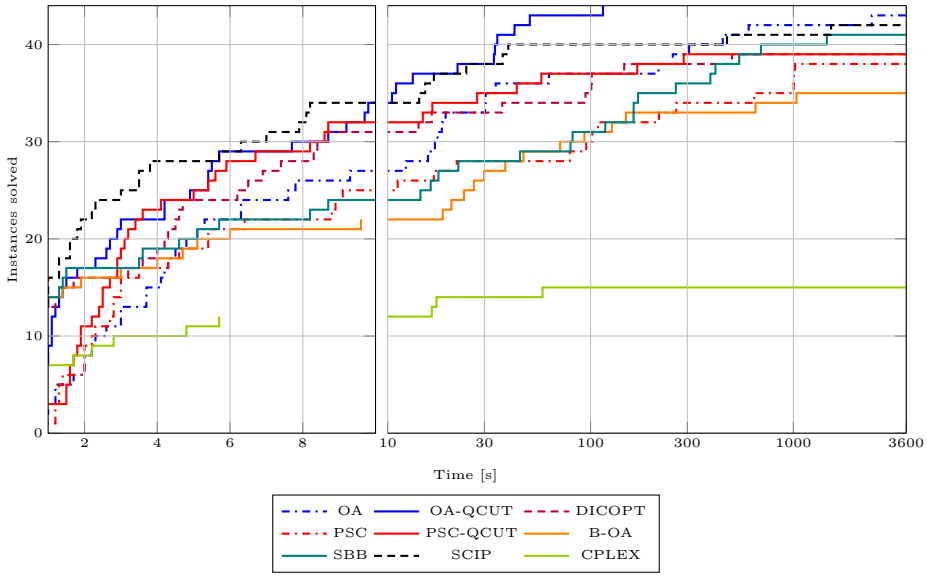


Figure 5.7: Absolute performance profiles for scaled quadratic approaches presented in this chapter compared to MINLP solvers.

DICOPT, SBB, and SCIP outperform OA-HCUT for the first 40% of the problems solved, and SCIP remains the most efficient solver until about 75% of the instances. OA-HCUT performs better than all other compared methods for the last 25% of the instances.

Comparing the performance of OA with the hybrid methods(OA-HCUT and OA-MHCUT) against the quadratic OA (OA-QCUT and OA-MQCUT), we notice that the hybrid approach was slightly less efficient than the quadratic only approach. This result can be supported by the fact that an essential part of the tested instances has quadratic objective and constraints, which were exactly approximated by the quadratic cuts. The idea behind the hybrid approach is that a quadratic approximation is not required for the initial iterations. Therefore, it would be enough to use the linear approximation instead, making

the master problem easier to solve.

The performances of PSC-QCUT and PSC-MQCUT are similar; occasionally, PSC-MQCUT outperforms PSC-QCUT, and both are better than the general PSC method.

5.7 Conclusions

This chapter has proposed scaled quadratic cuts for MINLP decomposition methods, namely OA and PSC. Based on the theoretical analysis, we derived the conditions that ensure that the scaled quadratic cuts can strictly underestimate a convex function. We designed the scaled quadratic cuts based on these observations for OA and PSC methods, which can obtain the optimal solutions for the convex MINLP problems. Integrating the strategies of scaled quadratic cuts, multi-generation cuts, and linear cuts, we propose six improved MINLP methods: OA-QCUT, OA-MQCUT, OA-HCUT, OA-MHCUT, PSC-QCUT, and PSC-MQCUT. Numerical experiments show that OA with scaled quadratic cuts and multi-generation cuts can solve all the tested MINLP benchmark problems. Besides, they require fewer iterations and shorter CPU solution times than OA, especially for MINLP problems with quadratic and highly nonlinear constraints, where the linear approximation is poor. PSC-QCUT and PSC-MQCUT similarly require fewer iterations and shorter CPU solution times than PSC.

After comparing the performance of each one of the six proposed MINLP methods, we can conclude that integrating second-order scaled underestimators to the nonlinear constraints can improve the performance of the two decomposition algorithms, OA and PSC. The OA method appeared to outperform the PSC method in the tested instances, showing that at least for these instances, the tradeoff between smaller but more MILP master problems solved is not preferable. The multi-generation cut strategy was beneficial

to improve the performance of each decomposition algorithm. However, it was less efficient in the compared tested instances than adding a single quadratic constraint [243]. Using the hybrid linear-quadratic OA method (OA-HCUT) was less efficient than using quadratic cuts from the beginning, which resulted in fewer iterations and a smaller difference between solving MILP and MIQCP problems for the tested instances. The most efficient proposed MINLP method was the OA-QCUT, which, compared to other implementations of the OA method such as DICOPT and BONMIN, shows that the quadratic cuts can improve the overall efficiency of the method. Future work will aim to solve the general nonconvex MINLP problems using OA and PSC with scaled quadratic cuts.

5.A On vertex and non-vertex solutions for α

As mentioned in Section 5.4.2, in order to obtain a valid underestimator for a convex function using the scaled second-order Taylor series expansion the scalar α has to be calculated using Eq. (5.9) or solving the following equivalent NLP problem,

$$\begin{aligned} \min_{\mathbf{x}} \alpha &= 2\frac{f(\mathbf{x}) - f(\mathbf{x}_0) - \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0]}{[\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0]} \\ \text{s.t. } \mathbf{x} &\in \mathcal{F} = \{\mathbf{x} : \mathbf{x} \leq \mathbf{x}^l \leq \mathbf{x}^u\} \setminus \{\mathbf{x}_0\}, \end{aligned} \tag{5.29}$$

at the expansion point $\mathbf{x}_0$ such that no element of the Hessian $\nabla^2 f(\mathbf{x}_0)_{ij}$ is equal to zero.

In this chapter, instead of solving the previous NLP problem to global optimality, we used Eq. (5.15) which assumes the minimum of the function to lie at the vertices. The stronger condition over the function defining α is component-wise monotonicity. The condition of the component-wise monotonicity can be ensured if none of the components of the gradient change their sign in the feasible set,

$$\begin{aligned} &\nabla\left(\frac{f(\mathbf{x}) - f(\mathbf{x}_0) - \nabla f(\mathbf{x}_0)^\top [\mathbf{x} - \mathbf{x}_0]}{[\mathbf{x} - \mathbf{x}_0]^\top \nabla^2 f(\mathbf{x}_0)[\mathbf{x} - \mathbf{x}_0]}\right) \text{ same sign} \\ &\text{s.t. } \mathbf{x} \in \mathcal{F} = \{\mathbf{x} : \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u\} \setminus \{\mathbf{x}_0\}. \end{aligned} \tag{5.30}$$

Let the scalar multiplying the quadratic term in the second-order expansion of the convex function $f(\mathbf{x})$ at the expansion point $\mathbf{x}_0$ be defined as in Eq. (5.9) or as in Eq. (5.29). In order to satisfy this condition, we propose the following proposition.

Proposition 5.A.1. The value of the scalar α in the definition of the scaled quadratic cut can be calculated using Eq. (5.15) and guarantee that $T(\mathbf{x},\alpha)$ is an underestimator of function $f(\mathbf{x})$ if the function satisfies the following condition,

$$\begin{aligned} 2\nabla^2 f(\mathbf{x}_0)[\mathbf{x}-\mathbf{x}_0]\Big(f(\mathbf{x})-f(\mathbf{x}_0)-\nabla f(\mathbf{x}_0)^\top[\mathbf{x}-\mathbf{x}_0]\Big) \neq \\ \Big([\mathbf{x}-\mathbf{x}_0]^\top\nabla^2 f(\mathbf{x}_0)[\mathbf{x}-\mathbf{x}_0]\Big)[\nabla f(\mathbf{x})-\nabla f(\mathbf{x}_0)], \end{aligned} \tag{5.31}$$

for all $\mathbf{x}\in\mathcal{F}=\{\mathbf{x}\in\mathbb{R}^n : \mathbf{x}^l\leq\mathbf{x}\leq\mathbf{x}^u\}\setminus\{\mathbf{x}_0\}$, where $\mathbf{x}^l$ and $\mathbf{x}^u$ are the lower and upper bounds of $\mathbf{x}$, respectively.

Proof. The function defining α is the quotient of two convex functions, which is not necessarily convex. This means that its minimum will not necessarily lie at one of the vertices of the hyper-box defined by the bounds of $\mathbf{x}$. However, if the function is component-wise monotonic, since we are minimizing the function α, we can assure that its minimum will lie at one of the vertices. A component-wise monotonic function is a function that is nonincreasing or nondecreasing in each variable independently.

Calculating the gradient of the previous function we obtain the following expression,

$$\begin{aligned} &\frac{2\nabla^2 f(\mathbf{x}_0)[\mathbf{x}-\mathbf{x}_0]\left(f(\mathbf{x})-f(\mathbf{x}_0)-\nabla f(\mathbf{x}_0)^\top[\mathbf{x}-\mathbf{x}_0]\right)}{([\mathbf{x}-\mathbf{x}_0]^\top\nabla^2 f(\mathbf{x}_0)[\mathbf{x}-\mathbf{x}_0])^2}- \\ &\frac{\Big([\mathbf{x}-\mathbf{x}_0]^\top\nabla^2 f(\mathbf{x}_0)[\mathbf{x}-\mathbf{x}_0]\Big)[\nabla f(\mathbf{x})-\nabla f(\mathbf{x}_0)]}{([\mathbf{x}-\mathbf{x}_0]^\top\nabla^2 f(\mathbf{x}_0)[\mathbf{x}-\mathbf{x}_0])^2}, \text{ same sign} \\ &\text{s.t. } \mathbf{x}\in\mathcal{F}=\{\mathbf{x} : \mathbf{x}^l\leq\mathbf{x}\leq\mathbf{x}^u\}\setminus\{\mathbf{x}_0\}. \end{aligned} \tag{5.32}$$

Since the function $f(\mathbf{x})$ is a convex function, the term $[\mathbf{x}-\mathbf{x}_0]^\top\nabla^2 f(\mathbf{x}_0)[\mathbf{x}-\mathbf{x}_0])$ is non-negative, therefore the denominator will be non-negative which avoids sign changes of all the components of the gradient. Since the function $f(\mathbf{x})$ is twice differentiable, then both

the function and its gradient are continuous, which implies that the two terms under the assumption of component-wise monotonicity can never be equal. This is the condition given in Eq. (5.31), proving the given proposition. □

The condition of component-wise monotonicity of the function defining α is stronger than requiring the function $f(\mathbf{x})$ to be convex, and it is strongly dependent of the expansion point $\mathbf{x}_0$. Although the nonlinear functions involved in the test instances did satisfy this condition, *e.g.*, quadratic constraints satisfy it trivially, it is not the general case.

Two counterexamples are given where the minimum of the does not lie in one of the vertices, which results in an approximation which does not underestimate the function in the whole domain. The first example has the following function in $f : \mathbb{R} \to \mathbb{R}$

$$f(x) = b - \sqrt{(r^2 - (x-a)^2)}, \tag{5.33}$$

where a, b, and r are scalar coefficients and the function $f(x)$ describes a semicircle off radoius r and center (a,b). The given function is defined if the variables x has bounds $x \in [a-r, a+r]$. Given an expansion point x_0, we obtain the following expression for calculating α,

$$\frac{\alpha}{2} = \min_{x \in \mathcal{F} = \{x : x^l \leq x \leq x^u\} \setminus \{x_0\}} \frac{b - \sqrt{(r^2 - (x-a)^2)} - b - \sqrt{(r^2 - (x_0-a)^2)} - \frac{x-a}{\sqrt{r^2-(x_0-a)^2}}(x-x_0)}{\frac{r^2}{(r^2-(x_0-a)^2)^{3/2}}(x-x_0)^2}. \tag{5.34}$$

Given an example of a semicircle of radius $r = 3$ and center in $(5,5)$, we obtain a plot of the function defining α in Figure 5.8.

Following Eq. (5.15), the value of α should be 1.33 corresponding to the value at the vertex $x = a + r = 8$. In case we set $\alpha = 1$, we would obtain the second-order approximation, but the minimum of the given function is neither in the vertex nor when $\alpha = 1$.

The minimum of the given function can be analytically found and it appears at the point $x^* = 2a - x_0$, corresponding to $\alpha = 0.89$ in this case. Figure 5.9 shows the original function

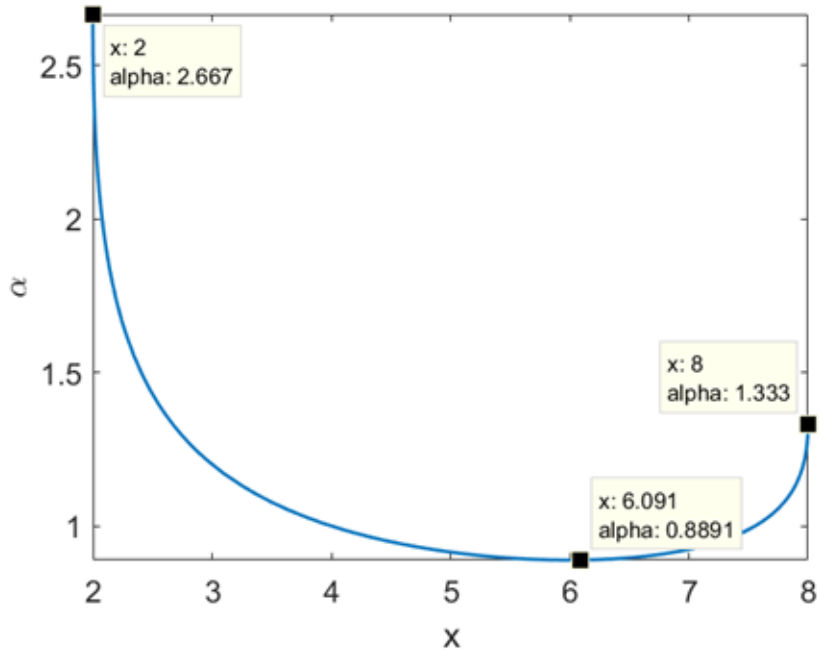


Figure 5.8: Scalar α for function $f(x) = 5 - \sqrt{(9-(x-5)^2)}$ according to Eq. (5.34).

with its first- and second-order Taylor series approximations.

Note that the only Taylor approximations that are valid underestimators in Figure 5.9 are the first-order and the scaled second-order using the minimum *alpha*, as proposed in Proposition 5.4.2. This can be confirmed in Figure 5.10, where the difference between the function and its approximation is shown. In case the approximation is an underestimator, the difference is positive for the entire feasible region.

The second example is the following convex function in $\mathbb{R}^2$,

$$f(x_1, x_2) = x_1^4 + x_2^4, \tag{5.35}$$

with variables douns $x_1 \in [0.3, 5]$, $x_2 \in [0.5, 7]$ and expansion point $(4.9, 4.9)$.

The first-order Taylor series approximation of this function is

$$T_1(x_1, x_2) = -\frac{17294403}{5000} + \frac{117649}{250}(x_1 + x_2). \tag{5.36}$$

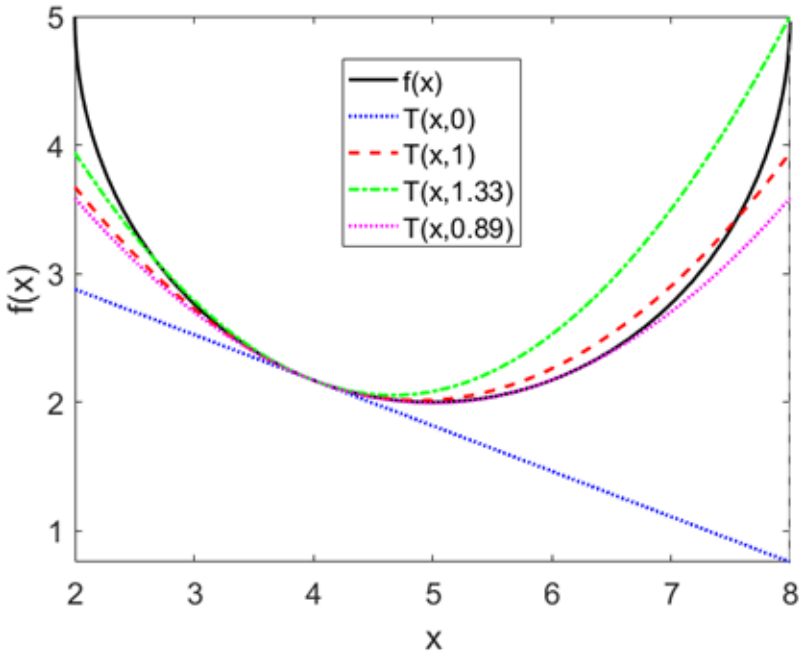


Figure 5.9: $f(x) = 5 - \sqrt{(9-(x-5)^2)}$ and its first-order, second-order, scaled second-order with the scalar α calculated from the minimum at the vertices and the scaled second-order with the minimum scalar α_{min}.

Its scaled second-order Taylor series approximation is defined as,

$$T(x_1,x_2,\alpha) = -\frac{17294403}{5000} + \frac{117649}{250}(x_1+x_2) + \alpha\left(\left(x_1\frac{7203}{50} - \frac{352947}{500}\right)\left(x_1 - \frac{49}{10}\right) + \left(x_2\frac{7203}{50} - \frac{352947}{500}\right)\left(x_2 - \frac{49}{10}\right)\right). \tag{5.37}$$

The function defining the scalar α is as follows,

$$\frac{\alpha}{2} = \min_{\mathbf{x}\in\mathcal{F}=\{\mathbf{x}:\mathbf{x}^l\leq\mathbf{x}\leq\mathbf{x}^u\}\setminus\{\mathbf{x}_0\}} \frac{x_1^4 + x_2^4 + \frac{17294403}{5000} - \frac{117649}{250}(x_1+x_2)}{\left(x_1\frac{7203}{50} - \frac{352947}{500}\right)\left(x_1 - \frac{49}{10}\right) + \left(x_2\frac{7203}{50} - \frac{352947}{500}\right)\left(x_2 - \frac{49}{10}\right)}. \tag{5.38}$$

According to the methodology proposed, α according to Eq. 5.15 should be the minimum among the vertices of the function. In this case, the values for each vertex are presented in Table 5.5.

According to Eq. 5.15, the value for α should be 0.5281, corresponding to vertex $(v_1^v, x_2^v) = (0.3, 0.5)$.

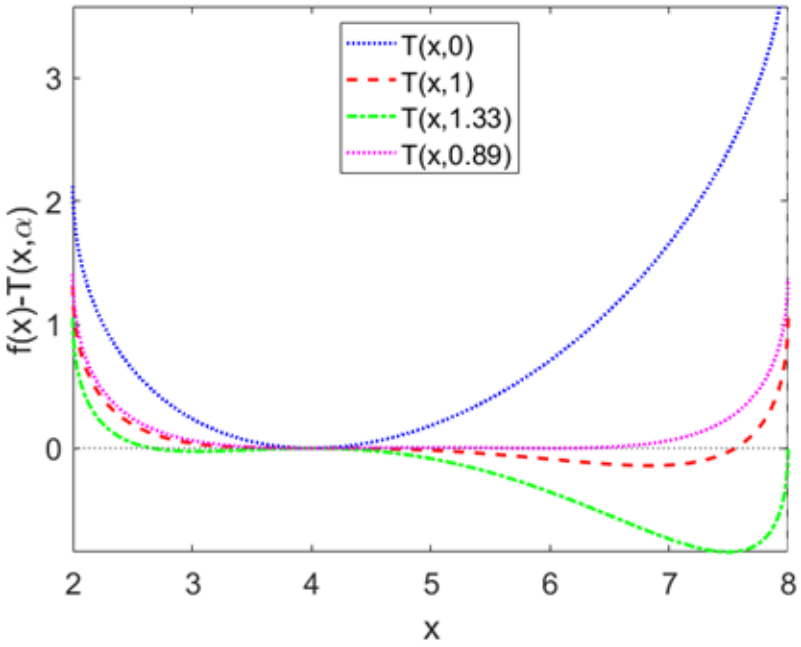
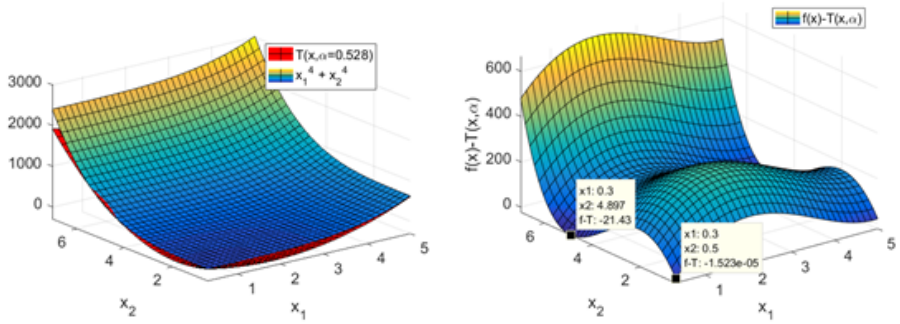


Figure 5.10: Difference between $f(x) = 5 - \sqrt{(9-(x-5)^2)})$ and and the different order Taylor series approximations.



(a) $f(x_1, x_2)$ and $T(x_1, x_2, 0.528)$ (b) $f(x_1, x_2) - T(x_1, x_2, 0.528)$

Figure 5.11: $f(x_1, x_2) = x_1^4 + x_2^4$ and scaled second-order Taylor series approximation with $alpha = 0.528$, $\mathbf{x}_0 = (4.9, 4.9)$

As seen in Figure 5.11, the scaled second order approximation is not a valid underestimator for the function $f(x_1, x_2) = x_1^4 + x_2^4$. Note that at the point $(x_1, x_2) = (0.3, 4.9)$ the difference

Table 5.5: Values of α the scalar in the vertices of the the function $f(x_1, x_2) = x_1^4 + x_2^4$ defined over $[0.3, 5] \times [0.5, 7]$ and expanded at point $(4.9, 4.9)$.

Variable	Vertex 1	Vertex 2	Vertex 3	Vertex 4
x_1	0.3	0.3	5	5
x_2	0.5	7	0.5	7
α	0.5281	0.6582	0.536	1.316

between these two functions is -21.43, violating the condition in Eq. 5.10. This happens since the minimum of the function defining α does not lie in a vertex, as seen in Figure 5.12.

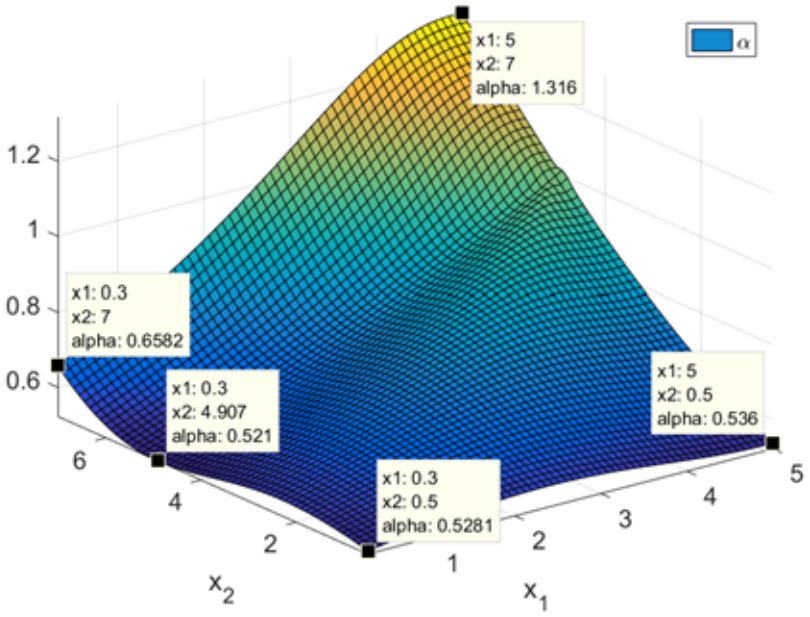
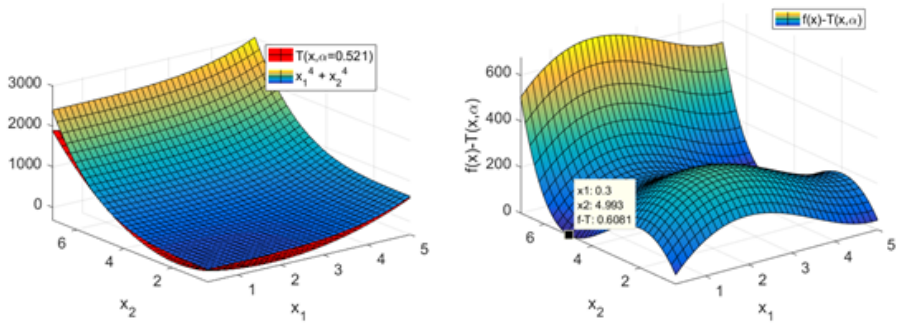


Figure 5.12: Scalar α for function in Eq. 5.35 according to Eq. 5.38.

The function in Figure 5.12 is not coordinate-wise monotonic, so its minimum lies at the point $(0.3, 4.9)$ a combination of the Taylor expansion point and a vertex, where $\alpha = 0.521$.

If we calculate the scaled second-order Taylor series approximation using the minimum value of α we obtain a valid underestimator of the convex function $f(\mathbf{x})$ as Proposition 5.4.2 indicates shown in Figure 5.13.



(a) $f(x_1, x_2)$ and $T(x_1, x_2, 0.521)$ (b) $f(x_1, x_2) - T(x_1, x_2, 0.521)$

Figure 5.13: $f(x_1, x_2) = x_1^4 + x_2^4$ and scaled second-order Taylor series approximation with $alpha = 0.521$, $\mathbf{x}_0 = (4.9, 4.9)$

An interesting observation is that all the minima for the scalar α that d not correspond to vertices, appear at combinations of the expansion point and the vertices. Therefore, we conjecture that given some conditions on the function $f(\mathbf{x})$ we can find its minimum at the set $\mathcal{W} = (x_1^l, x_{01}, x_1^u) \times \cdots \times (x_n^l, x_{0n}, x_n^u)$.

5.B Test problems statistics

The problem statistics for the test problems of the Section 5.6 are listed in Table 8, Table 9 and Table 10 respectively. Note that Eqns and Vars are the abbreviations for number of equations and variables, respectively. DVars represents the number of discrete variables. NZ represent the number of non-zero coefficients. NNZ is the number of nonlinear matrix entries in the problem Jacobian.

Table 5.6: Problem statistics for convex MINLP solved in Chapter 5.

Problems	Eqns	Vars	DVars	NZ	NNZ
`Synthes1`	7	7	3	23	6
`Synthes2`	15	12	5	49	8
`Synthes3`	24	18	8	91	12
`Gkocis`	9	12	3	28	2

Table 5.6: Problem statistics for convex MINLP solved in Chapter 5.

Problems	Eqns	Vars	DVars	NZ	NNZ
Alan	8	9	4	24	3
Ex1223b	10	8	4	32	17
St_e14	14	12	4	40	17
Clay42	93	53	32	319	64
Clay43	109	58	36	375	96
Clay44	125	62	40	431	128
Clay45	141	66	44	487	160
Clay52	138	82	50	483	80
Clay53	158	88	55	553	120
Clay54	178	92	60	623	160
Clay55	198	97	65	693	200
Slay04	55	45	24	189	8
Slay05	91	71	40	311	10
Slay06	136	103	60	463	12
Slay07	190	141	84	645	14
Slay08	253	185	112	857	16
Slay09	325	235	144	1099	18
Slay10	406	291	180	1371	20
Batch03	20	20	9	53	10
Batch06	74	47	24	191	22
Batch08	218	102	60	547	40
Batch10	1020	279	129	2866	49
Batch12	1512	407	203	4256	59
Batch15	1782	446	203	5069	62
Batch20	2328	559	251	6664	67
Csched1a	23	29	15	78	7
Csched1	20	77	63	174	8
Csched2a	138	233	140	622	57
Csched2	138	401	308	958	58
Flay02	12	15	4	39	2
Flay03	25	27	12	87	3
Flay04	43	43	24	155	4
Flay05	66	63	40	243	5
Flay06	94	87	60	351	6
Proc_21a	62	48	21	205	15
Proc_21b	113	69	42	328	15
Proc_31a	102	77	41	371	31
Proc_31b	235	128	82	678	31
Proc_36a	122	92	46	431	36

Table 5.6: Problem statistics for convex MINLP solved in Chapter 5.

Problems	**Eqns**	**Vars**	**DVars**	**NZ**	**NNZ**
`Proc_36b`	217	138	92	661	36

Chapter 6

Use of Regularization and Second-Order Information for Outer-approximation*

6.1 Introduction

Mixed-integer nonlinear programming (MINLP) is a class of optimization problems containing both integer and continuous variables as well as nonlinear functions. The integer variables make it possible to incorporate logic relations and discrete quantities in the mathematical model. Together with linear and nonlinear constraints, MINLP becomes a powerful framework for modeling real-world optimization problems, and thus, there is a vast number of applications in areas such as engineering, computational chemistry, and finance [4, 117]. MINLP problems are by definition non-convex; however, they are still commonly classified as either convex or non-convex. An MINLP problem is considered as convex if an integer relaxation results in a convex nonlinear programming (NLP) problem [251]. Convexity is a desirable property since it enables the direct use of several decomposition techniques for solving the problem. Such decomposition techniques are, *e.g.*, Outer-approximation (OA) [30], extended cutting plane (ECP) [118], extended supporting hyperplane (ESH) [121], generalized Benders decomposition (GBD) [28], and branch and bound (BB) techniques [27].

*Published as: Jan Kronqvist, David E Bernal, and Ignacio E Grossmann. "Using regularization and second order information in outer approximation for convex MINLP". *Mathematical Programming* 180.1 (2020), pp. 285–310.

For reviews of MINLP methods and applications see [5, 104, 129, 251]. Even if there are several methods available for solving convex MINLP problems, it is still a challenging type of optimization problems as shown in the solver benchmark in [121].

Methods such as OA, ECP, ESH, and GBD all generate an iteratively improving linear approximation of the MINLP problem, where the nonlinear functions are underestimated by first-order Taylor series expansions. The linear approximation is a mixed-integer linear programming (MILP) problem and is often referred to as the MILP-master problem. All these methods iteratively choose the integer trial solutions as the minimizer of the MILP-master problem. Choosing the iterative solutions as the minimizer of a linear approximation is similar to the approach used in Kelley's method [119], which is an algorithm intended for convex NLP problems. It is known that Kelley's method is not efficient at handling nonlinearities and it has a poor complexity bound, see, *e.g.,* [252]. Kelley's method is sometimes even referred to as unstable since the iterative solutions tend to make large jumps in the search space [253]. Since methods such as ECP, ESH, GBD, and OA choose the iterative integer solutions in the same manner as Kelley's method, they could also suffer from the same instability. Several techniques to reduce the instability of Kelley's method have successfully been used for NLP problems, *e.g.,* regularization to reduce the step size or the concept of a trust region [254].

Due to the non-convex nature of MINLP problems, it is not trivial to use regularization of the step size or a trust region when solving such problems, since the integer requirements may cause solutions to be far apart in the search space. However, recently there has been interest in the idea of using regularization for solving convex MINLP problems, *e.g.,* using quadratic stabilization with Benders decomposition was proposed in [255] and using regularization combined with a cutting plane method was presented in [256].

Here we present an approach for introducing stabilization in the subproblems for choos-

ing the integer combination in OA. The stabilization technique is inspired by the regularization used in the level method for NLP, see [257, 258], and therefore, the method is referred to as Level-based Outer-approximation (L-OA). By modifying the L-OA method it is possible to include second order information in the subproblems of choosing the integer combination, and we refer to this method as Quadratic Outer-approximation (Q-OA). In Q-OA we use a second order Taylor series expansion for the Lagrangean function as the objective in the subproblems for finding a new integer combination. A similar quadratic approach was presented in [25]. However, the level constraint used in the level method provides a more robust way of enforcing an improvement and avoiding cycling. Furthermore, the level constraint forces the solutions to be chosen as an interpolation between the minimizer of the Lagrangean approximation and the minimizer of the linear approximation in the MILP-master problem. The proposed methods are motivated by the strong convergence properties of the level method compared to Kelley's, and recent advances in software for solving MILP and mixed-integer quadratic programming (MIQP) problems.

The proposed methods are intended to accelerate the convergence of OA by choosing the integer combinations more carefully, using either a regularization technique or second-order information. Due to the regularization and the use of second-order derivatives, the proposed methods should be better suited for handling nonlinearities compared to OA. However, each iteration in L-OA and Q-OA will also be more complex than an iteration in OA. For MINLP problems with only a few nonlinear terms, there might not be significant improvements by the proposed methods. The methods are, thus, mainly intended for problems with moderate to high degree of nonlinearity. We begin with a brief review of OA in Section 6.2, and from there we continue by presenting the basics of L-OA and Q-OA in Sections 6.3 and 6.4. In Section 6.5, it is proven that the convergence properties of OA still hold with the modifications in the proposed methods. Finally, in Section 6.6 we present a

numerical comparison of Q-OA, L-OA, and OA, based on test problems from the problem library MINLib2 [203].

6.2 Background

The MINLP problems considered here can be written as follows,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} \quad & g_j(\mathbf{x},\mathbf{y}) \leq 0 \quad \forall j = 1,\dots l, \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m. \end{aligned} \tag{MINLP}$$

In order to guarantee global convergence, we need to assume some properties of the nonlinear functions. Throughout this chapter we rely on the following assumptions:

Assumption 1. The nonlinear functions $f, g_1, \dots, g_l : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ are convex and continuously differentiable.

Assumption 2. The linear constraints define a nonempty compact set.

Assumption 3. For each feasible integer combination $\mathbf{y}$, an integer combination such that there exist $\mathbf{x}$ variables for which the problem is feasible, a constraint qualification holds, *e.g.*, Slater's condition [226].

These are the typical assumptions needed for rigorously proving convergence of OA, see [25, 30]. OA can be generalized to be applicable to non-differentiable problems, see, *e.g.*, [259], although such problems are not considered here.

We begin by briefly presenting the main steps of the Outer-approximation method. As previously mentioned, the method uses a linear approximation of the MINLP problem to obtain trial solutions for the integer variables. Once an integer combination is obtained, the corresponding continuous variables can be determined by solving a continuous optimiza-

tion problem. The previously obtained trial solutions $\left\{(\mathbf{x^i}, \mathbf{y^i})\right\}_{i=0}^{k}$ are used to construct the linear approximation of the MINLP problem. At iteration k, the next integer combination $\mathbf{y}^{k+1}$ is obtained by solving the following MILP subproblem

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu} \quad & \mu \\
\text{s.t.} \quad & f(\mathbf{x^i}, \mathbf{y^i}) + \nabla f(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq \mu \quad \forall i = 1, \ldots, k, \\
& g_j(\mathbf{x^i}, \mathbf{y^i}) + \nabla g_j(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq 0 \quad \forall i = 1, \ldots k, \forall j \in \mathcal{I}_i, \\
& \mathbf{Ax} + \mathbf{By} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n, \ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R}.
\end{aligned}
\tag{OA-master}
$$

Here $\mathcal{I}_i$ are index sets containing the indexes of the nonlinear constraint active at the trial solution $(\mathbf{x^i}, \mathbf{y^i})$ [25]. Due to convexity, we know that the feasible set is overestimated and that the objective will be underestimated, see, *e.g.*, [30] and Lemma 6.5.1 in Section 6.5. The optimum of problem OA-master, thus, provides a valid lower bound to the MINLP problem, which is referred to as LB^{k+1}. Once the new integer combination $\mathbf{y^{k+1}}$ is obtained, the corresponding $\mathbf{x}$ variables can be obtained by solving the following convex NLP subproblem,

$$
\begin{aligned}
\min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y^{k+1}}) \\
\text{s.t.} \quad & g_j(\mathbf{x}, \mathbf{y^{k+1}}) \leq 0 \quad \forall j = 1, \ldots l, \\
& \mathbf{Ax} + \mathbf{By^{k+1}} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n.
\end{aligned}
\tag{NLP-I}
$$

If problem NLP-I is feasible and solved to optimality, we obtain $\mathbf{x^{k+1}}$ and furthermore, the optimum provides a valid upper bound UB^{k+1} to the MINLP problem. Otherwise, if the NLP problem is infeasible we need a different approach to obtain the $\mathbf{x}$ variables and

this can be done, for example, by solving a feasibility problem. The feasibility problem minimizes the constraint violation with the current choice of $\mathbf{y}$ variables, *e.g.*, using the ℓ_∞ norm, and it can be defined as,

$$\begin{aligned} \min_{\mathbf{x},r} \quad & r \\ \text{s.t.} \quad & g_j(\mathbf{x}, \mathbf{y^{k+1}}) \leq r \quad \forall j = 1, \dots l, \\ & \mathbf{Ax} + \mathbf{By^{k+1}} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n,\ r \in \mathbb{R}_+. \end{aligned} \tag{NLP-f}$$

By solving problem NLP-f the continuous variables $\mathbf{x^{k+1}}$ are obtained. However, in this case, $(\mathbf{x^{k+1}}, \mathbf{y^{k+1}})$ is not a feasible solution, and thus, no upper bound is obtained at this iteration. The feasibility problem always satisfies Slater's condition and due to the convexity assumption, we know that the feasibility problem is always feasible and tractable.

In case the difference between the upper and lower bound is not within the desired tolerance, we improve the linear approximation by adding new linearizations to problem OA-master. These linearizations are often referred to as cutting planes or supporting hyperplanes, and they are given by,

$$\begin{aligned} f(\mathbf{x^{k+1}}, \mathbf{y^{k+1}}) + \nabla f(\mathbf{x^{k+1}}, \mathbf{y^{k+1}})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^{k+1}} \\ \mathbf{y} - \mathbf{y^{k+1}} \end{bmatrix} \leq \mu, \\ g_j(\mathbf{x^{k+1}}, \mathbf{y^{k+1}}) + \nabla g_j(\mathbf{x^{k+1}}, \mathbf{y^{k+1}})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^{k+1}} \\ \mathbf{y} - \mathbf{y^{k+1}} \end{bmatrix} \leq 0 \quad \forall j \in \mathcal{I}_{k+1}. \end{aligned} \tag{6.1}$$

Due to convexity, the cuts will not exclude any feasible solution from the search space [227]. Adding these cuts to the MILP subproblem ensures that the integer combination $\mathbf{y^{k+1}}$ will not be obtained in a consecutive iteration unless it is the optimal integer solution. Convergence can be ensured since each iteration will either result in a new integer combination or verify

optimality. For more details of OA see [25, 30, 124]. The basic steps of OA are summarized as a pseudo-code in Algorithm 5.

Algorithm 5 An algorithm summarizing the basic steps of the Outer-approximation method

Define accepted optimality gap $\epsilon \geq 0$.
1. Initialization.
   - 1.1 Obtain a relaxed solution $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ by solving an integer relaxation of the MINLP problem.
   - 1.2 Generate cuts at $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ according to (6.1) and construct problems OA-master.
   - 1.3 Set iteration counter $k = 1$, $UB^0 = \inf$ and $LB^0 = -\inf$.
2. Repeat until $UB^{k-1} - LB^{k-1} \leq \epsilon$.
   - 2.1 Solve problem OA-master to obtain $\mathbf{y^k}$ and LB^k
   - 2.4 Solve problem NLP-I with integer variables fixed as $\mathbf{y^k}$ to obtain $\mathbf{x^k}$.
     - 2.4.1 If problem NLP-I is infeasible, obtain $\mathbf{x^k}$ by solving feasibility problem NLP-f and set $UB^k = UB^{k-1}$.
   - 2.5 Generate cuts at $\mathbf{x^k}, \mathbf{y^k}$ according to (6.1) and add these to problems OA-master.
   - 2.6 If $\mathbf{x^k}, \mathbf{y^k}$ is feasible, set $UB^k = \min\{f(\mathbf{x^k}, \mathbf{y^k}), UB^{k-1}\}$.
   - 2.7 Increase iteration counter, $k = k + 1$
3. Return the best found solution.

Here we have not considered the integer cuts used in [30], since these are not needed for convex problems. To get a better understanding of OA and to highlight the differences compared to the other methods, consider the following simple example

$$\begin{aligned} \text{minimize} \quad & -6x - y \\ \text{s.t.} \quad & 0.3(x-8)^2 + 0.04(y-6)^4 + 0.1e^{2x}y^{-4} \leq 56 \\ & 1/x + 1/y - x^{0.5}y^{0.5} \leq -4 \\ & 2x - 5y \leq -1 \\ & 1 \leq x \leq 20, \quad 1 \leq y \leq 20, \quad x \in \mathbb{R},\ y \in \mathbb{Z}. \end{aligned} \tag{Ex 1}$$

The basic features of problem Ex 1 are illustrated in Figure 6.2, showing the constraints, objective, and the optimal solution.

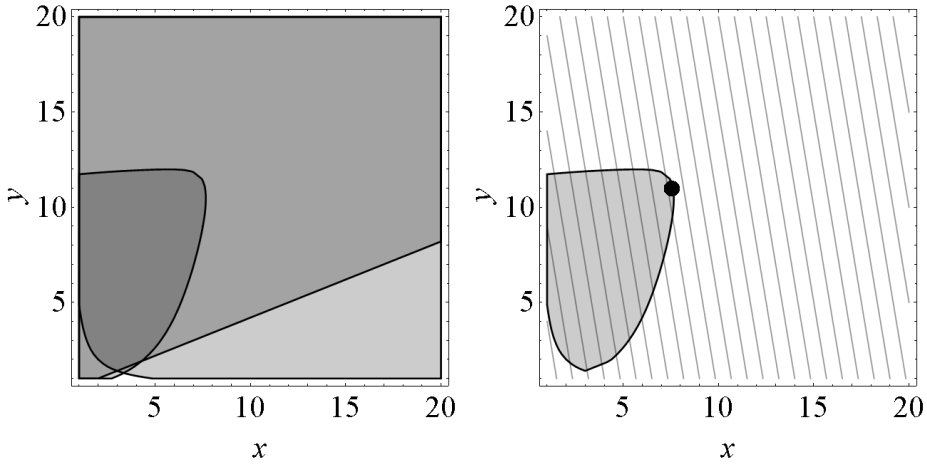


Figure 6.1: The figure to the left shows the feasible regions of the constraints in problem Ex 1. The second figure shows the integer relaxed feasible region, contours of the objective and the optimal solution.

Later, we use the same example to illustrate the differences between the original OA and the proposed methods. To make the results comparable, we will use the starting point, $x^0 = 5.29$, $y^0 = 3$ with all the methods. Instead of solving the relaxed problem in the initialization step in Algorithm 5, we simply use (x^0, y^0) as a starting point. OA required 7 iterations to solve this problem, of which the first six iterations are shown in Figure 6.2. For this specific problem, the first four iterations all result in infeasible solutions where one of the nonlinear constraints are violated. The optimal solution is obtained in iteration five, but verifying optimality requires two additional iterations.

Next, we will show how ideas from the level method can be combined with OA to obtain a stabilized approach for choosing new integer combinations.

6.3 Level-based OA

The level method was originally presented in [257], as a method for solving non-smooth NLP problems. Like OA, the level method also constructs a linear approximation of the

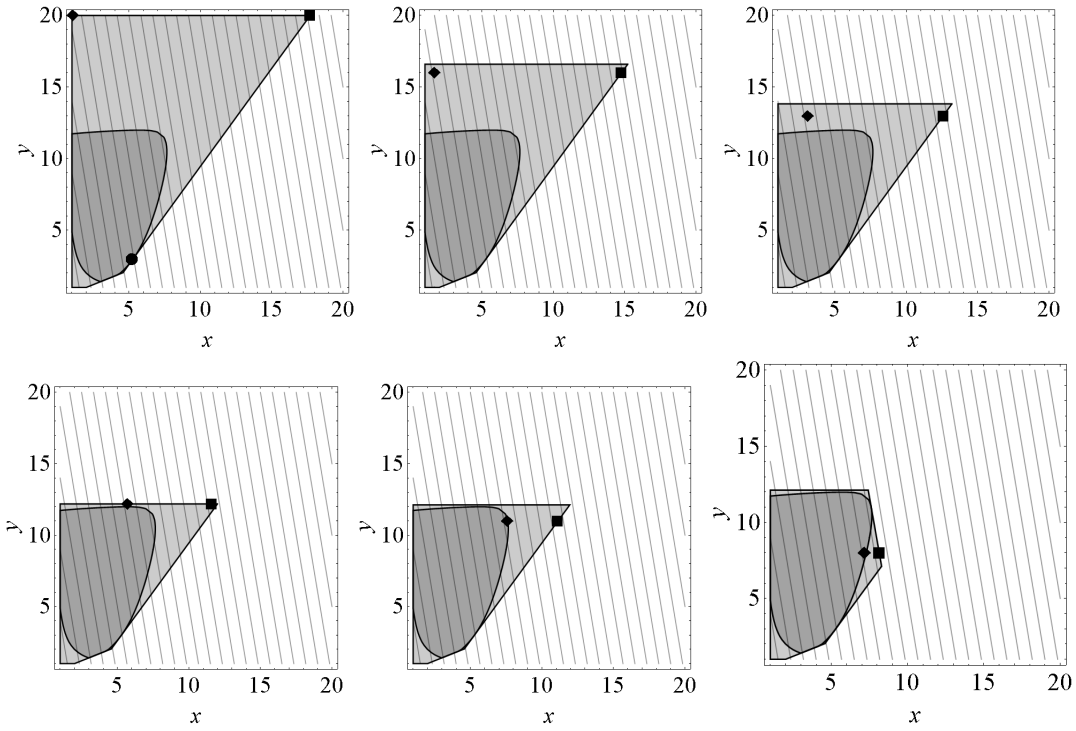


Figure 6.2: The figures show the feasible region defined by the nonlinear constraints in dark gray, and the light gray areas show the outer approximation obtained by the generated cuts. The squared dots represent the solutions obtained from the MILP subproblem and diamond shaped dots represent the solutions obtained by one of the NLP subproblems. The dot in the first figure shows the starting point (x^0, y^0).

original optimization problem. However, the trial solutions are not chosen as the minimizer of the linear approximation. Instead, the trial solutions are obtained by projecting the current solution onto a specific level set of the linearly approximated objective function. For more details see [252, 258]. Here we will use a similar approach combined with OA, which we show is equivalent to adding specific trust regions to the problems OA-master in the original OA.

Here we assume that a feasible solution to the MINLP problem $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is known. Such a solution can for example be obtained by first preforming some original OA iterations or by using a specific procedure such as the feasibility pump [143]. An upper bound to the

MINLP problem is, thus, given by $f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ and cuts at $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ can be generated according to (6.1) to form problem OA-master. A valid lower bound LB^1 can be obtained by solving the linear subproblem OA-master, and thus, the bounds for the optimal objective f^* are given by $LB^1 \leq f^* \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$.

From the bounds of the optimal objective we can in each iteration k estimate a value of the optimal objective according to,

$$\widehat{f_k^*} = (1-\alpha) f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) + \alpha LB^k, \tag{6.2}$$

where $\alpha \in (0, 1]$ and $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is chosen as the best found feasible solution, similarly as in the level method. The lower bound LB^k is obtained as in the original OA, by solving problem OA-master. In Eq. (6.2) α is a parameter which represents how much the linear approximation of the MINLP problem is trusted. Setting α close to one results in an estimated optimum $\widehat{f_k^*}$ close to the lower bound, while setting it close to zero results in an estimated optimum close to the best incumbent solution. The next integer solution $\mathbf{y^{k+1}}$ can now be obtained by projecting $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ onto the $\widehat{f_k^*}$ level set of the linearly approximated objective function. The projection is performed by solving the following MIQP problem,

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu} \quad & \left\| \begin{matrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{matrix} \right\|^2 \\
\text{s.t.} \quad & \mu \leq \widehat{f_k^*} \\
& f(\mathbf{x^i}, \mathbf{y^i}) + \nabla f(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq \mu \quad \forall i = 1, \ldots, k, \\
& g_j(\mathbf{x^i}, \mathbf{y^i}) + \nabla g_j(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq 0 \quad \forall i = 1, \ldots k, \forall j \in \mathcal{I}_i, \\
& \mathbf{Ax} + \mathbf{By} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n, \ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R},
\end{aligned} \tag{MIQP-Proj}$$

where $\|\cdot\|$ is the Euclidean norm. The MIQP problem should contain all the supporting hyperplanes and cutting planes present in problem OA-master, which was solved to obtain the lower bound. The objective function of problem MIQP-Proj introduces a regularization to each iteration, by penalizing the change from the best known solution (the step size). The next integer solution $\mathbf{y^{k+1}}$ is thereby chosen as a point as close as possible to the best known feasible solution which reduces the linearly approximated objective to at most $\widehat{f_k^*}$. Since $\widehat{f_k^*}$ is calculated according to (6.2) there always exists a solution to the MIQP problem, *e.g.*, the minimizer of problem OA-master will satisfy all the constraints. Once the new integer combination is obtained, the corresponding continuous variables can be determined using the same technique as described in the previous section. We summarize the Level-based Outer-approximation as a pseudo-code in Algorithm 6.

The regularization will not only reduce the step size between the iterative solutions, but it will also try to keep the trial solutions close to the best known solution and simultaneously close to the feasible set. This gives an advantage over the original OA, especially, in early iterations where the linear MILP-master problems might only provide a poor approximation which can result in trial solutions far from the feasible set.

The main difference of L-OA compared to OA is the two-step procedure for obtaining the new integer combination, which involves both the solution of an MILP and an MIQP subproblem. This increases the complexity of each iteration. However, as we will prove later the MIQP need not to be solved to optimality. Basically, any feasible solution to the MIQP will be sufficient for ensuring convergence. The computational aspects are described in more detail in Section 6.6 and convergence of both L-OA and Q-OA is proved in Section 6.5.

To obtain a geometrical understanding of how L-OA differs from the original OA, we again consider problem Ex 1. Here we use the same starting point as before and we set the level parameter as $\alpha = 0.4$. To solve the problem with these parameters L-OA requires

Algorithm 6 An algorithm summarizing the basic steps of Level-based Outer-approximation (L-OA) method

Define accepted optimality gap $\epsilon \geq 0$ and choose the parameter $\alpha \in (0, 1]$.

1. Initialization.
   - 1.1 Obtain a feasible solution $\bar{\mathbf{x}}, \bar{\mathbf{y}}$, either by OA or by any other technique.
   - 1.2 Generate cuts at $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ according to (6.1) and construct problems OA-master and (MIQP-Proj).
   - 1.3 Set iteration counter $k = 1$, and $LB^0 = -\inf$.
2. Repeat until $f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - LB^{k-1} \leq \epsilon$.
   - 2.1 Solve problem OA-master to obtain LB^k
   - 2.2 Calculate the estimated optimal value $\widehat{f_k^*}$ according to (6.2).
   - 2.3 Solve problem MIQP-Proj to obtain $\mathbf{y^k}$
   - 2.4 Solve problem NLP-I with integer variables fixed as $\mathbf{y^k}$ to obtain $\mathbf{x^k}$.
     - 2.4.1 If problem NLP-I is infeasible, obtain $\mathbf{x^k}$ by solving feasibility problem NLP-f.
   - 2.5 Generate cuts at $\mathbf{x^k}, \mathbf{y^k}$ according to (6.1) and add these to problems OA-master and (MIQP-Proj).
   - 2.6 If $\mathbf{x^k}, \mathbf{y^k}$ is feasible and $f(\mathbf{x^k}, \mathbf{y^k}) \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, set $\bar{\mathbf{x}}, \bar{\mathbf{y}} = \mathbf{x^k}, \mathbf{y^k}$.
   - 2.7 Increase iteration counter, $k = k + 1$
3. Return $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ as the optimal solution.

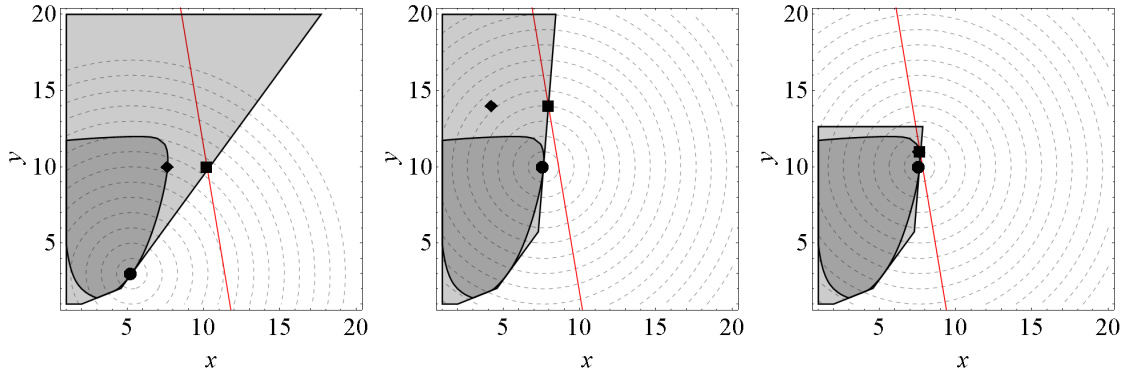


Figure 6.3: The figure illustrates the first three iterations needed to solve problem Ex 1 with the L-OA method. The dashed circles represent the contours of the objective function in the MIQP subproblems and the red line shows the level constraint given by $\mu \leq \widehat{f_k^*}$. The circular dots represent the best-found solution so far, the squared dots represent the solutions obtained from the MIQP subproblem and diamond shaped dots represent the solutions obtained by one of the NLP subproblems.

four iterations. The three first iterations are shown in Figure 6.3. In the fourth iteration, we are able to verify optimality directly after solving the MILP subproblem since we obtain $LB^4 = f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$.

As mentioned earlier, L-OA will find similar integer solutions as adding specific trust regions to the MILP subproblems in the original OA. This property is further described in Theorem 6.3.1.

Theorem 6.3.1. The procedure of solving problems OA-master and (MIQP-Proj) will result in a solution equivalent to adding the trust region constraint

$$\left\| \begin{matrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{matrix} \right\|^2 \leq r_k, \tag{6.3}$$

to problem OA-master in the original OA, where r_k is chosen as the optimum of problem MIQP-Proj.

Proof. First, assume that there exists a unique solution to problem MIQP-Proj, and denote the minimizer as $\mathbf{x}^{\text{MIQP}}, \mathbf{y}^{\text{MIQP}}, \mu^{\text{MIQP}}$. As stated the radius of the trust region constraint is chosen as

$$r_k = \left\| \begin{matrix} \mathbf{x}^{\text{MIQP}} - \bar{\mathbf{x}} \\ \mathbf{y}^{\text{MIQP}} - \bar{\mathbf{y}} \end{matrix} \right\|^2 \tag{6.4}$$

Adding the trust region constraint given by Eq. (6.3) with radius r_k to problem OA-master gives the solution $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$. Now, assume this solution is not the same as the MIQP solution. Since the MIQP solution is assumed to be unique and not equal to the MILP solution, it follows that,

$$r_k > \left\| \begin{matrix} \mathbf{x}^{\text{MILP}} - \bar{\mathbf{x}} \\ \mathbf{y}^{\text{MILP}} - \bar{\mathbf{y}} \end{matrix} \right\|^2. \tag{6.5}$$

Furthermore, since OA-master minimizes μ we get $\mu^{\text{MILP}} \leq \mu^{\text{MIQP}} \leq \widehat{f}_k^*$. This leads to a contradiction since $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$ would then define a feasible solution to problem MIQP-Proj

with an objective strictly lower than the solution obtained by solving the minimization problem.

Next, we consider the case where there is not a unique optimal solution to problem MIQP-Proj, but multiple optimal solutions. As before, we assume that $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$ is not an optimal solution to problem MIQP-Proj. However, Eq. (6.5) must still hold with strict inequality, since the MILP solution satisfies the trust region constraint given by Eq. (6.3) and it is not an optimal solution to problem MIQP-Proj. This leads to the same contradiction as in the case of a unique solution, and therefore, the MILP solution must be an optimal solution to problem MIQP-Proj. □

Note that there are no practical implications that follow from Theorem 6.3.1 because the radius of the trust region resulting in similar solutions cannot be determined in advance. However, the theorem proves that the procedure used in L-OA can be viewed as a technique of using a trust region with OA. Next, we show that it is possible to use a similar approach as L-OA to incorporate second order information in the task of obtaining the integer combinations.

6.4 Quadratic Outer-approximation

In order to obtain better integer solutions, it would be desirable to use information regarding the curvature of the constraints and objective in the task of choosing the integer combinations. We propose a technique where second-order information is incorporated by minimizing a second order Taylor series expansion of the Lagrangean function, which was also suggested in [25]. By using the Lagrangean it is possible to include curvature of both the constraints and objective while keeping the constraints of the subproblems linear.

Here we define the Lagrangean function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l \to \mathbb{R}$ as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \lambda) = f(\mathbf{x}, \mathbf{y}) + \sum_{j=1}^{l} \lambda_j g_j(\mathbf{x}, \mathbf{y}), \tag{6.6}$$

where $\lambda_j \geq 0$ is the Lagrange multiplier of the j-th nonlinear constraint. We do not include the linear constraints in the Lagrangean, since these are handled directly in the subproblems. The Lagrangean is frequently used in NLP techniques and has the following important properties

Properties 1. If all nonlinear functions $f, g_1, \ldots, g_j$ in problem MINLP are convex, then for nonnegative multipliers the Lagrangean defined in Eq. (6.6) will be a convex function in the $\mathbf{x}, \mathbf{y}$ variables, see, *e.g.,* [227, 260].

Properties 2. Strong duality holds for convex optimization problems that satisfy Slater's condition; *i.e.,* there exists valid multipliers such that the minimum of the Lagrangean is equal to the minimum of the original problem [227].

Since the MINLP problems are non-convex by nature, we cannot expect strong duality to hold. However, the first property is important since it will ensure that the subproblem we use for finding the integer combinations will be tractable. We do not want to directly minimize the Lagrangean, because, that problem is basically as difficult as the original problem. Therefore, we will use a second order approximation of the Lagrangean, which is given by

$$\mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda}) + \nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})^T \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix}^T \nabla^2_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda}) \begin{bmatrix} \Delta\mathbf{x} \\ \Delta\mathbf{y} \end{bmatrix}, \tag{6.7}$$

where $\nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}$ is the gradient of the Lagrangean with respect to $\mathbf{x}, \mathbf{y}$ and $\nabla^2_{\mathbf{x},\mathbf{y}}$ denotes the Hessian matrix. To make the notation more compact we have introduced the Δ-variables that are given by $\Delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ and $\Delta\mathbf{y} = \mathbf{y} - \bar{\mathbf{y}}$. Due to Property 1, we know that that the Hessian $\nabla^2_{\mathbf{x},\mathbf{y}}$ will be positive semidefinite for all $\lambda \geq \mathbf{0}$. For small changes in the Δ-variables

Eq. (6.7) should give a good approximation, although the approximation does not under- or overestimate the real Lagrangean function.

The natural approach of using the quadratic approximation in OA would be to replace the linear objective of the MILP-master problem with the quadratic function given by Eq. (6.7). However, this approach will not guarantee convergence on its own, because, unlike the original OA the quadratic master problem will not always result in new integer combinations. Since the second order approximation does not necessarily underestimate the Lagrangean, it is possible that the approximation point $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is the optimum of the approximation even if it is not the optimal solution to the original problem, and thus, the approach could stagnate at non-optimal solutions. To avoid this, the method presented in [25] uses an ϵ improvement strategy, where the next solution must reduce the linearly approximated objective by a small ϵ-value. The ϵ improvement is enforced by the following constraints

$$\begin{aligned} &\mu \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - \epsilon \\ &f(\mathbf{x^i}, \mathbf{y^i}) + \nabla f(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq \mu \quad \forall i = 1, \ldots, k, \end{aligned} \tag{6.8}$$

where $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is the best found solution. With this approach ϵ must be chosen smaller than the desired optimality gap. Thus, it will only result in a small reduction requirement. Therefore, the Quadratic Outer-approximation method in [25] will rely heavily on the second order approximation of the Lagrangean. In case the approximation point $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ with the corresponding multipliers $\bar{\lambda}$ is not the optimal solution to the MINLP, then the Lagrangean might not give a good approximation of the original problem and this might cause slow convergence. Due to the discrete nature of MINLP problems, it is possible that only the optimal integer combination with the corresponding continuous variables will result in the optimal set of active constraints and nonzero multipliers.

In this case, we use a different approach, which combines information from both the linear approximation with the quadratic approximation of the Lagrangean, to make sure the proposed method does not stagnate at non-optimal solutions. By using the same approach as in L-OA, an estimate of the optimal objective $\widehat{f_k^*}$ can be calculated according to Eq. (6.2). The estimated optimum can further be used to construct the following reduction constraint,

$$\begin{aligned} &\mu \leq \widehat{f_k^*} \\ &f(\mathbf{x^i}, \mathbf{y^i}) + \nabla f(\mathbf{x^i}, \mathbf{y^i})^T \begin{bmatrix} \mathbf{x} - \mathbf{x^i} \\ \mathbf{y} - \mathbf{y^i} \end{bmatrix} \leq \mu \quad \forall i = 1, \ldots, k. \end{aligned} \tag{6.9}$$

As long as $\widehat{f_k^*}$ is calculated using the same technique as in L-OA there will always exist a solution that satisfies the reduction constraints in Eq. (6.9). Furthermore, since $\widehat{f_k^*}$ is chosen as an interpolation between the upper and lower bound it will usually result in a stricter reduction constraint. We will construct the master problem by minimizing the quadratic approximation of the Lagrangean with the reduction constraint given by Eq. (6.9), the accumulated cuts given by Eq. (6.1) and all linear constraints from the MINLP problem. The new integer combination y^{k+1} is, thus, obtained by solving the following MIQP problem,

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu} \quad & \nabla_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})^T \begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{y}\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{y}\end{bmatrix}^T \nabla^2_{\mathbf{x},\mathbf{y}}\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda}) \begin{bmatrix}\Delta\mathbf{x}\\ \Delta\mathbf{y}\end{bmatrix} \\
\text{s.t.} \quad & \mu \leq \widehat{f_k^*} \\
& f(\mathbf{x^i},\mathbf{y^i}) + \nabla f(\mathbf{x^i},\mathbf{y^i})^T \begin{bmatrix}\mathbf{x}-\mathbf{x^i}\\ \mathbf{y}-\mathbf{y^i}\end{bmatrix} \leq \mu \quad \forall i = 1,\ldots,k \\
& g_j(\mathbf{x^i},\mathbf{y^i}) + \nabla g_j(\mathbf{x^i},\mathbf{y^i})^T \begin{bmatrix}\mathbf{x}-\mathbf{x^i}\\ \mathbf{y}-\mathbf{y^i}\end{bmatrix} \leq 0 \quad \forall i = 1,\ldots k, \forall j \in \mathcal{I}_i, \\
& \mathbf{Ax} + \mathbf{By} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R},
\end{aligned}
\tag{QOA-master}
$$

where $\Delta\mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$ and $\Delta\mathbf{y} = \mathbf{y} - \bar{\mathbf{y}}$. As in L-OA $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ is chosen as the best found feasible solution and $\bar{\lambda}$ are the corresponding Lagrangean multipliers obtained by solving problem NLP-I. The NLP subproblem with fixed integer variables will provide both the $\mathbf{x}$ variables and the multipliers λ. If the NLP subproblem is infeasible we solve the problem NLP-f, from which we obtain the corresponding multipliers. As mentioned before $\nabla^2_{\mathbf{x},\mathbf{y}}$ is positive semidefinite due to the convexity of the nonlinear functions; therefore, the MIQP problem can be solved efficiently with software such as Gurobi[51] or CPLEX[52].

Once the next integer solution has been obtained, the continuous variables are determined as in OA or L-OA, and more cuts are generated according to Eq. (6.1). The lower bound is updated in each iteration as in L-OA by solving problem OA-master. The Quadratic Outer-approximation method is summarized as a pseudocode in Algorithm 7.

As in L-OA, each iteration includes both an MILP and an MIQP subproblem. We will show later that it is sufficient to merely obtain a feasible solution to the MIQP, which can reduce the computational complexity of both the L-OA and Q-OA method. Section 6.5 proves the method's convergence to the optimal solution in a finite number of iterations,

Algorithm 7 An algorithm summarizing the basic steps of the Quadratic Outer-approximation (Q-OA) method

Define accepted optimality gap $\epsilon \geq 0$ and choose the parameter $\alpha \in]0, 1]$.

1. Initialization.
   - 1.1 Obtain a feasible solution $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ and the multipliers $\bar{\lambda}$, either by OA or by any other technique.
   - 1.2 Generate cuts at $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ according to (6.1) and construct problems OA-master and (QOA-master).
   - 1.3 Set iteration counter $k = 1$, and $LB^0 = -\inf$.
2. Repeat until $f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - LB^{k-1} \leq \epsilon$.
   - 2.1 Solve problem OA-master to obtain LB^k
   - 2.2 Calculate the estimated optimal value $\widehat{f_k^*}$ according to (6.2).
   - 2.3 Solve problem QOA-master to obtain $\mathbf{y^k}$
   - 2.4 Solve problem NLP-I with integer variables fixed as $\mathbf{y^k}$ to obtain $\mathbf{x^k}$ and $\lambda^{\mathbf{k}}$.
     - 2.4.1 If problem NLP-I is infeasible, obtain $\mathbf{x^k}$ by solving feasibility problem NLP-f.
   - 2.5 Generate cuts at $\mathbf{x^k}, \mathbf{y^k}$ according to (6.1) and add these to problems OA-master and (QOA-master).
   - 2.6 If $\mathbf{x^k}, \mathbf{y^k}$ is feasible and $f(\mathbf{x^k}, \mathbf{y^k}) \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, set $\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda} = \mathbf{x^k}, \mathbf{y^k}, \lambda^{\mathbf{k}}$.
   - 2.7 Increase iteration counter, $k = k + 1$
3. Return $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ as the optimal solution.

and Section 6.6 discusses the computational aspect more in detail.

The technique used for obtaining the integer combinations in Q-OA actually results in an interpolation between the minimizer of the linear approximation in problem OA-master and the minimizer of the Lagrangean approximation, where α in Eq. (6.2) is the interpolation parameter. Setting $\alpha = 1$ will force the solution of problem QOA-master to the minimizer of problem OA-master, and setting α close to zero will allow the solution to be close to the minimizer of the Lagrangean approximation. The Q-OA method will, therefore, be less sensitive to the accuracy of the Lagrangean approximation, compared to the method in [25]. In the next section, we prove that finite convergence of Q-OA can still be guaranteed even if the Hessian of the Lagrangean is only estimated as long as it remains positive semidefinite.

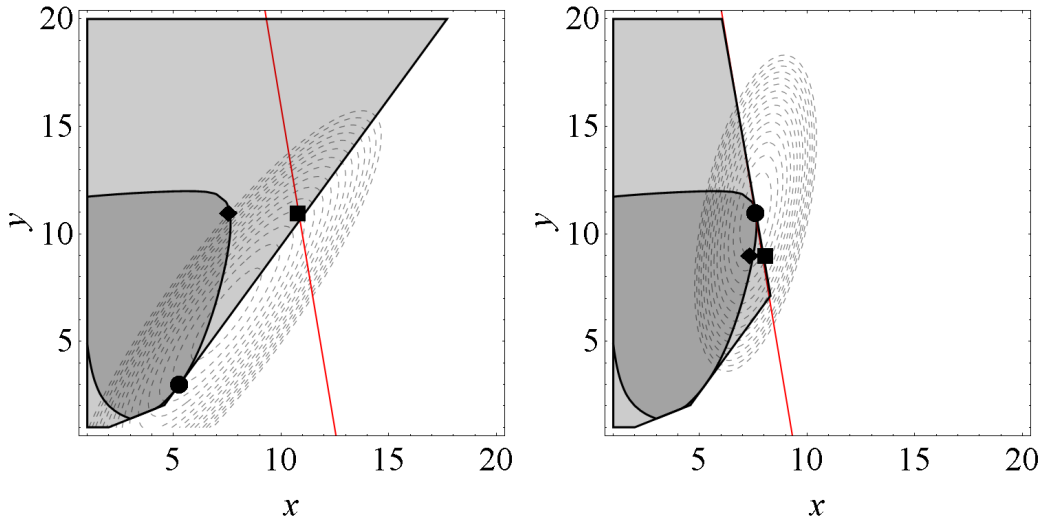


Figure 6.4: The figures illustrate the first two iterations needed to solve problem Ex 1 with the Q-OA method. The dashed ellipses represent the contours of the approximated Lagrangean used as the objective in the MIQP subproblem and the red line shows the level constraint given by $\mu \leq \widehat{f_k^*}$. The circular dots represent the best found solution so far, the squared dots represent the solutions obtained from the MIQP subproblem and diamond shaped dots represent the solutions obtained by one of the NLP subproblems.

To provide a geometric interpretation of the method and to show how it differs from OA and L-OA, we apply the method to the illustrative test problem Ex 1. We use the same starting point (x^0, y^0) as before and we set the level parameter as $\alpha = 0.5$. To solve the problem with these parameters Q-OA requires three iterations. The first two iterations are shown are shown in Figure 6.4. In the third iteration, we are able to verify optimality after only solving the MILP subproblem, since we obtain $LB^3 = f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. From the figure, note that the reduction constraint given by Eq. (6.9), prevents the algorithm form taking a too short step in the first iteration and the optimal solution is actually obtained in the first iteration. If the trial solution had only been chosen as the minimizer of the Lagrangean relaxation, it would have resulted in less progress per iteration. It should also be noted that not a single infeasible integer combination was encountered.

6.5 Convergence properties

Proving finite convergence of L-OA and Q-OA can be done similarly as for the original OA, and some of the results from [25, 30] are directly applicable. Finite convergence can be proven as follows. We show that an infeasible integer combination obtained by L-OA or Q-OA will be cut off by the cuts generated according to Eq. (6.1) and therefore, this integer combination cannot be obtained in any future iteration. Next, we prove that a specific integer combination cannot be obtained twice with either method unless optimality is proven. The methods, therefore, obtain new integer combinations at each iteration, and since there are only a finite number of such combinations, the methods will converge in a finite number of iterations.

Convexity of the nonlinear functions is crucial since it ensures that no feasible integer solution is cut off by the cuts generated by L-OA or Q-OA and that problem OA-master gives a valid lower bound, as is stated in Lemma 6.5.1. The lemma and a proof is also found in [25].

Lemma 6.5.1. Solving problem OA-master yields a valid lower bound to the optimum of the MINLP problem.

Proof. From the first order convexity condition we know that for any convex differentiable function $\phi(\mathbf{x},\mathbf{y})$,

$$\phi(\mathbf{x},\mathbf{y}) \geq \phi(\mathbf{x^0},\mathbf{y^0}) + \nabla\phi(\mathbf{x^0},\mathbf{y^0})^T \begin{bmatrix} \mathbf{x}-\mathbf{x^0} \\ \mathbf{y}-\mathbf{y^0} \end{bmatrix} \quad \forall(\mathbf{x},\mathbf{y}),(\mathbf{x^0},\mathbf{y^0}) \in D_\phi,$$

where D_ϕ is the domain in which the function is convex. Therefore the feasible region of the problem MINLP will be overestimated and the objective function will be underestimated at each iteration. □

In Theorem6.5.1, we prove that L-OA and Q-OA always find new integer combinations as long as optimality is not guaranteed, which requires some intermediate results given in the following two lemmas.

Lemma 6.5.2. An infeasible integer combination $\mathbf{y^k}$, *i.e.,* an integer combination such that problem NLP-I is infeasible, will be cut off by the cuts generated in L-OA and Q-OA.

Proof. It is proved in [25], that solving the feasibility problem and adding cuts for the active constraints will cut off $\mathbf{y^k}$ from the search space. For more details see [25, Lemma 1, page 331]. □

Lemma 6.5.3. If the lower bound is not equal to the upper bound, then there exists a solution to the MIQP subproblems in L-OA and Q-OA.

Proof. Due to convexity, the linearly approximated problem OA-master is always feasible if the MINLP problem is feasible. The MIQP subproblem in both L-OA and Q-OA contains the same constraints as problem OA-master and the reduction constraint. Since $\widehat{f_k^*}$ is calculated according to Eq. (6.2), the solution to problem OA-master is a feasible solution to the MIQP subproblem. If problem OA-master is infeasible, the search will be terminated since it verifies that the MINLP is infeasible. □

Theorem 6.5.1. If the lower bound is not equal to the upper bound, then the MIQP subproblems in L-OA and Q-OA will give a new integer combination.

Proof. By Lemma 6.5.2, we know that any infeasible integer combination that has been found is cut off from the search space by the cuts added to the subproblems. Since the upper and lower bound are not equal, we know that the estimated optimum will be smaller than the upper bound, *i.e.,* $\widehat{f_k^*} < f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$. This is obviously true for all feasible solutions found

so far, which we denote as $\widehat{\mathbf{x}}^{\mathbf{i}}, \widehat{\mathbf{y}}^{\mathbf{i}}$, and the following relation is obtained,

$$\widehat{f_k^*} < f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \leq f(\widehat{\mathbf{x}}^{\mathbf{i}}, \widehat{\mathbf{y}}^{\mathbf{i}}) \quad \forall i. \tag{6.10}$$

At all the obtained feasible solutions $\widehat{\mathbf{x}}^{\mathbf{i}}, \widehat{\mathbf{y}}^{\mathbf{i}}$ the methods generate the following linearizations of the objective,

$$f(\widehat{\mathbf{x}}^{\mathbf{i}}, \widehat{\mathbf{y}}^{\mathbf{i}}) + \nabla f(\widehat{\mathbf{x}}^{\mathbf{i}}, \widehat{\mathbf{y}}^{\mathbf{i}})^T \begin{bmatrix} \mathbf{x} - \widehat{\mathbf{x}}^{\mathbf{i}} \\ \mathbf{y} - \widehat{\mathbf{y}}^{\mathbf{i}} \end{bmatrix} \leq \mu. \tag{6.11}$$

In both the MIQP subproblem in L-OA and in Q-OA, we have the reduction constraint $\mu \leq \widehat{f_k^*}$, and from Eq. (6.11) it follows that the next solution must satisfy,

$$\nabla f(\widehat{\mathbf{x}}^{\mathbf{i}}, \widehat{\mathbf{y}}^{\mathbf{i}})^T \begin{bmatrix} \mathbf{x} - \widehat{\mathbf{x}}^{\mathbf{i}} \\ \mathbf{y} - \widehat{\mathbf{y}}^{\mathbf{i}} \end{bmatrix} < 0 \quad \forall i. \tag{6.12}$$

Now, assume that one of the feasible solutions $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}} \in \{\widehat{\mathbf{x}}^{\mathbf{i}}, \widehat{\mathbf{y}}^{\mathbf{i}}\}$ can be perturbed in the $\mathbf{x}$-variables by $\Delta\mathbf{x}$ such that it satisfies all constraints of the MIQP subproblem and the property given by Eq. (6.12). Since $\widetilde{\mathbf{x}}$ was obtained by solving problem NLP-I, it must satisfy the KKT-conditions,

$$\begin{aligned}
&\nabla_{\mathbf{x}} f(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}) + \sum_{j=1}^{l} \lambda_j \nabla_{\mathbf{x}} g_j(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}) + \mathbf{A}^T \gamma = \mathbf{0} \\
&g_j(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}) \leq 0 \quad \forall j = 1, \ldots, l \\
&\mathbf{A}\widetilde{\mathbf{x}} + \mathbf{B}\widetilde{\mathbf{y}} \leq \mathbf{b} \\
&\lambda, \gamma \geq \mathbf{0} \\
&\lambda_j g_j(\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}) = 0 \quad \forall j = 1, \ldots, l \\
&(\mathbf{A}\widetilde{\mathbf{x}} + \mathbf{B}\widetilde{\mathbf{y}} - \mathbf{b}) \circ \gamma = \mathbf{0},
\end{aligned} \tag{6.13}$$

where $\nabla_{\mathbf{x}}$ is the gradient with respect to $\mathbf{x}$-variables, and γ are the multipliers of the linear constraints. At the solution $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$, the methods will generate the following supporting hyperplanes,

$$g_j(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}}) + \nabla g_j(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})^T \begin{bmatrix} \mathbf{x}-\widetilde{\mathbf{x}} \\ \mathbf{y}-\widetilde{\mathbf{y}} \end{bmatrix} \leq 0 \quad \forall j \mid \lambda_j \neq 0. \tag{6.14}$$

Since these are all active constraints, the constant on the left-hand side must be zero, *i.e.*, $g_j(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}}) = 0$. The perturbation $\Delta\mathbf{x}$ must satisfy Eq. (6.14), which can be written as ,

$$\lambda_j \nabla_x g_j(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})^T \Delta\mathbf{x} \leq 0 \quad \forall j = 1,\ldots,l. \tag{6.15}$$

The same is also true for the linear constraints. For all active linear constraints $\Delta\mathbf{x}$ cannot increase the value of the left hand side. This condition can be summed over all linear constraints by the multipliers γ as

$$\gamma^T \mathbf{A}\Delta\mathbf{x} \leq 0. \tag{6.16}$$

The perturbation also has to satisfy the reduction stated in Eq. (6.12), which yields,

$$\nabla_x f(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})^T \Delta\mathbf{x} < 0. \tag{6.17}$$

Adding all inequalities from Eq. (6.15), (6.16) and (6.17) results in the following strict inequality,

$$\nabla_{\mathbf{x}} f(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})^T \Delta\mathbf{x} + \sum_{j=1}^{l} \lambda_j \nabla_{\mathbf{x}} g_j(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})^T \Delta\mathbf{x} + \gamma^T \mathbf{A}\Delta\mathbf{x} < 0. \tag{6.18}$$

However, taking the inner product of $\Delta\mathbf{x}$ and both sides of the first KKT condition results in the following equality

$$\nabla_{\mathbf{x}} f(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})^T \Delta\mathbf{x} + \sum_{j=1}^{l} \lambda_j \nabla_{\mathbf{x}} g_j(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})^T \Delta\mathbf{x} + \gamma^T \mathbf{A}\Delta\mathbf{x} = 0, \tag{6.19}$$

which leads to a contradiction. Therefore, there cannot exist a $\Delta\mathbf{x}$ that satisfies all constraints of the MIQP subproblems without a change in the $\mathbf{y}$-variables. As stated in Lemma 6.5.3, there always exists a solution to the MIQP subproblems as long as the lower bound is not equal to the upper bound. Solving the MIQP subproblem will, therefore, result in a new integer combination different from all previously obtained solutions. □

Note that, no assumptions were made in Theorem 6.5.1 regarding optimality of the MIQP subproblem. Therefore, the theorem is true for any solution that satisfies all constraints of the MIQP subproblem, optimal or not. Furthermore, Theorem 6.5.1 holds even if we make an arbitrary change to the objective function in the MIQP subproblems. An estimate of the Hessian in Q-OA will, therefore, be sufficient for Theorem 6.5.1 to hold. The next theorem summarizes the convergence properties.

Theorem 6.5.2. Both L-OA and Q-OA will terminate after a finite number of iterations, either by verifying optimality of the best-found solution or proving that the MINLP problem is infeasible.

Proof. From Lemma 6.5.1, it is clear that solving problem OA-master will either provide a valid lower bound or prove infeasibility. Furthermore, the proof of Lemma 6.5.1 also establishes that no feasible solution will be excluded from the search space. According to Theorem 6.5.1, both L-OA and Q-OA will find new integer combinations at each iteration as long as the gap between the upper and lower bound is not equal to zero. Since the linear constraints are assumed to give rise to a compact set, it is clear that there can only exist a finite number of different integer combinations, and thus, both methods must terminate after a finite number of iterations. □

Hence, we have proved that both proposed methods converge to a global optimal solution in a finite number of iterations. In the next section, we present a numerical comparison of the proposed methods and compare the results to the original OA method.

6.6 Computational results

In this section, we discuss our computational experiments and the obtained results. To compare the practical performance of the methods, we have implemented the original OA as

well as L-OA and Q-OA. The main advantage of L-OA and Q-OA compared to the original OA, is the ability to handle highly nonlinear MINLP problems more efficiently. L-OA is more conservative when choosing the trial solutions, and tries to stay close to the best found feasible solution, which should reduce the number of infeasible integer combinations obtained. In Q-OA we are also able to incorporate second order information when choosing new integer combinations. Hence, the new integer combination is chosen with information regarding the curvature around the current solution. From the test problems, we observed a significant reduction in the number of iterations with both L-OA and Q-OA compared to the original OA.

Problem set 1 contains 358 instances and the list the provided in Appendix. Problem set 2 derives from the Problem set 1 by filtering on the following conditions.

To test and compare the methods, 358 MINLP problems are selected from the MINLPlib2 (rev. 373, as of 2017-11-07)⋆ [231].

To test and compare the methods we have implemented them and applied them to convex MINLP problems obtained from MINLPlib2 (rev. 373, as of 2017-11-07)⋆ [231]. This set was chosen since it contains a large variety of different test problems originating from both practical applications as well as theoretical test problems. As mentioned earlier both L-OA and Q-OA are intended for problems with high to medium degrees of non-linearity, and therefore, we used the following criteria for choosing the test problems

1. Classified as convex.
2. Having at least one discrete variable.
3. Having at least one continuous variable.

⋆http://www.gamsworld.org/minlp/minlplib2/html/index.html

⋆http://www.gamsworld.org/minlp/minlplib2/html/index.html

4. Satisfying the following inequality

$$\frac{n_{nonlin}}{n+m} > 0.5, \tag{6.20}$$

where n_{nonlin} is the number of variables present in some nonlinear term and $m+n$ is the total number of discrete and continuous variables. There are in total 109 convex MINLP problems in MINLPlib2 (rev. 373, as of 2017-11-07) that satisfy the given criteria. These problems originate from several applications such as process synthesis, facility layout problems, batch design with storage, portfolio optimization and MINLP test problems. The test instances have between 7 and 4530 variables and 0 to 1822 constraints. More details of the test instances are provided in the supplemental material.

Next, we describe some details regarding the implementation of the methods and the computational results are presented below.

6.6.1 Implementation details

The implementation of the methods compared here was made in MATLAB using Gurobi 7.5.1[51] as subsolver for the MILP/MIQP subproblems and IPOPT 3.12.7 [158] for the NLP subproblems. Furthermore, we use some functionality from OPTI Toolbox [99] to read the test problems.

Both L-OA and Q-OA require a feasible starting solution, and to obtain such a solution we start by performing a few original OA iterations. Once a feasible solution is obtained, we switch to either L-OA or Q-OA. The level parameter α was set to 0.5 with both methods in the comparison.

According to Theorem 6.5.1, it is not necessary to find the optimal solution for the MIQP subproblems, and any feasible solution for these problems is sufficient for guaranteeing that both L-OA and Q-OA converge to the global optimum. This is an important property, since

solvers such as Gurobi or CPLEX are often able to quickly find several feasible solutions, and quite often the majority of the solution time is spent proving optimality. We use a strategy of stopping the solver once a certain number of feasible solutions have been found, and specifically, we stop after 10 solutions have been found. This is simply done by setting the SolutionLimit parameter to 10. Using this approach, we hope to ensure that we obtain a good solution to the MIQP problem, while significantly reducing the total solution time. For the MIQP subproblems, we always have a feasible solution available, the solution to the MILP subproblem OA-master, and providing this as a starting solution to Gurobi also improved the performance. For the MILP subproblems, we used the default settings in Gurobi, and we also used the default settings for IPOPT.

The NLP subproblem NLP-I is always convex for these test problems. However, for some specific test problems we encountered some difficulties where the solver failed to find the optimal solution. Such difficulties could, for example, be caused by a specific integer combination not satisfying the constraint qualifications. These issues were not frequent and they only occurred for a few test problem in the entire MINLPLib2. To deal with such issues we chose a simple approach; if the NLP subproblem NLP-I is feasible but the NLP solver fails, we generate cutting planes for all violated constraints at the solution given by the MILP subproblem OA-master according to Eq. (6.1). These cuts will exclude the current solution to subproblem OA-master from the search space [120], and thus prevent cycling. Adding these cuts is equivalent to performing an iteration of the ECP method. From the convergence properties of the ECP method, we know that adding these cuts will eventually result in a new integer combination or verify optimality of an obtained solution.

Since the problems we consider are all convex, the Hessian of the Lagrangean is always positive semidefinite. However, due to numerical accuracy we did encounter a few cases where the Hessian was not strictly positive semidefinite, *i.e.,* the smallest eigenvalue was

not positive but in the range of -10^{-9}. To make sure that the MIQP subproblems are convex, we slightly modify the diagonal elements of the Hessian. For each row i of the Hessian which contains a nonzero element, we modify the diagonal by

$$\nabla^2_{\mathbf{x},\mathbf{y}}\mathcal{L}(i,i) := \nabla^2_{\mathbf{x},\mathbf{y}}\mathcal{L}(i,i) + |\lambda_{min}|, \tag{6.21}$$

where λ_{min} is chosen as the smallest eigenvalue of the Hessian. This modification guarantees that all eigenvalues are positive [261], and thus, ensures convexity of the MIQP subproblem. The modification of the Hessian is only done in case one of the eigenvalues are negative.

As termination criteria, we used both an absolute optimality tolerance ϵ and a relative optimality tolerance ϵ_{rel}. The search is, thus terminated if either

$$f(\bar{\mathbf{x}},\bar{\mathbf{y}}) - LB \leq \epsilon \quad \text{or} \quad \frac{f(\bar{\mathbf{x}},\bar{\mathbf{y}}) - LB}{|f(\bar{\mathbf{x}},\bar{\mathbf{y}})| + 10^{-10}} \leq \epsilon_{rel}$$

are satisfied. Here LB denotes the current lower bound. These can be considered as the standard termination criteria for MINLP problems.

All tests were performed on an Intel Core i7 2.93GHz CPU desktop with 16GB of RAM running Windows 7. As termination criteria, we set the tolerances $\epsilon = 10^{-5}$ and $\epsilon_{rel} = 10^{-3}$, and a time limit of $900s$.

6.6.2 Illustrative examples

In this section, we present more detailed results of two particular instances of the selected test set. These instances were chosen such that they could exemplify the results shown in the following section. The selected instances are `cvxnonsep_nsig40`* and `ibs2`*. The first instance was proposed by Kronqvist et al. [262], it contains 20 integer variables and 20 continuous variables, a linear objective and a signomial constraint. This seemingly

*http://www.gamsworld.org/minlp/minlplib2/html/cvxnonsep_nsig40.html

*http://www.gamsworld.org/minlp/minlplib2/html/ibs2.html

simple problem is designed to be challenging for methods such as OA and ECP due to a highly nonlinear constraint. The second instance has 1500 binary variables, 1510 continuous variables, a linear objective, and 1821 constraints, of which 10 are nonlinear including square and logarithm operators. This problem represents a particular challenge for the OA method given its combinatorial complexity and the fact that most of its variables, discrete and continuous, are involved in a nonlinear fashion in the constraints.

To illustrate how the methods differ for these problems, we show the upper and lower bounds obtained by each method. Figures 6.5 and 6.6 show the progress of the bounds as a function of time for problems `cvxnonsep_nsig40` and `ibs2`, respectively. From the figures, it can be observed that Q-OA is able to improve the upper bound more quickly than the other methods. This is usually the case and is explained by that fact that Q-OA utilizes more information when choosing the integer combinations than the other methods. Especially for the instance `ibs2`, there was a clear advantage of incorporating information from the second order derivatives, and Q-OA clearly performs better than L-OA.

From the results presented in the bounds profiles and in Table 6.1, we notice how including the level regularization can improve the performance of the OA method while solving convex MINLPs. For the instance `cvxnonsep_nsig40` we notice a reduction in time of 59% and 66% using the L-OA and the Q-OA method, respectively. Although the new methods require the solution of an MIQP subproblem in each iteration, the extra time invested in finding the next integer combination is compensated with a reduction in both iterations and time.

For the instance `ibs2`, OA is unable to close the optimality gap under 0.1% within the time limit of 900 seconds even though it performs 431 iterations. When solving the problem with L-OA the upper bound initially diminishes faster in terms of both time and iterations compared to OA, but the MIQP subproblems become hard to solve resulting in only 41

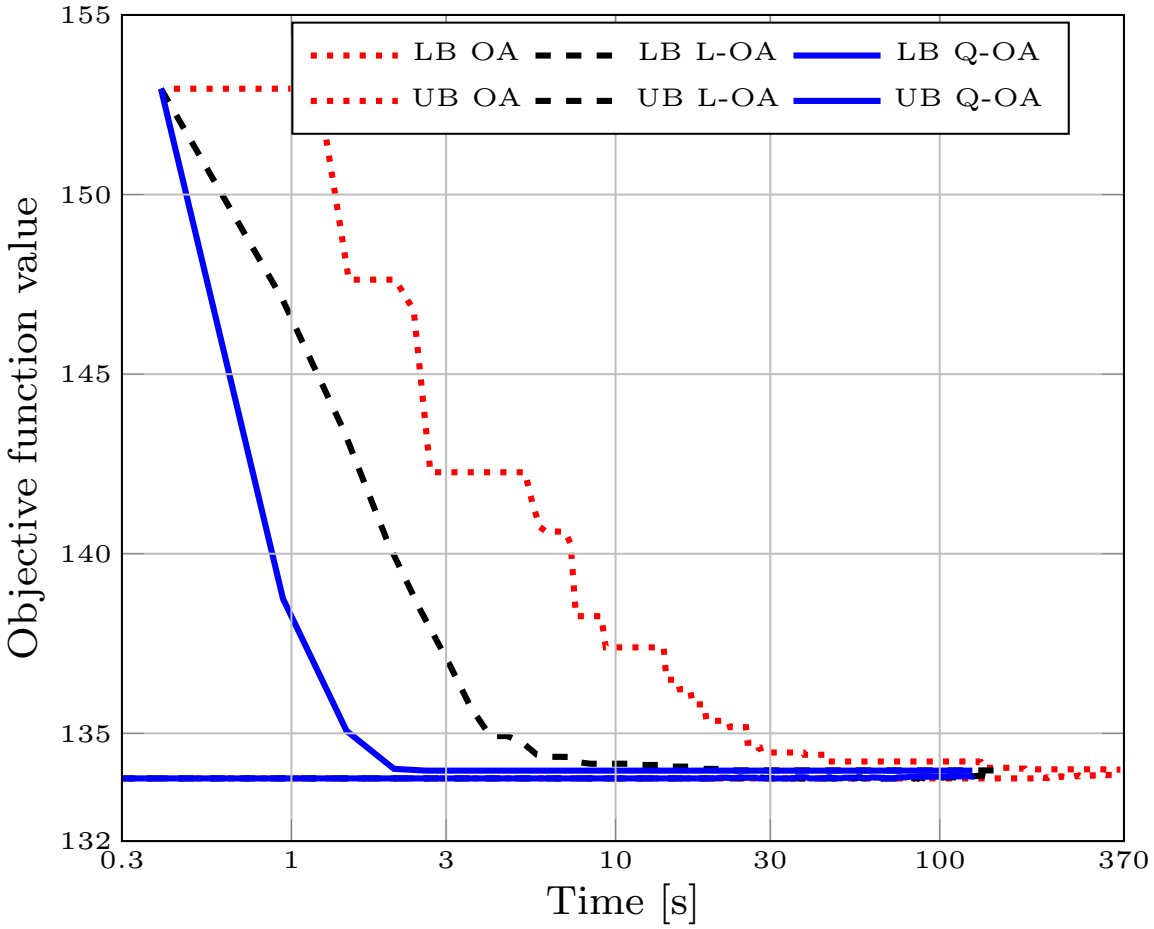


Figure 6.5: Bound profiles for instance `cvxnonsep_nsig40` against time. The figure shows the upper bound (UB) and lower bound (LB) obtained by the OA, L-OA, and Q-OA methods.

Table 6.1: Detailed results of the illustrative examples while solving them with the OA, L-OA, and Q-OA methods

Instance	Solution method	Time [s]	Iterations	NLP time [s]	MILP time [s]	MIQP time [s]	Infeasible NLPs	Optimality Gap
`cvxnonsep_nsig40`	OA	360.86	663	31.69	328.13	0	1	0.00098
	L-OA	147.87	201	9.84	55.35	82.37	0	0.000999
	Q-OA	121.55	144	6.65	38.68	75.99	0	0.000913
`ibs2`	OA	900*	431	152.86	747.14	0	6	0.005368
	L-OA	900*	41	40.16	17.98	841.86	6	0.1093
	Q-OA	103.89	16	48.60	6.469	48.68	2	0.000997

*Time limit.

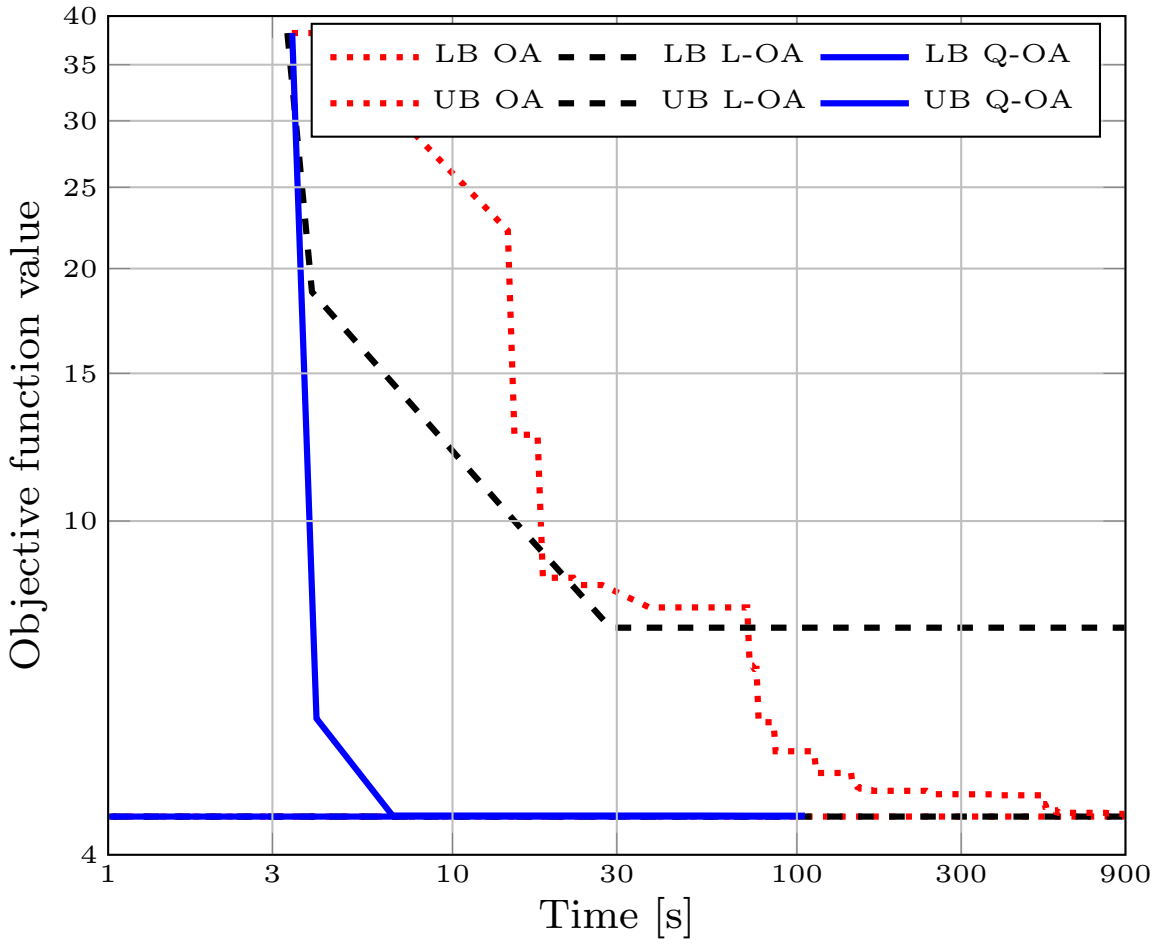


Figure 6.6: Bound profiles for instance `ibs2` against time. The figure shows the upper bound (UB) and lower bound (LB) obtained by the OA, L-OA, and Q-OA methods.

iterations before hitting the time limit. When utilizing second order information with the Q-OA method, the problem is solved within an optimality gap of 0.1% in 104 seconds and just after 16 iterations while only encountering 2 infeasible NLP subproblems. Note that, for both illustrative examples the methods are all able to obtain a tight lower bound already in the first iteration, which is not generally the case. Usually, the lower bounds can also vary significantly between the methods.

6.6.3 Numerical results

Having observed the improvement in performance of the proposed methods compared to OA in the illustrative examples, we considered the whole test set defined at the beginning of this section. In order to compare the performance of the methods, we have used performance profiles [204] both in terms of solution time and iterations in Figures 6.7 and 6.8, respectively. The profiles show the number of problems solved against the respective performance ratio threshold τ. A data point at each plot represents the number of instances that each method solved within a factor τ of the best solver.

Figure 6.7 shows how the Q-OA method is superior to both L-OA and OA for the selected test set in terms of solution time. The figure shows that Q-OA solves most instances to the desired optimality gap, and it solves the problems in the least amount of time. L-OA has initially the worst performance of the 3 methods for $\tau \leq 3$, but in the end, the performance is similar to that of OA without reaching the number of solved instances by Q-OA. It is also worth mentioning, that all the instances that remained unsolved with Q-OA are also unsolved with both OA and L-OA. Q-OA is thus able to solve all the problems solved with the other methods and some additional problems.

The performance profiles in terms of iterations in Figure 6.8 show a clear advantage of Q-OA compared to the other 2 methods. Considering iterations L-OA also performs better

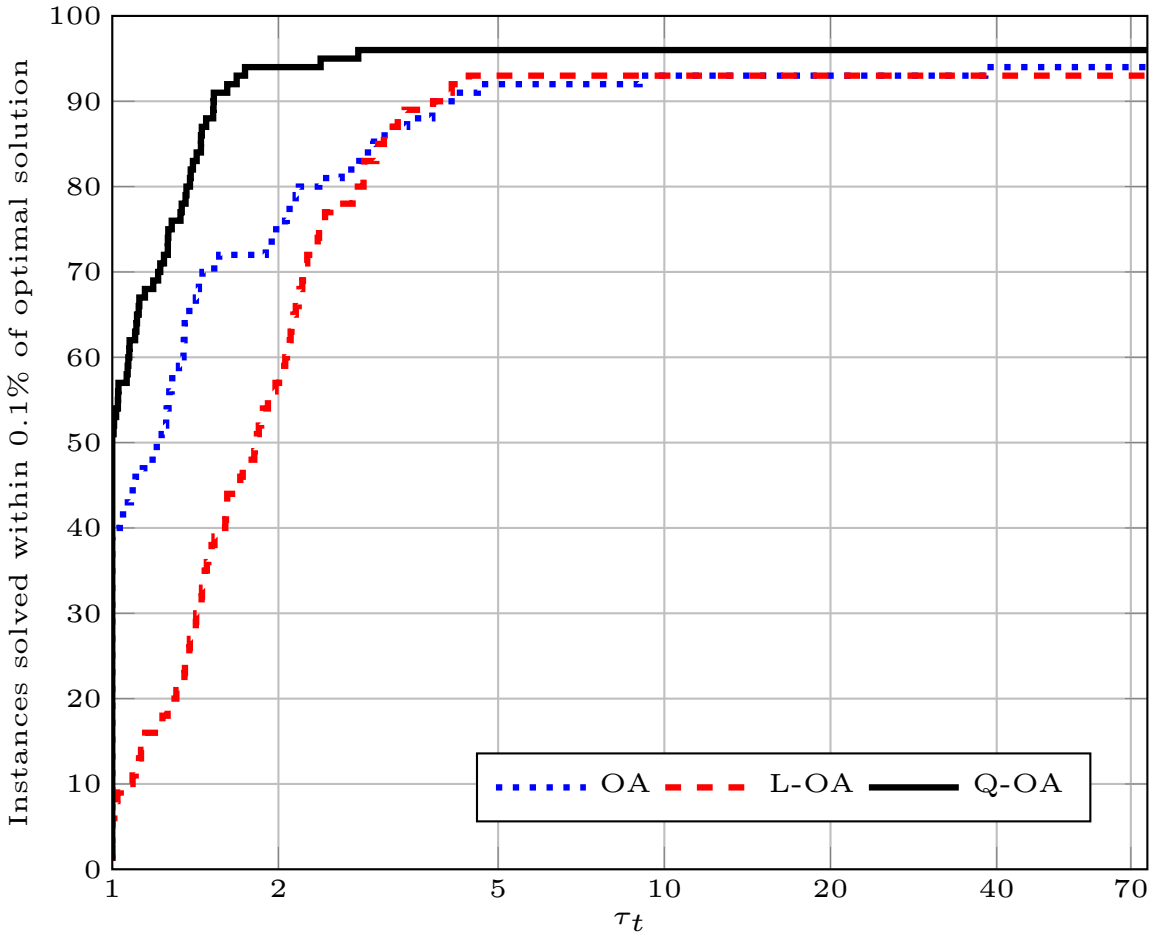


Figure 6.7: Time performance profiles for test problems using second-oder regularization methods in OA.

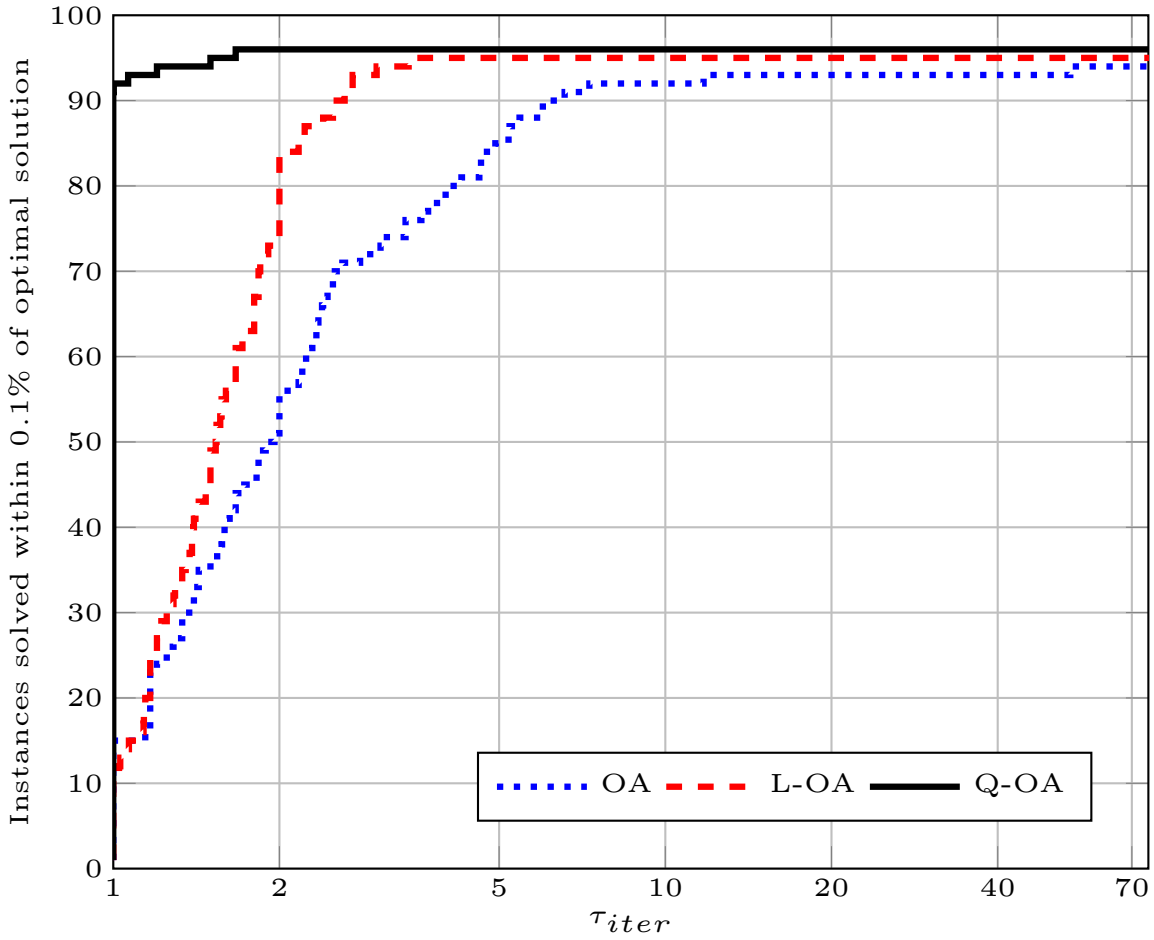


Figure 6.8: Iterations performance profiles for test problems using second-oder regularization methods in OA.

Table 6.2: Number of instances solved and comparison in solution time and iterations of OA, L-OA, and Q-OA.

Method	Instances solved	Less time than OA	Fewer iterations than OA	Less time than L-OA	Fewer iterations than L-OA	Less time than Q-OA	Fewer iterations than Q-OA
OA	94 / 109	-	-	61 / 94	23 / 94	38 / 94	3 / 94
L-OA	95 / 109	34 / 95	57 / 95	-	-	9 / 94	2 / 94
Q-OA	96 / 109	57 / 96	80 / 96	86 / 96	84 / 96	-	-

than OA, and the profiles show a clear reduction in terms of iterations for both L-OA and Q-OA.

Given that the performance profiles show the results without distinguishing the individual instances, we include Table 6.2, which shows a direct comparison of the methods in terms of solution time and iterations. Note that Q-OA is able to solve 1 and 2 instances more than L-OA and OA, respectively.

None of the methods were able to find a solution within 0.1% of optimality gap for 13 instances in the test set. When comparing the proposed methods to OA, we see that L-OA is able to reduce the solution time in 36% of the instances and the iterations in 60%, while Q-OA reduced the time in 59% of the instances and the iterations in 83%. From the results, it was also noticed that the benefits of Q-OA are more apparent for the more challenging instances. By comparing the two proposed methods we see that Q-OA solves 90% of the instances in less time and 87.5% of the instances in fewer iterations than L-OA. Detailed solution information for all instances and methods can be found in the supplemental material.

An interesting result is that the proposed methods significantly decreased the number of infeasible NLP subproblems found while solving the selected problems. Using OA we obtained 877 infeasible NLP subproblems, while using L-OA and Q-OA we only obtained 259 and 257, respectively. This can be explained by the fact that the integer combinations are chosen closer in the search space to the best feasible solution, and information about the

curvature is utilized with the proposed methods. Choosing an integer combination close to a feasible solution also results in trial solutions close to the feasible region, which resulted in fewer infeasible trial solutions.

The solutions reported here were obtained using the level parameter $\alpha = 0.5$. We performed several tests varying the value of α, which resulted in significant changes for individual instances but rather insignificant when considering the whole test set. Overall, $\alpha = 0.5$ gave us the best results for this set of test problems.

6.7 Conclusions and future work

We have presented two new methods for solving convex MINLP problems, based on a regularization technique and a second order approximation of the Lagrangean. We have proven that both methods converge to the global optimal solution in a finite number of iterations, and shown that the proofs hold even if the MIQP subproblem is only solved approximately. Both methods are mainly intended for problems with moderate to high degrees of nonlinearity, and for such problems, both methods performed better than the original OA method. The new method called Q-OA required significantly fewer iteration than the original OA, and there was also a clear advantage in the solution time. The advantage is due to the fact that more information is utilized when choosing the integer combinations. The method L-OA uses a regularization technique which we showed is equivalent to using a trust region. The regularization prevents large jumps between iterations and tries to keep the trial solutions close to the feasible region, and for the test problems, it gave an advantage over the original OA. For the test problems, Q-OA performed better than the other methods, both with respect to the number of iteration and time. Furthermore, using Q-OA we were able to solve a larger percentage of the test problems within the time limit.

As future work we plan to implement the methods in a more efficient and flexible framework, *e.g.,* within an MINLP solver like DICOPT[105] or as part of a Toolkit in an optimization modeling software such as Pyomo or JuliaOpt. It could also be worth to investigate a dynamic update of the level parameter α. For example, it could be possible to adjust the parameter based on the current optimality gap. Another idea would be to investigate if the concepts used within L-OA and Q-OA could be effectively integrated within the framework of the NLP/LP based branch and bound algorithm presented in [90].

Chapter 7

Alternative Regularizations for Outer-approximation

7.1 Introduction

Optimization problems whose objective and constraints can be represented by algebraic linear and nonlinear functions of both continuous and discrete variables are commonly referred to as mixed-integer nonlinear programs (MINLP). MINLP is a highly versatile modeling paradigm, allowing even Universal Turing Machines to be encoded via a Minsky's register machine [3]. There is a large variety of practical applications and optimization tasks that can be modeled using MINLP, see *e.g.,* [6, 60, 106, 251].

Although MINLPs are non-convex optimization problems because of some variables' discreteness, the term convex MINLP is used to denote problems where the feasible region described by the constraints and the objective function are convex [26]. Convex MINLP problems are an important class of problems, as the convex properties can be exploited to derive efficient decomposition algorithms. These decomposition algorithms rely on the solution of different subproblems derived from the original MINLP. Among these decomposition algorithms for MINLP, we have Branch & Bound (B&B) [27], Generalized Benders Decomposition [28], Outer-approximation (OA) [30], Partial Surrogate Cuts [90], Extended Cutting Plane (ECP) [118], Feasibility Pump [143, 263] Extended Supported Hyperplanes (ESH) [121], and the center-cut [145] method. Moreover, the OA method has been

extended in several pieces of work, such as the single-tree OA [90], Quadratic-cuts OA [235], conic-based OA [44], Decomposition-based OA [264], and Proximal OA [265] methods. Most of these methods exploit the properties of convex MINLP to derive linearizations of the nonlinear constraints based on their gradients. These linearizations are equivalent to first-order Taylor expansions of the nonlinear inequalities. They define a linear region that overestimates the problem's nonlinear feasible region because of the convexity property.

OA has proven to be one of the most efficient algorithms for convex MINLP [26], and several state-of-the-art solvers build upon the OA method. Recent benchmarks [26, 266] have also shown good performance with so-called singe-tree search methods based on an OA approach. Single-tree methods only constructs a single B&B tree where the linear relaxation is dynamically updated, and these are implemented in several state-of-the-art solvers,*e.g.,* AOA [163], BARON [47], BONMIN [129], FilMint [128], SHOT [266], and Pajarito [44].

Methods such as OA, ECP, and ESH all rely on the successive solution of MILP relaxations for solving convex MINLP problems. Given that each MILP relaxation problem is solved via a B&B seach tree, these methods are known as multi-tree methods [44, 266]. With these methods, the linear relaxation is used in the same fashion as in Kelley's cutting plane method [119], *i.e.,* to derive the following trial solutions for either all variables or only the integer variables. Kelley's method, relying on the iterative solution on linear programming (LP) problems arising from the gradient-based linearizations at previous minimizers, is known to be unstable given its large jumps in the search space [253]. It has been proven that Kelley's cutting plane method has a poor complexity bound and is also not practically efficient at handling nonlinearities, see, *e.g.,* [252]. Stabilization techniques through regularization of the step-size and trust-region approaches [254, 267] have been proposed to tackle this shortcoming. In the continuous setting, the level bundle method [257,

258] has proven to work well in stabilizing cutting-plane methods for nonsmooth problems. This method derives the following trial solutions by projecting the current solution (or stabilization center) onto a specific level of the linearly approximated objective function.

Directly using a trust-region or regularization for convex MINLP is nontrivial as neighboring solutions can be far apart in the search space due to the discrete space. For nonsmooth convex MINLP Oliveira [256] proposed a regularized algorithm based on the ECP method. Combining OA and bundle methods, Delfino and Oliveira [268] derived a method for nonsmooth convex MINLP. Kronqvist, Bernal, and Grossmann [89] showed that using ideas from the level method makes it possible to integrate regularization and second-order derivatives in an OA framework efficiently. Using a second-order Taylor approximation of the Lagrangean within a level-based OA, the Q-OA method [89] significantly reduced the number of iterations for highly nonlinear convex MINLP problems.

In this paper, we build upon the work by Kronqvist, Bernal, and Grossmann [89] and present a general regularization framework for OA. We refer to the new method as Regularized Outer-approximation (ROA), which enables different regularization functions to be used while guaranteeing global convergence. We propose a set of regularization functions based on both distance metrics and the Lagrangean. The motivation behind the Lagrangean-based regularization functions is to incorporate more information from both the objective and constraint function.

Moreover, there has been a recent interest in algorithm developers for MINLP in solving these problems in a Branch & Cut scheme. First proposed by Quesada and Grossmann [90] in a method called LP/NLP B&B, the OA linearizations are added at every incumbent solution found while solving a single MILP problem using a B&B procedure. This method addresses a key weakness of the multi-tree methods, where MILP problems solved in each iteration are similar to one another, requiring repeated search effort. This method has been

further improved on several fronts. Leyffer [114] integrate it to a Sequential Quadratic Programming (SQP) for the NLP problems, Tawarmalani and Sahinidis [269] derive a Brnach & Cut algorithm based on polyhedral relaxations of non-convex functions for global optimization to implemente the global solver BARON, which was later improved upon by Khajavirad and Sahinidis [47] who incorporate techniques to derive valid linearizations for non-convex constraints, and Coey, Lubin, and Vielma [44] take advantage of conic programming techniques to provide certificates for convex mixed-integer programs. This idea, denoted as single-tree approach, has been implemented by several MINLP solvers such as BONMIN [129], FilMint [128], AOA [163], SHOT [266], Pajarito [44], and BARON [47]. We also integrate the regularization framework with the single-tree search algorithm in a method we denominate as regularized LP/NLP (RLP/NLP).

7.1.1 Contributions and outline

In this paper, we propose a general framework for integrating regularization mixed-integer subproblems in the OA method in the multi-tree and single-tree setting for solving convex MINLP problems. We prove that these methods are guaranteed to converge to the optimal solution of MINLP problems, regardless of the choice of regularization function. Seven different regularization functions are proposed as objectives in this work, three of them coming from distance metrics to the incumbent solutions, and the other four with approximations of the Lagrangean function around the best-found solution. We implemented these methods in the open-source **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy [190], making the methods readily available. With this implementation, we perform a comprenhensive computational study by solving all convex MINLP problems available in the benchmark library MINLPLib [100].

The remaining chapter is organized as follows. In Section 7.2 we provide the neces-

sary background on the OA and LP/NLP methods.Section 7.3 introduces the Regularized Outer-approximation (ROA) method and proposes the norm-based objective functions for the regularization subproblem. Next, we introduce four objective functions obtained through approximations of the Lagrangean function in Section 7.4. We provide a convergence analysis of the proposed methods in Section 7.5. The single-tree extension of the regularization method as the Regularization LP/NLP Branch & Bound (RLP/NLP) method and its implementation are presented in Section 7.6. Finally, the computational results of the methods' benchmarking in presented in Section 7.7.

7.2 Background

The MINLP problems considered in this paper are of the form,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} \quad & g_j(\mathbf{x},\mathbf{y}) \leq 0 \quad \forall j = 1,\ldots,l, \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m. \end{aligned} \tag{MINLP}$$

Later in the algorithms, the (nonlinear) objective function is transformed into a constraint by the epigraph formulation, $f(\mathbf{x},\mathbf{y}) \leq \mu$, where the continuous variable μ represents the objective value. To guarantee global convergence for OA-type algorithms typically requires convexity assumptions, a bounded search space, and some form of constraint qualification for problem MINLP [25, 30, 129]. Throughout this paper, we rely on the following assumptions:

Assumption 1. The nonlinear functions $f, g_1, \ldots, g_l : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ are convex and continuously differentiable.

Assumption 2. The linear constraints define a nonempty compact set.

Assumption 3. For each feasible integer combination $\mathbf{y}$, an integer combination such that there exist $\mathbf{x}$ variables for which the problem is feasible, a constraint qualification holds, *e.g.*, Slater's condition [226].

We begin by presenting the Outer-approximation method's main steps, on which the other algorithms build upon. The OA method uses a linear approximation (or relaxation) of the MINLP problem to obtain trial solutions for the integer variables and derives improving lower bounds on the optimal objective value. The linear approximation at each iteration is refined by using the previously obtained trial solutions $\left\{(\mathbf{x}^i, \mathbf{y}^i)\right\}_{i=0}^{k}$ as expansion points for first-order Taylor approximations of the nonlinear constraints

$$\begin{aligned} f(\mathbf{x}^k, \mathbf{y}^k) + \nabla f(\mathbf{x}^k, \mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x} - \mathbf{x}^k \\ \mathbf{y} - \mathbf{y}^k \end{bmatrix} \leq \mu, \\ g_j(\mathbf{x}^k, \mathbf{y}^k) + \nabla g_j(\mathbf{x}^k, \mathbf{y}^k)^\top \begin{bmatrix} \mathbf{x} - \mathbf{x}^k \\ \mathbf{y} - \mathbf{y}^k \end{bmatrix} \leq 0 \quad \forall j \in \mathcal{I}_k, \end{aligned} \tag{7.1}$$

forming a polyhedral outer approximation of the feasible set of problem MINLP. The linear inequality constraints in (7.1) are often referred to as cuts, as they refine the outer approximation by cutting off infeasible parts of the search space.

Using an accumulation of cuts given by (7.1) over k iterations, an approximation of the nonlinear constraints, the next integer combination $\mathbf{y}^{k+1}$ is obtained by solving the following

MILP problem

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu} \quad & \mu \\
\text{s.t.} \quad & f(\mathbf{x}^i,\mathbf{y}^i) + \nabla f(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq \mu \quad \forall i = 1,\dots,k, \\
& g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq 0 \quad \forall i = 1,\dots k, \forall j \in \mathcal{I}_i, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R},
\end{aligned}
\tag{OA-MILP}
$$

which is often referred to as the master problem. Here $\mathcal{I}_i$ are index sets containing the indices of the nonlinear constraint active at the trial solution $(\mathbf{x}^i,\mathbf{y}^i)$ [25]. From the convexity assumption, it is clear that the feasible set is overestimated and that the objective function will be underestimated, see, *e.g.*, [30]. Therefore, the optimum of problem OA-MILP provides a valid lower bound (LB) to the MINLP problem, referred to as LB^{k+1}, and the minimizer gives a new integer combination $\mathbf{y}^{k+1}$. Next, the corresponding continuous variables $\mathbf{x}^{k+1}$ are determined by solving the following convex NLP subproblem,

$$
\begin{aligned}
\min_{\mathbf{x}} \quad & f(\mathbf{x},\mathbf{y}^{k+1}) \\
\text{s.t.} \quad & g_j(\mathbf{x},\mathbf{y}^{k+1}) \leq 0 \quad \forall j = 1,\dots l, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y}^{k+1} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n,
\end{aligned}
\tag{NLP-I}
$$

which is the original MINLP problem with all the integer variables fixed. In case problem NLP-I is feasible, then $\mathbf{x}^{k+1}$ is given by the minimizer and $f(\mathbf{x}^{k+1},\mathbf{y}^{k+1})$ gives a valid upper bound (UB) UB^{k+1} to the MINLP problem. If problem NLP-I is infeasible, then the current integer combination is infeasible for all feasible values of the continuous variables.This

situation can be handled by solving a feasibility problem, that typically minimizes a norm of the constraint violation $\mathbf{s}$ with the current choice of $\mathbf{y}$ variables as follows,

$$\begin{aligned} \min_{\mathbf{x},\mathbf{r}} \quad & \|\mathbf{s}\|_p \\ \text{s.t.} \quad & g_j(\mathbf{x}, \mathbf{y}^{k+1}) \leq s_j \quad \forall j = 1, \dots l, \\ & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y}^{k+1} \leq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n,\ \mathbf{s} \in \mathbb{R}^l_+. \end{aligned} \tag{NLP-f}$$

Common choices of the norm for the constraint violations are the ℓ_∞ and the ℓ_1 norms.Solving the feasibility problems yields the values of the continuous variables $\mathbf{x}^{k+1}$. Notice that in this case $(\mathbf{x}^{k+1}, \mathbf{y}^{k+1})$ is not a feasible solution; therefore, it does not provide an UB on the optimal objective. Problem NLP-f always satisfies Slater's condition, and due to Assumptions 1 and 2, it is always feasible and tractable.

In case the difference between the UB and LB is not within the desired tolerance, the procedure is repeated in the next iteration, and the outer approximation in problem OA-MILP is improved by including new cuts. With the new cuts, the master problem returns a new integer combination and an improved LB. Due to convexity, the cuts will not exclude any feasible solutions from the search space [227]. However, the cuts are sufficient to ensure that the integer combination $\mathbf{y}^{k+1}$ will not be obtained in a consecutive iteration unless it is the optimal integer solution. Due to Assumption 2, the search space only contains a finite number of different integer combinations. Finite convergence follows from the fact that each iteration either finds a new integer combination or verifies optimality. For more details of OA, see [25, 30, 124]. In the Appendix, the pseudo-code presented in Algorithm 10 summarizes the OA algorithm's main steps.

Every iteration of the OA algorithm solves a new MILP problem OA-MILP. Note that the master problem solved in iteration k only differs from the one in iteration $k-1$ by the cuts added in that iteration. Solving each one of the MILP master problems can be

computationally challenging. To avoid solving a large number of similar MILP problems, Quesada and Grossmann [90] proposed the LP/NLP-based B&B algorithm that combines OA and B&B. The LP/NLP-based B&B algorithm dynamically updates the master problem and only builds a single B&B tree. Each node, or leaf, of the search tree forms a continuous linear programming (LP) problem where the integer variables are relaxed as continuous, and the cuts in (7.1) are used to approximate the nonlinear constraints. Integer solutions are obtained through branching on the LP problems. Once an integer solution is found in the search tree, it is used as a new integer combination in the OA algorithm resulting in new cuts by solving the corresponding NLP subproblem. The best-found feasible solution to the original problem is known as the incumbent solution. It provides the UB used in the search tree. The new cuts, derived from the new integer combination, are added to all open nodes of the B&B tree, and the linear B&B procedure continues with an improved approximation of the nonlinear constraints. Nodes are pruned as usual in a B&B method: if it becomes infeasible or its objective value exceeds the current UB. The LP/NLP-based B&B technique is also known in the literature as single-tree OA [44, 266] to differentiate it from the traditional OA methods, where each problem OA-MILP is solved individually and sequentially through its own B&B tree, hence the name multi-tree OA.

The main steps in the LP/NLP B&B are outlined in Algorithm 11 in the Appendix.

We consider the following illustrative example to highlight the features of the presented

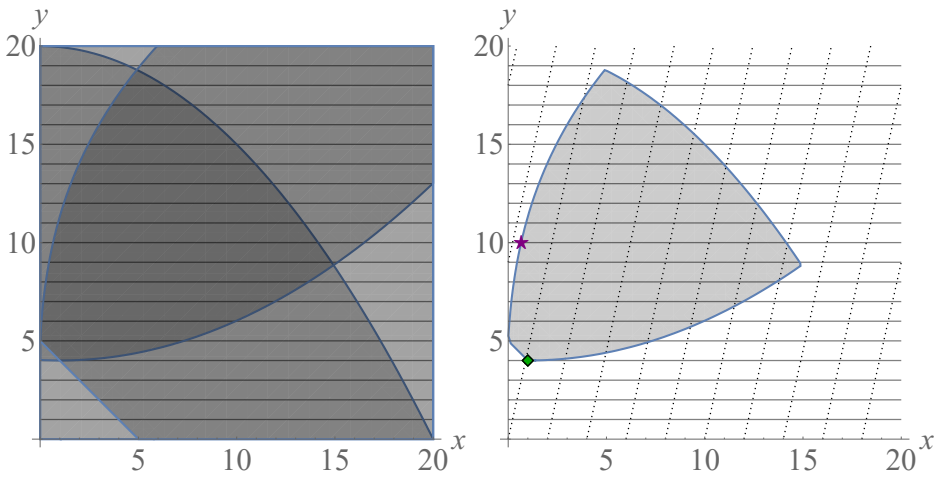


Figure 7.1: Left: Feasible region of problem Ex 1. Right: integer relaxed feasible region, optimal solution of the problem (★), initialization point (◆), and the contours of the objective.

methods and show how they differ from OA.

$$\begin{aligned} \text{minimize} \quad & x - y/4.5 + 2 \\ \text{s.t.} \quad & x^2/20 + y \le 20 \\ & (x-1)^2/40 - y \le -4 \\ & 0.275y^{1.5} - 10(x+0.1)^{0.5} \le 0 \\ & 0 \le x \le 20, \quad 0 \le y \le 20, \quad x \in \mathbb{R},\ y \in \mathbb{Z}. \end{aligned} \tag{Ex 1}$$

The basic features of problem Ex 1 are illustrated in Figure 7.1, showing the constraints, objective, and the optimal solution $(x^\star, y^\star) = (0.65625, 10)$.

We use the feasible point, $(x^0, y^0) = (1, 4)$, as the starting point for all the methods instead of solving the continuous relaxation. OA requires five iterations to solve this problem, of which the first four iterations are shown in Figure 7.2. In this specific problem, the first iteration results in an infeasible solution. The optimal solution is obtained in iteration four, and verifying optimality requires an additional iteration.

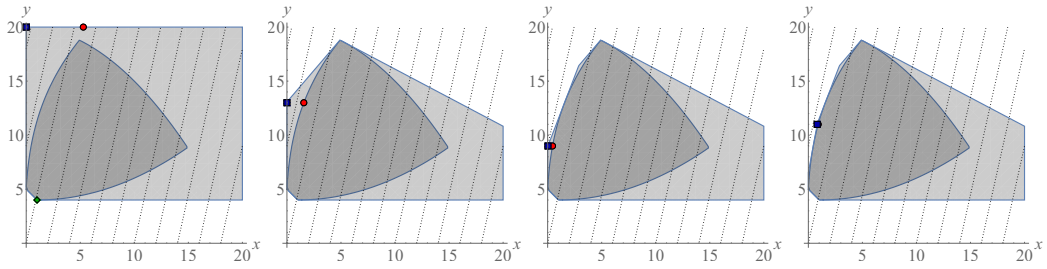


Figure 7.2: Progress of OA in problem Ex 1, with each figure being an iteration. The feasible region defined by the nonlinear constraints (dark gray), the outer approximation obtained by the generated cuts (light gray), the MILP master problem solution (■), and NLP subproblem solution (●) are included.

7.3 Regularized Outer-approximation

The level-based OA (L-OA) method was presented by Kronqvist, Bernal, and Grossmann [89], where the authors used a squared ℓ_2-regularization to the subproblem of obtaining new integer assignments. It was shown in the paper that the regularization technique is equivalent to adding a trust region, given by squared ℓ_2-norm, with a center at the incumbent solution. We give a brief overview of the L-OA algorithm since the other regularization techniques in this paper are also based on this framework. For more details, we refer to [89].

At iteration k, the master problem OA-MILP is solved to obtain a LB LB^k on the optimal objective value. Given an incumbent solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ and the UB resulting from $f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, we estimate the optimal objective value of the MINLP problem $f^\star = f(\mathbf{x}^\star, \mathbf{y}^\star)$ as

$$\widehat{f_k^\star} = (1-\alpha) f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) + \alpha LB^k, \tag{7.2}$$

where $\alpha \in (0, 1]$. The estimated optimum $\widehat{f_k^\star}$ is chosen as an interpolation between the UB and LB, where α is the interpolation parameter representing how much the linear approximation, *i.e.*, the master problem, is trusted. For the continuous setting, within the level method proposed by Lemaréchal, Nemirovskii, and Nesterov [257], a value of

$\alpha = 1 - \sqrt{2}/2 \approx 0.29$ is found to be optimal. The proof does not generalize for the mixed-integer case, meaning that an ideal value for α is not known a-priori. As in [89], in this work we use $\alpha = 0.5$. The next integer assignment $\mathbf{y}^{k+1}$ is now determined by projecting $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ onto the $\widehat{f_k^\star}$ level set of the linearly approximated objective function intersected with the current outer approximation of the feasible set. The projected solution is obtained as the minimizer of the following MIP problem,

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu} \quad & \phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) \\
\text{s.t.} \quad & \mu \leq \widehat{f_k^\star} \\
& f(\mathbf{x}^i,\mathbf{y}^i) + \nabla f(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq \mu \quad \forall i = 1,\dots,k, \\
& g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq 0 \quad \forall i = 1,\dots,k, \forall j \in \mathcal{I}_i, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m, \mu \in \mathbb{R},
\end{aligned}
\tag{MIP-Proj}
$$

where $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is a convex regularization function represent by the symbol h. The L-OA algorithm in [89] use the regularization function

$$
\phi^{\ell_2^2}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) := \left\| \begin{bmatrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{bmatrix} \right\|_2^2, \tag{7.3}
$$

and the authors mention that the convergence guarantees of the algorithm are independent of the choice of objective function in MIP-Proj. The regularization problem MIP-Proj must contain all the cuts accumulated in problem OA-MILP to ensure convergence. The regularization role is to favor solutions close to the incumbent solution with regards to a specific metric. The new integer assignment $\mathbf{y}^{k+1}$ is chosen as a point as close as possible to the incumbent solution, such that the linearly approximated objective is reduced to at

most $\widehat{f_k^\star}$. By construction, the regularization problem MIP-Proj is always feasible, *e.g.*, the minimizer of problem OA-MILP will satisfy all the constraints, and it is used to derive the next integer assignment $\mathbf{y}^{k+1}$. Once the new integer combination is obtained, the corresponding continuous variables can be determined using the same technique as in the OA method. The difference between the L-OA and OA methods is how the new integer assignments are obtained. Otherwise, both methods use the same techniques for determining the continuous variables and improving the outer approximation of the feasible set.

Since finite convergence of L-OA holds for any objective function in the regularization problem [89], other regularization techniques can easily be incorporated into the L-OA framework. A general framework based on the L-OA concept, where the regularization function is not specified, is summarized as a pseudo-code in Algorithm 8. We refer to this algorithm as Regularized Outer-approximation (ROA).

Two alternative regularization functions that fits directly into the L-OA are

$$\phi^{\ell_1}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) := \left\| \begin{matrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{matrix} \right\|_1 , \tag{7.4}$$

$$\phi^{\ell_\infty}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) := \left\| \begin{matrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{matrix} \right\|_\infty . \tag{7.5}$$

A benefit of using a regularization based on either the ℓ_1-norm or ℓ_∞-norm is that the regularization problem can be encoded as a MILP problem. We define for the remaining of the paper the L-OA approach from [89] as ROA-ℓ_2^2, and the proposed linear regularization approaches that use (7.4) and (7.5) as regularization functions as ROA-ℓ_1 and ROA-ℓ_∞, respectively.

As shown in [89], L-OA finds the same integer solutions as the master problems in OA with specific trust-region constraints. In fact, the equivalence to a trust region still holds

Algorithm 8 An algorithm summarizing the Regularized Outer-approximation (ROA) method.

Define accepted optimality gap $\epsilon \geq 0$, the regularization function $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$, and choose the parameter $\alpha \in (0, 1]$.

1. Initialization.
   - 1.1 Obtain a relaxed solution $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ by solving an integer relaxation of the MINLP problem.
   - 1.2 Generate cuts at $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ according to (7.1) and construct problems OA-MILP.
   - 1.3 Set iteration counter $k = 1$, $UB^0 = \infty$ and $LB^0 = -\infty$.
2. Repeat until $UB^{k-1} - LB^{k-1} \leq \epsilon$.
   - 2.1 Solve problem OA-MILP to obtain $\mathbf{y}^k$ and LB^k.
   - 2.2 If a feasible solution $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ has been found, calculate the estimated optimal value $\widehat{f^\star_k}$ according to (7.2) and solve problem MIP-Proj to update $\mathbf{y}^k$.
   - 2.3 Solve problem NLP-I with integer variables fixed as $\mathbf{y}^k$ to obtain $\mathbf{x}^k$.
     - 2.3.1 If problem NLP-I is feasible, set $UB^k = \min\{f(\mathbf{x}^k, \mathbf{y}^k), UB^{k-1}\}$.
       - 2.3.1.1 If $f(\mathbf{x}^k, \mathbf{y}^k) \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, set $\bar{\mathbf{x}}, \bar{\mathbf{y}} = \mathbf{x}^k, \mathbf{y}^k$.
     - 2.3.2 If problem NLP-I is infeasible, obtain $\mathbf{x}^k$ by solving feasibility problem NLP-f and set $UB^k = UB^{k-1}$.
   - 2.4 Generate cuts at $\mathbf{x}^k, \mathbf{y}^k$ according to (7.1) and add these to problems OA-MILP and MIP-Proj.
   - 2.5 (Optional) Generate no-good cuts at $\mathbf{y}^k$ and add these to problems OA-MILP.
   - 2.5 Increase iteration counter, $k = k + 1$,
3. Return $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ as the optimal solution $\mathbf{x}^\star, \mathbf{y}^\star$.

with the regularization given by any p-norm. This property is stated in Theorem 7.3.1. The proof uses the same argumentation as in [89] but is included for the sake of completeness.

Theorem 7.3.1. With the regularization given by a p-norm, the procedure of solving problems OA-MILP and MIP-Proj in ROA results in solution equivalent to adding the trust region constraint

$$\left\| \begin{matrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{matrix} \right\|_p \leq r_k \tag{7.6}$$

to problem OA-MILP in OA, where r_k is chosen as the optimum of problem MIP-Proj.

Proof. As mentioned earlier, MIP-Proj is always feasible and and we denote the minimizer by $\mathbf{x}^{\text{MIP-Proj}}, \mathbf{y}^{\text{MIP-Proj}}, \mu^{\text{MIP-Proj}}$. The radius of the equivalent trust region constraint is then given by

$$r_k = \left\| \begin{matrix} \mathbf{x}^{\text{MIP-Proj}} - \bar{\mathbf{x}} \\ \mathbf{y}^{\text{MIP-Proj}} - \bar{\mathbf{y}} \end{matrix} \right\|_p \tag{7.7}$$

Solving problem OA-MILP, with the trust region constraint, gives the solution $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$. Now, assume this solution is not an optimal solution to problem MIP-Proj. Since $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$ is not an optimal solution, it follows that

$$r_k > \left\| \begin{matrix} \mathbf{x}^{\text{MILP}} - \bar{\mathbf{x}} \\ \mathbf{y}^{\text{MILP}} - \bar{\mathbf{y}} \end{matrix} \right\|_p . \tag{7.8}$$

Since OA-MILP minimizes μ, we know that $\mu^{\text{MILP}} \leq \mu^{\text{MIP-Proj}} \leq \widehat{f_k^\star}$. This leads to a contradiction since $\mathbf{x}^{\text{MILP}}, \mathbf{y}^{\text{MILP}}, \mu^{\text{MILP}}$ is a feasible solution to problem MIP-Proj with an objective value strictly lower than the solution obtained by solving the minimization problem. Therefore, the solution to problem OA-MILP, with the trust-region constraint, must also be an optimal solution to problem MIP-Proj. □ □

Depending on which function $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$ is used in the ROA method, we obtain different variants of the algorithm. These variants are denoted as ROA-h, *e.g.*, we refer to ROA-ℓ_1 when (7.4) is used as the objective for the regularization problem. Next, we illustrate the difference between these variants with example Ex 1.

For example Ex 1, the three level regularization norms presented here; ROA-ℓ_2^2, ROA-ℓ_1, and ROA-ℓ_∞; converge to the optimal solution in three iterations. Thanks to the regularization, all three approaches find the optimal solution in the first iteration as observed in Figure 7.3. Although the simple example does not show this behavior, the regularization objective's choice might affect which integer combination gets chosen to solve problem NLP-I.

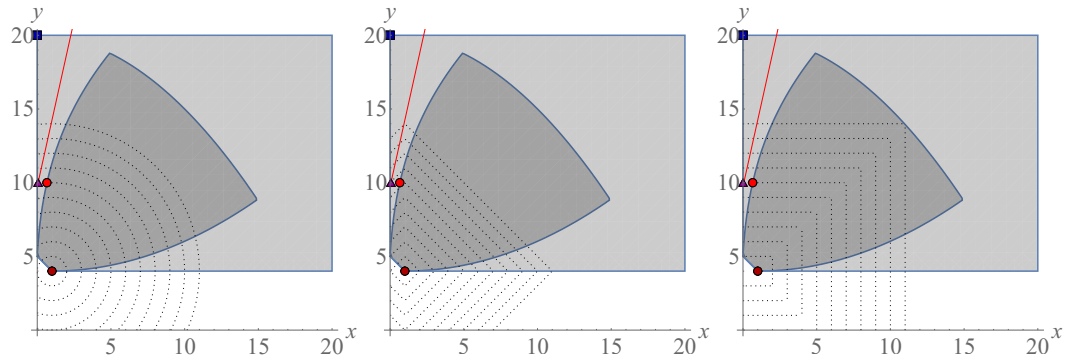


Figure 7.3: First iteration of ROA for problem Ex 1 with the three level norms presented in this work. The format from Figure 7.2 is used here, with the additional features that the regularization objective contours, the regularization problem solution (▲), the incumbent solution (●), and the level constraint (7.2) with $\alpha = 0.5$ (red line) are included. Left: ROA-ℓ_2^2. Center: ROA-ℓ_1. Right: ROA-ℓ_∞.

In every case, the regularization tries to keep this integer combination close to the incumbent solution. Moreover, the choice of the objective may impact the computational time required to solve the regularization problem. As mentioned above, choosing the squared ℓ_2 norm as in L-OA [89] leads to the regularization problem becoming an MIQP. On the other hand, the ℓ_1 and ℓ_∞ norms in the objectives can be modeled using linear inequalities and auxiliary variables, as presented in the Appendix, leading to MILP regularization subproblems. For all approaches, it takes two more iterations to close the LB.

In the next section, we present two new regularization strategies that also fit within the ROA framework and incorporate information from the Lagrangean function.

7.4 Lagrangean based regularization

To take advantage of second-order derivatives for selecting the new integer assignment $\mathbf{y}^{k+1}$, Kronqvist, Bernal, and Grossmann [89] proposed a technique they refer to as Quadratic Outer-approximation (Q-OA). Instead of a regularization function, Q-OA uses a second-

order Taylor series expansion of the Lagrangean function as the objective function in MIP-Proj. Thus, the new integer assignment is chosen by minimizing a quadratic approximation of the Lagrangean within an outer approximation of the feasible set subject to a level constraint, *i.e.*, $\mu \leq \widehat{f_k^\star}$. Except for the level constraint, there is an apparent similarity with the sequential quadratic programming (SQP) approach [270].

The Lagrangean function $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$ associated with the MINLP problem can be written as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}, \lambda) = f(\mathbf{x}, \mathbf{y}) + \lambda^\top \widetilde{g}(\mathbf{x}, \mathbf{y}), \tag{7.9}$$

where $\widetilde{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ contains all the constraints in the form $\widetilde{g}(\mathbf{x}, \mathbf{y}) \leq 0$, linear and nonlinear. From the fixed NLP problem giving the incumbent solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, the corresponding dual variable $\bar{\lambda}$ is also obtained. Now, by defining the regularization function $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$ as

$$\phi^{\mathcal{L}_2}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x}, \mathbf{y}) := \nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})^\top \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}^\top \nabla^2_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda}) \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}, \tag{7.10}$$

the ROA method in Algorithm 8 will result in the Q-OA algorithm. Note that $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$ in (7.10) can be considered a regularizer with a stabilization center at the minimizer of the quadratic approximation of the Lagrangean. In case the Hessian of the Lagrangean is not positive definite (only positive semidefinite), the stabilization center may not be a unique point but a subspace.

Remark 7.4.1. With the integer variables fixed as $\bar{\mathbf{y}}$, the point $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})$ is a stationary point of the Lagrangean and, therefore, all partial derivatives corresponding to the continuous variables will be zero in $\nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})$. This follows directly from the KKT conditions of the NLP problem NLP-I.

Next, we derive two new regularization functions based on the Lagrangean that can be directly implemented in Algorithm 8. Using the Hessian of the Lagragian, we can define a

norm as

$$\left\| \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \right\|_{\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})} := \sqrt{\begin{bmatrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{bmatrix}^\top \nabla^2_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda}) \begin{bmatrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{bmatrix}}, \tag{7.11}$$

which is a proper norm if in the Hessian is positive definite or a semi norm if the Hessian is positive semidefinite [271]. Based on this (semi) norm, we define a new regularization function as

$$\phi^{\nabla^2\mathcal{L}}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) := \left\| \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \right\|^2_{\mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})} = \begin{bmatrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{bmatrix}^\top \nabla^2_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda}) \begin{bmatrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{bmatrix}. \tag{7.12}$$

This regularization function's motivation is to favor search directions in which the Lagrangean has a locally linear behavior. This regularization, therefore, favors regions of the search space where the outer approximation is expected to be more accurate. In case the Hessian has at least one zero eigenvalue, the trust-region will be unbounded in directions in which the quadratic approximation of the Lagrangean changes linearly.

There are situations in which the Hessian is not known or too expensive to compute. One of the simplest approximations of the Hessian is a scaled identity matrix ρI. The BFGS algorithm [272], for example, uses the scaled identity as the first estimate of the Hessian of the Lagrangean. Using this trivial approximation of the Hessian in the quadratic approximation of the Lagrangean, gives us the following regularization function

$$\phi^{\mathcal{L}_1/\ell_2^2}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y}) := \nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}},\bar{\mathbf{y}},\bar{\lambda})^\top \begin{bmatrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{bmatrix} + \rho \begin{bmatrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{bmatrix}^\top I \begin{bmatrix} \mathbf{x}-\bar{\mathbf{x}} \\ \mathbf{y}-\bar{\mathbf{y}} \end{bmatrix}, \tag{7.13}$$

where $\rho \in \mathbb{R}_+$ is a scaling factor. This gives a regularization function with a stabilization center shifted in the direction of the negative gradient of the Lagrangean. Since the gradient is zero for all the continuous variables, the stabilization center is only shifted for the discrete variables. The stabilization center $(\mathbf{x}_c, \mathbf{y}_c)$ can easily be determined from the stationary

conditions of the regularization function, and is given by

$$\begin{bmatrix} \mathbf{x}_c \\ \mathbf{y}_c \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{x}} \\ \bar{\mathbf{y}} \end{bmatrix} - \frac{\nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})}{2\rho}. \tag{7.14}$$

Depending on the magnitude of both $\nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})$ and ρ, the stabilization center might be far from the incumbent solution and even outside of the variable bounds. However, we can directly control how far from the incumbent solution the stabilization center lies by scaling ρ. By selecting ρ as

$$\rho = \left\| \frac{\nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})}{2d} \right\|_2, \tag{7.15}$$

the euclidean distance between the stabilization center and the incumbent solution becomes d. We can, thus, use the parameter d to determine how far the stabilization center is shifted.

If we completely remove the quadratic term from the Lagrangean approximation, we are left with the linear approximation function

$$\phi^{\mathcal{L}_1}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x}, \mathbf{y}) := \nabla_{\mathbf{x},\mathbf{y}} \mathcal{L}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda})^\top \begin{bmatrix} \mathbf{x} - \bar{\mathbf{x}} \\ \mathbf{y} - \bar{\mathbf{y}} \end{bmatrix}. \tag{7.16}$$

Note that function (7.16) will not result in a regularization in ROA! However, since the linear approximation function combines the gradients of both the constraints and the objective, it could provide a direction more favorable for finding feasible solutions. Based on the computational results in Section 7.7, we observe that using (7.16) as the objective function in the regularization subproblem is not advantageous compared to the other approaches presented in this paper; supporting the use of a regularizer.

Similarly to the level-based approaches, we use the following notation for the regularization methods derived using the Lagrangean: The Q-OA method presented in [89] is presented as ROA-$\mathcal{L}_2$. We denote ROA-$\mathcal{L}_1$ the method using the first-order approximation of the Lagrangean as in (7.16), and following that notation the regularization methods involving (7.12) and (7.13) are denoted ROA-$\nabla^2\mathcal{L}$ and ROA-$\mathcal{L}_1/\ell_2^2$, respectively.

Next, we illustrate the differences between these regularization functions derived from the Lagrangean in ROA with example Ex 1. In Figure 7.4 we observe the first three iterations of the ROA methods with objective functions for problem MIP-Proj given by the second-order Taylor approximation, the Hessian of the Lagrangean based norm, and the first-order Taylor approximations of the Lagrangean function. Notice that the second-order Taylor approximation of the Lagrangean, proposed initially as Q-OA in [89], has a regularization objective equivalent to the sum of the two other methods presented in Figure 7.4. This can be observed as the contours of the regularization objective in the ROA-$\mathcal{L}_2$ method have a stabilization center (sometimes beyond the domain of the figure) specified by the ROA-$\nabla^2\mathcal{L}$ with a shift given by ROA-$\mathcal{L}_1$ corresponding objective in the direction of the discrete variable. This observation corresponds with Remark 7.4.1. Moreover, the gradient of the Lagrangean switches from pointing up or down depending on whether the incumbent solution is below or above the optimal solution, respectively. Although all the methods shown in Figure 7.4 can find the optimal solution following an infeasible first iteration, the number of iterations required to close the gap between UB and LB and guarantee optimality varies. It takes ROA-$\mathcal{L}_2$ five iterations, ROA-$\nabla^2\mathcal{L}$ six iterations, and ROA-$\mathcal{L}_1$ seven iterations to guarantee the optimality of the solution, after finding the optimal solution in the last iteration for the first method and in the second-to-last iteration for the other two methods.

The progress of the ROA-$\mathcal{L}_1/\ell_2^2$ method is shown in Figure 7.5. This Figure exemplifies how the ℓ_2^2 norm stabilization center is shifted from the incumbent solution in the direction of the gradient of the Lagrangean. This distance of the shifting is given by parameter d, equal to one in this example. It can be seen from the smallest contour, representing a regularization objective of zero, on which the incumbent solution lies. In terms of the number of iterations, ROA-$\mathcal{L}_1/\ell_2^2$ is the most efficient method among all the ones presented here at solving problem Ex 1, finding the optimal solution on its first iteration and closing

the gap between UB and LB in three iterations.

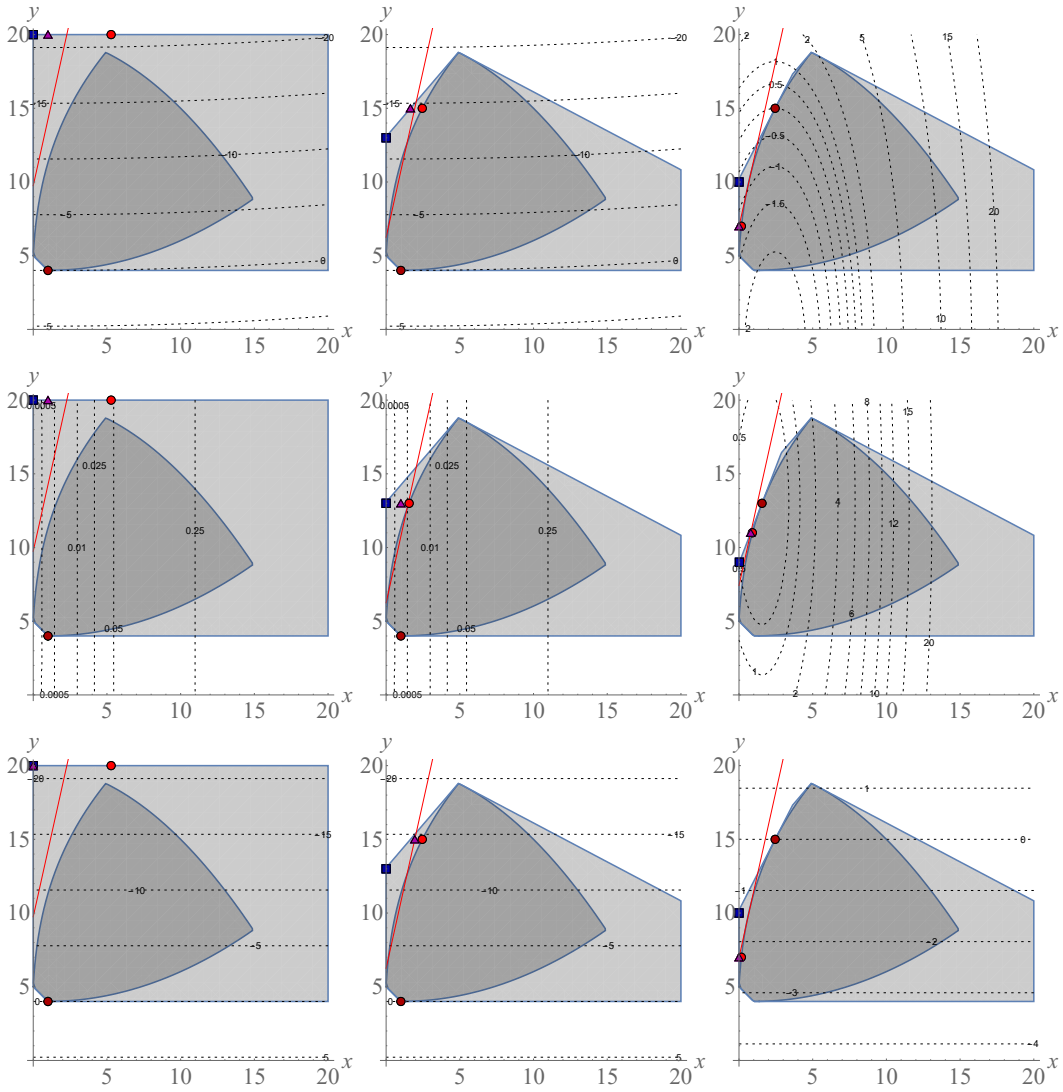


Figure 7.4: First three iterations (from left to right) for proposed Lagrangean-based regularization methods for problem Ex 1. The format from Figure 7.3 is used here. Top: ROA-$\mathcal{L}_2$. Center: ROA-$\nabla^2\mathcal{L}$. Bottom: ROA-$\mathcal{L}_1$.

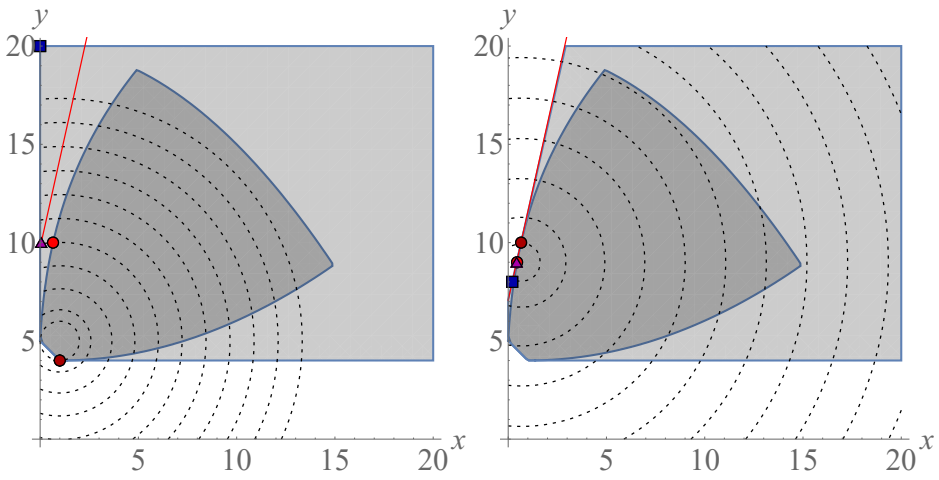


Figure 7.5: First two iteration of ROA-$\mathcal{L}_1/\ell_2^2$ for problem Ex 1. The format from Figure 7.3 is used here, considering the scaling factor ρ such that the shifting of the stabilization center is $d = 1$.

7.5 Convergence properties

The convergence proof of the L-OA algorithm presented in [89] is entirely independent of the objective function of the regularization problem MIP-Proj. Therefore, finite convergence of ROA, for any function $\phi^h_{\bar{\mathbf{x}},\bar{\mathbf{y}}}$, directly follows from the convergence proofs of L-OA. For completeness, we outline the main convergence property of ROA. For more details, we refer the reader to [89, Section 5].

From the start, we assumed that all the nonlinear functions are convex (Assumption 1). This assumption is crucial since it ensures that the ROA methods' cuts do not cut off any feasible integer solution and that the master problem OA-MILP gives a valid LB. For a complete proof that the master problem OA-MILP gives a valid LB, see [25, 30, 89]. With all the ROA methods, the regularization problem will be feasible in each iteration. The feasibility of the regularization problem is given by the fact that the constraints in the regularization problem MIP-Proj are the same as in OA-MILP, besides the reduction

constraint controlled by the confidence parameter α, for more details, see [89, Lemma 4]. As stated in [89, Lemma 3], it is clear that each infeasible integer combination obtained in the search will be excluded from the search space by the generated cuts. An essential property of the ROA methods is that, as long as the UB and LB of the optimal objective function are different, the regularization problem will provide a new integer combination in each iteration. This property is formally stated in the following theorem.

Theorem 7.5.1. If the lower bound is not equal to the upper bound, then the minimizer of regularization subproblem MIP-Proj provides a new integer combination.

Proof. By [89, Lemma 3], it is clear that each infeasible integer combination encountered will be excluded from the search space by the cuts generated in the ROA algorithm. As proven in [89, Theorem 5], all feasible integer combinations found by the ROA algorithm will also be excluded from the search space as long as there is a gap between the upper and lower bound. □ □

The main convergence property of ROA is summarized in the following theorem.

Theorem 7.5.2. The ROA algorithm will terminate after a finite number of iterations, either by proving the best-found solution's optimality or by verifying that the MINLP problem is infeasible.

Proof. By Theorem 7.5.1, it is clear that problem MIP-Proj will in each iteration find a new, previously unexplored integer assignment, as long as the UB is not equal to the LB. As stated in [89, Lemma 1], the LB is valid in each iteration of the algorithm. Due to Assumption 2, the search space only contains a finite number of different integer assignments. Therefore, the algorithm must terminate after a finite number of iterations with either the UB equal to the LB or by proving infeasibility by the master problem being infeasible. □ □

For more details and a complete convergence proof, we refer the reader to [89, Section 5].

7.6 Regularization in LP/NLP Branch & Bound algorithm

Solvers based on a single-tree search or LP/NLP-based B&B algorithms have shown outstanding performance in recent benchmarks [26, 266]. A natural extension of the regularization framework from the previous section is, thus, the integration of regularization in LP/NLP-based B&B. This is also suggested in the conclusions and future work section of [89].

To introduce regularization into the LP/NLP-based B&B framework, we use the regularization problem MIP-Proj for each node in the search tree where an integer feasible solution is found. The regularization problem intends to choose new integer combinations close to the incumbent solution. To ensure convergence, it is also necessary to generate cuts at the nodes' variable values in the search tree with integer feasible solutions. Otherwise, an infeasible integer combination encountered in the search tree might not be excluded as the regularization might result in a different integer combination. Except for these two modifications, the algorithm follows the same procedure as the standard LP/NLP-based B&B algorithm. The regularized LP/NLP-based B&B algorithm is summarized as a pseudo-code in Algorithm 9. We denote all the algorithms implementing regularization approaches on the LP/NLP-based B&B method as RLP/NLP. Similarly to ROA, depending on the objective function used in the regularization subproblem, $\phi^{h}_{\bar{\mathbf{x}},\bar{\mathbf{y}}}(\mathbf{x},\mathbf{y})$, we denominate the approach as RLP/NLP-h, *e.g.,* the single-tree approach using (7.5) is denoted RLP/NLP-ℓ_∞.

A significant difference compared to ROA is that the optimum of the linear approximation OA-MILP is not available during the search. This is an important detail and is described further in the following remark.

Remark 7.6.1. During the B&B tree search, the minimum of the current linear approximation OA-MILP is not known. Only a lower bound to problem OA-MILP is available, and all

integer feasible solutions to OA-MILP may have a larger objective function value. Using the available LB to calculate the estimated optimum $\widehat{f_k^\star}$ may, therefore, result in an infeasible level constraint, *i.e.,* there does not exist an integer feasible solutions to problem OA-MILP with an objective value less than or equal to $\widehat{f_k^\star}$. In such a situation, the regularization problem MIP-Proj will also be infeasible.

In case the regularization is infeasible, the RLP/NLP algorithm continues by using the integer combination obtained at the current node. Note that each integer feasible node of the search tree involves a possibly expensive regularization problem. Therefore, to be competitive, the additional regularization problem must significantly reduce the number of integer combinations explored. As shown in [89] the regularization can lead to a drastic reduction of iterations and explored integer combinations.

Since $\widehat{f_k^\star}$ is not necessarily a valid LB to the current linear approximation OA-MILP, the convergence proofs from the previous section do not hold. However, the convergence can still be guaranteed as cuts are generated for each node's variable values with an integer feasible solution. Therefore, the convergence is guaranteed due to the convergence of the ECP algorithm, see [128] for details of a single-tree ECP algorithm.

7.7 Computational results

This section introduces our implementation details of the seven ROA methods and analyzes their performance through benchmark tests. The OA method is selected as the baseline. As a general observation, we notice that the regularization methods are able to handle highly nonlinear convex MINLP problems more efficiently than OA. This aligns well with the observations of the two regularization methods in [89]. Using regularization methods induces a more careful choice of the integer combination to be evaluated, having

that trial solution to lie preferably close to the best found feasible solution. In general, these regularization methods favor the choice of the next integer combination close to an stabilization center. This center is constructed using the incumbent solution and the curvature of the constraints, using information from the Lagrangean of the problem or a p-norm. The choice of the new integer solution comes at the expense of solving a mixed-integer regularization subproblem. This extra step might become exorbitant for mostly linear instances, where tight outer-approximations of the nonlinear feasible region can be obtained with only a few gradient-based cuts.

To test the performance of the proposed methods, we use test instances from the problem library MINLPLib* [100]. There are 456 convex problems in MINLPLib, from which we select the 358 instances that have at least one discrete variable and at least one continuous variable. We denote the 358-instances set as Problem Set 1. Compared to the OA method, ROA methods use regularization to keep the trial solutions close to the incumbent solution and the feasible set. Favouring solutions close to the incumbent, also favours areas close to a linearization point, *i.e.,* areas where the outer approximation is accurate. In theory regularization-based methods lead to a higher efficiency gain with respect to OA in highly nonlinear instances. This has been corroborated experimentally [89]. Therefore, we establish Problem Set 2 with highly nonlinear instances selected from Problem Set 1 according to the following criterion:

$$\frac{n_{nonlin}}{n+m} > 0.4, \tag{7.17}$$

where n_{nonlin} is the number of variables present in some nonlinear term, and $m+n$ is the total number of discrete and continuous variables. There are in total 122 convex MINLP problems in MINLPLib that satisfy (7.17). The instances in Problem Set 1 have between 2 to 4530 variables and 0 to 5329 constraints, while the instances in Problem Set 2 range from 6

*Retrieved on Jul. 15 2019 from http://www.minlplib.org/

to 4530 variables and 0 to 4650 constraints.

7.7.1 Implementation details

The OA methods and seven ROA methods are implemented as part of the **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy [190]. This toolbox presents an open-source* implementation of several solution techniques for MINLP based on problem decomposition. Through a Python implementation relying on the algebraic modeling language Pyomo [273], MindtPy can easily access a wide range of solvers to address the subproblems arising from the decomposition. The methods implemented in MindtPy for the solution of convex MINLP include OA [30] and ECP [118]. These are complemented with other decomposition methods such as the feasibility pump [143, 263] and the center cut algorithm [145]. Besides, MindtPy includes an implementation of the LB/NLP B&B method [90]. Its flexible framework allows users to easily tailor the algorithm to fit their particular application *i.e.,* by using different initialization procedures, feasibility norms, cutting planes generators, and call-back procedures.

For the results presented herein, we use CPLEX 20.1.0.0 [52] as the solver for the MILP/MIQP subproblems and IPOPT 3.12 [158] for the NLP subproblems using the Harwell Subroutine Library (HSL) MA27 [274] as a solver for linear systems. The level parameter in ROA methods is set to $\alpha = 0.5$ for all approaches. Moreover, for ROA-$\mathcal{L}_1/\ell_2^2$ we use as shifting radius $d = 1$. We implement the multi-tree and single-tree approaches, described in Algorithms 8 and 9 respectively. We consider the zero tolerance for checking if a constraint is active as 10^{-8}; hence we set the IPOPT parameter `constr_viol_tol` to that value.

According to Theorem 7.5.2, it is not necessary to solve the regularization subproblems to optimality. As noted in [89], any feasible solution for the regularization problems is

*https://pyomo.readthedocs.io/en/stable/contributed_packages/mindtpy.html

sufficient to guarantee the convergence of the ROA methods. These regularization problems are of the same size as the master OA-MILP, and solving them might be a limiting factor in ROA, as observed by [89] previously. The performance was improved by not solving the regularization problem to optimality, by using the setting the MIP solution limit parameter, `mip limits solutions`, to 10. If CPLEX uses multiple threads, the number of MIP solutions found by CPLEX might slightly greater than the `mip limits solutions` given that if the limit is reached, the nodes being processed in other threads will not be interrupted. CPLEX will stop after all the current working threads are completed.

Since the problems we consider are all convex, the Hessian of the Lagrangean is always positive semidefinite, and the regularization subproblems are always convex. However, due to numerical accuracy, the regularization problem ended up nonconvex for a few cases, *e.g.*, the smallest eigenvalue of the Hessian was slightly negative. Therefore, we set the `optimalitytarget` parameter to 3 to enable CPLEX to solve nonconvex MIQPs in the ROA-$\mathcal{L}_2$ and ROA-$\nabla^2\mathcal{L}$ methods. Another approach to deal with the nonconvexities induced by numerical accuracy is to add small perturbations to the diagonal of the Hessian [89].

The solution procedure is initialized by solving the continuous relaxation of problem MINLP, which provides a valid LB of the optimal objective function. In each iteration k, the problem OA-MILP is initialized with the NLP subproblem solution in iteration $k-1$. Since the solution of OA-MILP is feasible to the problem MIP-Proj, we use its optimal value to initialize the regularization subproblems. Along this line, as problem NLP-I is solved for the integer combination of the regularization problem, we use the solution to MIP-Proj to initialize the nonlinear subproblems. All the other settings in MindtPy, CPLEX, and IPOPT are the same as the default.

As termination criteria, we use the standard criteria of both an absolute optimality

tolerance ϵ and a relative optimality tolerance ϵ_{rel}. The search is, thus terminated if either

$$f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - LB \leq \epsilon \quad \text{or} \quad \frac{f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) - LB}{|f(\bar{\mathbf{x}}, \bar{\mathbf{y}})| + 10^{-10}} \leq \epsilon_{rel}$$

are satisfied. All tests ran on an Intel® Xeon® CPU (24 cores) 2.67 GHz server with 128GB of RAM running Ubuntu. For the termination criteria, we set the tolerances $\epsilon = 10^{-5}$ and $\epsilon_{rel} = 10^{-3}$, and a time limit of $900s$. The multi-tree results are run with up to 8 threads, while the single-tree results are run with a single thread.

The LP/NLP-based B&B algorithm can be implemented through so-called call-backs to the MILP solver in the B&B process, both for solving the continuous nonlinear subproblems and adding new cuts to the LP problems in the B&B search. The initialization procedure is the same as in traditional OA to set up the first master problem. For each feasible integer solution $\widehat{\mathbf{y}}$ found in the B&B process, it is checked whether that specific integer combination has been found earlier in the search, *i.e.*, $\widehat{\mathbf{y}} \in \{\mathbf{y}^i | i = 1, \ldots, k-1\}$. If $\widehat{\mathbf{y}} \notin \{\mathbf{y}^i | i = 1, \ldots, k-1\}$, then the $\mathbf{x}^k$ variables can be obtained by solving NLP-I using that integer combination. If NLP-I is infeasible, the feasibility problem NLP-f is solved to determine the continuous variables. Cuts are generated according to (7.1) and added to all open nodes in the search tree, and practically implemented as lazy constraints. If $\widehat{\mathbf{y}} \in \{\mathbf{y}^i | i = 1, \ldots, k-1\}$, then there is no need to again solve NLP-I as the cuts already added for this integer combination is sufficient for the linear approximation to be tight for this integer combination [129]. This situation can, for example, occur if two different feasible solutions of the original problem only differ in the values of the continuous variables. This situation can be avoided by adding no-good cuts at every found solution [263].

7.7.2 Detailed examples

We first present detailed results for six particular instances of the selected test set. These problems were chosen to illustrate in detail the advantage of the ROA methods. The

selected instances are `cvxnonsep_normcon20`*, `cvxnonsep_psig40`*, `nvs11`*, `nvs12`*, `slay08m`*, and `smallinvDAXr1b150-165`* .

The statistics of these instances and their solution details are presented in Table 7.1. `slay08m` corresponds to the Big-M formulation of a safety layout problem, introduced in [275]. This instance is a Mixed-binary Quadratically Constrained Program (MBQP). `cvxnonsep_normcon20` and `cvxnonsep_psig40` are numerical instances proposed by Kronqvist, Lundell, and Westerlund [262]. The first one considers a single norm-2 constraint of 10 integer and 10 continuous variables. The second one minimizes a signomial function in terms of integer and continuous variables, making them a Mixed-integer Quadratically Constrained Program (MIQCP) and a general MINLP, respectively. The `cvxnonsep` instances are designed to be particularly difficult for OA type methods. `nvs11` and `nvs12` instances proposed by Gupta and Ravindran [108] that have been widely used for benchmarking MINLP solver, see *e.g.,* [276]. They contain only integer variables, and quadratic constraints and objective function; making them Integer Quadratically Constrained Quadratic Programs (IQCQP). `smallinvDAXr1b150-165` models an Extension of the Markovitz Mean-Variance-Optimization model by constraints for small investors. These problems belong to MIQCP.

To illustrate how the methods differ for these problems, we first show the upper and lower bounds obtained by each method in Figures 7.6 and 7.7. Each figure shows the percentage gap with the known optimal solution with respect to time and iterations. These plots have a semi-log vertical axis, where the values within $[-\epsilon_{rel}, \epsilon_{rel}]$ are presented in a

*http://www.minlplib.org/cvxnonsep_normcon20.html
*http://www.minlplib.org/cvxnonsep_psig40.html
*http://www.minlplib.org/nvs11.html
*http://www.minlplib.org/nvs12.html
*http://www.minlplib.org/slay08m.html
*http://www.minlplib.org/smallinvDAXr1b150-165.html

linear scale, while values beyond that are presented in a logarithmic scale.

Figure 7.6 shows the progress of the bounds as a function of time and iterations for problem `cvxnonsep_psig40` in the multi-tree setting. We observe that the UB is quickly reduced to the optimal solution by the regularization methods compared to OA, except for ROA-$\mathcal{L}_1$ corresponding to the previous observation that the gradient of the Lagrangean does not provide a stabilization center, hence performing worse than the other methods. This was the only approach unable to converge within the time limit in the multi-tree setting. The LB is then improved to reach convergence within the specified optimality tolerances. When observing the bounds progress with respect to the iterations, the difference is even more drastic, showing the positive effect of regularization for this problem.

The bounds profiles for all the presented methods through a single-tree implementation when solving problem `cvxnonsep_normcon20` are presented in Figure 7.7. Contrary to problem `cvxnonsep_psig40`, the optimal solution is found by all methods in the first iteration, leaving the remaining of the task to improve the LB until the gap is within the specified tolerance. Although the regularization problems address the UB improvement part of OA directly by providing integer combinations close to stabilization centers defined by the incumbent solutions, we see that they also favor the more efficient convergence of the LB. This effect is best observed in the bounds plot against the number of NLPs solved, considered for the single-tree as a measure of iterations. The regularization methods require only a fraction of the NLP subproblems to obtain a LB within the optimality tolerance. This difference is not as prominent in terms of computational time. However, for this problem, the most efficient regularization method ROA-ℓ_∞ reduces the run time by approximately a third.

Table 7.1 presents a more detailed view of the results for the different examples. Here we notice that, although the ROA method spends extra time to solve the regularization

problem, this eventually leads to a reduction in the total solution time compared to OA. Instance `cvxnonsep_normcon20` shows a positive effect of the regularization methods, where the number of infeasible NLP problems is drastically reduced from 175 and 20 in the multi-tree and single-tree cases, respectively, to zero in all regularization cases. This leads to an advantage of the regularization methods against OA for this instance. A similar situation happens with instances `nvs11` and `nvs12`, where all regularization methods reduce the number of infeasible NLPs compared to OA, with the exception of ROA-$\mathcal{L}_1$. This supports the notion that the gradient of the Lagrangean does not define a stabilization center in the regularization objective; therefore, it is not a regularization per se. Moreover, using the gradient of the Lagrangean as a regularization objective (7.16) fails to converge to the optimal solution of examples `slay08m` and `smallinvDAXr1b150-165` in the single-tree implementation and of `cvxnonsep_psig40` in the both single- and multi-tree implementations. However, OA converged in a little under 8 minutes. This highlights the advantages of flexible implementation that allow these different approaches to be simply activated or deactivated.

As a general observation, the regularization methods reduce the respective number of iterations (number of NLP subproblems) required for convergence compared to the case without a regularization. The time spent in solving those mixed-integer subproblems leads to reducing the total algorithm time in most cases. These methods also tend to more quickly find the optimal solutions, having a practical effect if the time limit exceeds the time required to solve the problems. Besides, finding reasonable feasible solutions early in the search leads to tighter linearizations to the polyhedral approximation of the continuously relaxed nonlinear feasible region, leading to a better LB and faster convergence of the methods.

Table 7.1: Itemized results of the detailed examples.

Instance (# of discrete vars., # of continuous vars., # of const.)	Regularization method	ROA								RLP/NLP						
		Iterations	Time [s]	MILP time [s]	Regularization time [s]	NLP time [s]	Best solution found time [s]	Infeasible NLPs	Relative opt. gap	Nodes	Time [s]	Regularization time [s]	NLP time [s]	Best solution found time [s]	Infeasible NLPs	Relative opt. gap
cvxnonsep_normcon20 (10, 10, 1)	None	426	148.9	67.7	-	70.5	0.2	175	0.09%	239	27.2	-	19.8	24.5	20	0.01%
	ℓ_2^2	**64**	**19.3**	**5.6**	**7**	**5.3**	**0.2**	**0**	**0.07%**	77	25.6	6.6	5.9	0.2	0	0.01%
	ℓ_1	65	22.2	5.9	6.8	7.4	0.3	0	0.10%	71	28.7	10.5	5.3	0.4	0	0.01%
	ℓ_∞	103	35.8	10.5	10.6	11.5	0.6	0	0.07%	**37**	**19.4**	**1.6**	**2.7**	**0.2**	**0**	**0.01%**
	$\mathcal{L}_2$	67	31.7	6.2	10.9	8.7	0.4	0	0.08%	47	25.6	4.2	4.4	0.2	0	0.01%
	$\nabla^2\mathcal{L}$	64	27.4	5.3	9.4	7.4	0.3	0	0.07%	60	29	5.6	5.3	0.4	0	0.01%
	$\mathcal{L}_1/\ell_2^2$	62	20.9	3.2	6.8	7	0.4	0	0.10%	73	26.5	6.2	6.7	0.2	0	0.01%
	$\mathcal{L}_1$	125	43.4	13.1	16.3	10.4	0.3	0	0.10%	35	21.1	2.2	3	0.2	0	0.01%
cvxnonsep_psig40 (20, 20, 0)	None	905	667.5	512.5	-	112.8	612.9	0	0.09%	**857**	**462.1**	**-**	**191.2**	**274.1**	**0**	**0.01%**
	ℓ_2^2	213	124.3	35	66.8	15.8	4.8	0	0.06%	173	899.5	112.4	30	194.5	0	0.34%
	ℓ_1	223	161.7	51.8	70.2	26	8.6	0	0.10%	166	899.7	107.9	29.8	204.7	0	0.34%
	ℓ_∞	286	172.2	66.4	59	31.8	61.3	0	0.10%	202	900.5	52.5	61	26.3	0	0.34%
	$\mathcal{L}_2$	**195**	**306**	**54.2**	**180.9**	**29.3**	**22.3**	**0**	**0.07%**	3	899.4	0.8	1.2	33	0	0.34%
	$\nabla^2\mathcal{L}$	209	203.3	38	113.3	22.7	6.8	0	0.10%	3	899.3	0.7	0.9	26.4	0	0.34%
	$\mathcal{L}_1/\ell_2^2$	211	195.1	21.2	103	29.3	7.9	0	0.10%	2	899.4	0.3	0.9	0.7	0	0.35%
	$\mathcal{L}_1$	773	901.3	241.6	542	73.6	877.2	0	1.78%	2	899.4	0.3	0.9	0.7	0	0.35%
nvs11 (3, 0, 3)	None	24	5.1	1.2	-	3.2	4.2	13	0.00%	18	5.5	-	4.9	0.6	12	0.00%
	ℓ_2^2	13	2.9	0.6	0.6	1.5	2.3	3	0.00%	6	2.2	0.5	1.4	0.4	3	0.00%
	ℓ_1	**14**	**2.8**	**0.6**	**0.5**	**1.3**	**1.8**	**3**	**0.00%**	6	2.2	0.5	1.4	0.4	3	0.00%
	ℓ_∞	13	3.4	0.6	0.8	1.6	2.6	3	0.00%	6	3.2	0.6	2.1	0.6	2	0.00%
	$\mathcal{L}_2$	9	2.9	0.5	0.7	1.1	1	3	0.00%	5	2.8	0.5	1.6	0.6	2	0.00%
	$\nabla^2\mathcal{L}$	14	3.7	0.6	1.1	1.4	2.4	3	0.00%	5	2.4	0.5	1.4	0.5	2	0.00%
	$\mathcal{L}_1/\ell_2^2$	13	5.6	0.3	1.5	2.7	4	3	0.00%	5	3.2	0.7	2	0.7	2	0.00%
	$\mathcal{L}_1$	27	7.5	1.2	2.3	3.1	4.9	18	0.00%	**2**	**1.7**	**0.3**	**1**	**0.7**	**2**	**0.00%**
nvs12 (4, 0, 4)	None	26	6.2	1.6	-	3.6	5.9	13	0.00%	23	7.1	-	6.3	0.5	13	0.00%
	ℓ_2^2	16	3.8	0.8	1	1.5	2.8	4	0.00%	6	2.2	0.4	1.3	0.4	3	0.00%
	ℓ_1	16	3.5	0.7	0.7	1.6	2.4	4	0.00%	8	2.9	0.6	1.8	0.5	4	0.00%
	ℓ_∞	14	3.5	0.8	0.9	1.3	2.6	3	0.00%	6	3.6	0.7	2.2	0.7	1	0.00%
	$\mathcal{L}_2$	**9**	**3.3**	**0.6**	**0.8**	**1.3**	**1.2**	**3**	**0.00%**	6	3.4	0.6	1.9	0.6	3	0.00%
	$\nabla^2\mathcal{L}$	16	4.6	1	1.2	1.6	3	3	0.00%	10	4.2	0.9	2.5	0.5	7	0.00%
	$\mathcal{L}_1/\ell_2^2$	14	6.2	0.8	1.6	2.6	4.5	3	0.00%	6	4	0.9	2.3	0.7	3	0.00%
	$\mathcal{L}_1$	33	10.1	2.2	2.7	3.8	9.4	20	0.00%	**2**	**1.8**	**0.3**	**1**	**0.7**	**2**	**0.00%**
slay08m (112, 72, 252)	None	24	24.4	10.7	-	8.2	20.4	0	0.07%	50	89.3	-	23.3	83.4	0	0.00%
	ℓ_2^2	**13**	**15**	**4.3**	**4.3**	**3.8**	**12.6**	**0**	**0.07%**	23	68.2	21.1	8.7	56.4	0	0.00%
	ℓ_1	16	16.5	4.9	3.8	4.3	7.2	0	0.07%	20	58.5	7.9	8.1	53.4	0	0.01%
	ℓ_∞	17	20.9	6	5.6	5.5	6.3	0	0.00%	15	88.5	7.1	7.9	77.5	0	0.00%
	$\mathcal{L}_2$	18	51.1	8.1	6.3	6.9	16.3	0	0.07%	22	92.4	7.4	8.9	24.5	0	0.00%
	$\nabla^2\mathcal{L}$	12	21.3	2.8	2.4	2.8	19.7	0	0.07%	22	100	8.2	7.7	93.9	0	0.00%
	$\mathcal{L}_1/\ell_2^2$	13	24	5.7	6.9	4.9	7.3	0	0.07%	**18**	**57.2**	**9.2**	**8.7**	**54.6**	**0**	**0.01%**
	$\mathcal{L}_1$	55	94.4	24.6	28.8	21.1	92.8	0	0.00%	23	83.1	5.7	11.1	51.7	0	31.65%
smallinvDAXr1b150-165 (30, 1, 4)	None	98	51.3	18.1	-	24.9	51.3	0	0.10%	119	64.4	-	52.6	62	0	0.01%
	ℓ_2^2	14	9.2	1.7	2.3	3.3	9.2	0	0.09%	**27**	**22.3**	**5.7**	**9.5**	**15.7**	**0**	**0.01%**
	ℓ_1	16	13.6	2.8	2.6	5.1	13.6	0	0.05%	36	28.3	7.7	13.5	11.4	0	0.01%
	ℓ_∞	18	15.6	3.2	3.3	6	15.5	0	0.07%	36	34.3	9.3	17.4	15	0	0.01%
	$\mathcal{L}_2$	**2**	**2.3**	**0.3**	**0.4**	**0.6**	**2.3**	**0**	**0.02%**	20	30.1	16.3	7.6	3.5	0	0.01%
	$\nabla^2\mathcal{L}$	13	9.9	1.3	2.7	3	9.9	0	0.06%	26	23.8	8.1	7.9	13.3	0	0.01%
	$\mathcal{L}_1/\ell_2^2$	14	10.8	0.6	2.8	4.1	10.8	0	0.08%	29	30	8.2	13	17	0	0.01%
	$\mathcal{L}_1$	225	265.1	57.6	82.2	79.4	265.1	0	0.09%	15	16.8	2.6	7	16.2	0	16.28%

7.7.3 Numerical results

Based on the ROA method's good performance in the previous section, we perform a benchmark on Problem Set 1 and Problem Set 2. The software Paver 2 [202] is used to analyze the performance of the different methods proposed in this work. We decide to present our results in the form of absolute performance profiles, as seen in Figures 7.8, 7.9, 7.10 and 7.11. These plots show the total number of instances found to be solved within 0.1% of the known optimal solution of the problem against a measure of algorithmic effort, either solution time or iterations. Two extra lines are included in these figures, where the "Virtual best" and "Virtual worst" alternatives are included. These cases are constructed with the best and worst solver for each instance, respectively. Notice that we define iterations in the single-tree context as the number of NLP-I problems solved.

Figures 7.8 and 7.9 show the performance of the multi-tree implementation of the different methods for the highly nonlinear instances, defined according to (7.17). In general, the regularization methods achieve a better performance in terms of solution time and iterations than OA. For simple examples, highlighted on the left side of the profiles and given by instances solvable in less than 10 seconds or requiring fewer than ten iterations, OA seems to outperform most of the regularization methods, except for ROA-$\mathcal{L}_2$. This method, called Q-OA in [89] performs almost as the Virtual best solver in terms of iterations, demonstrating the value of incorporating the constraint curvature information in the regularization via the second-order Taylor approximation of the Lagrangean. In terms of solution time, the advantages of this approach are reduced given the complexity associated with obtaining the Hessian of the Lagrangean and, more importantly, addressing an MIQP regularization problem. Toward the end of the time limit, the other regularization methods catch up to the performance of ROA-$\mathcal{L}_2$, with ROA-$\nabla^2\mathcal{L}$ being able to solve 104 problems, the most among all the methods, after 15 minutes. Note that the worst method is ROA-$\mathcal{L}_1$, which, as

mentioned above, is not an actual regularization method given that its projection objective function does not induce a stabilization center. Traditional OA can solve 93 of the 122 problems to 0.1% of the optimal solution in the Problem Set 2 within 900 seconds.

When considering all convex MINLP in MINLPLib, Problem Set 1, the gap between regularization-based methods and OA reduces, mainly since most of these instances have low nonlinearity. Some alternatives of regularization methods solve more instances than OA, with ROA-ℓ_1 solving 303 out of the 358 instances within the time limit, 11 more than OA. The performance profiles for the instances in Problem Set 1 are included in the Appendix in Figures 7.12 and 7.13.

The performance profiles for the Problem Set 2 of RLP/NLP are shown in Figures 7.10 and 7.11. In terms of NLP subproblems solved, the OA method is almost equivalent to the virtual worst, demonstrating that the regularization approaches lead to a more meaningful solution of NLP problems in the solution procedure. This observation is not directly translated into the time profiles, considering that solving an extra mixed-integer program for every incumbent solution in the tree is an expensive step, although justifiable with reducing iterations. Considering Problem set 2, the most successful approach is RLP/NLP-ℓ_2^2 being able to solve 111 instances, 12 more than OA.

When considering Problem Set 1, the single-tree implementation of RLP/NLP-$\mathcal{L}_2$ solves the least number of instances within the time limit. This contrasts with the good performance this regularization had for the multi-tree implementation. Both in multi-tree and single-tree, when considering all the convex MINLP instances from MINLPLib, the most successful regularization type was ℓ_1. This demonstrates the potential that linearly representable regularizations have in terms of performance.

Out of Problem Set 2, 76 problems out of 122 could be solved by all methods, and 12 could not be solved using the multi-tree methods. All single-tree methods were able to solve 96 of

Table 7.2: Details for each method infeasible subproblems when solving Problem Set 1 (358 instances).

	ROA		RLP/NLP			
Regularization method	# of instances with infeasible NLP-I	Fraction of infeasible NLP-I	# of instances with infeasible NLP-I	Fraction of infeasible NLP-I	# of instances with infeasible MIP-Proj	Fraction of infeasible MIP-Proj
None	80	2376/13656=17.4%	90	2825/24113=11.7%	-	-
ℓ_2^2	73	1350/6420=21.0%	78	1452/6415=22.6%	290	2486/6006=41.4%
ℓ_1	73	1488/6926=21.5%	82	1432/7295=19.6%	293	3028/7482=40.5%
ℓ_∞	70	1615/8699=18.6%	77	1870/7478=25.0%	289	2697/7262=37.1%
$\mathcal{L}_2$	72	1456/5283=27.6%	80	883/4852=18.2%	282	2105/4419=47.6%
$\nabla^2\mathcal{L}$	73	1451/5499=26.4%	78	891/5111=17.4%	278	2221/4740=46.9%
$\mathcal{L}_1/\ell_2^2$	69	1435/6347=22.6%	80	926/5342=17.3%	278	2086/4878=42.8%
$\mathcal{L}_1$	79	1722/15190=11.3%	86	883/6090=14.5%	259	2371/5636=42.1%

these instances, and none could solve 10 of those instances. For the whole test set, Problem Set 1, 249 instances of 358 were solved by all methods, and 48 were not solved by any in the multi-tree implementation. The single-tree methods were slightly more successful, with 258 instances being solved by all and 46 by none.

As similarly found in [89], the number of infeasible NLP-I problems encountered diminished when using regularization methods. Both in the multi- and single-tree implementations, the regularization approaches were able to solve fewer instances requiring the solution of problem NLP-f compared to OA.

An exciting finding of Table 7.2 is that the fraction of problems NLP-I that were infeasible was larger for the regularization methods! This result was surprising given the hypothesis that choosing an integer combination close to a feasible solution results in trial solutions close to the feasible region, resulting in fewer infeasible trial solutions. The explanation for this behavior has to do with the total number of NLP-I problems solved by each method. Being the OA iterations less time-consuming more could be performed within the time limit. This would set the denominator in the fraction to be large, making the fraction smaller than for the other methods. This is not a measure to be considered independently from the previous results. The fewer iterations a problem requires to converge, the fewer NLP-I problem it needs to solve. Therefore, the large number of solved NLPs indicates inefficient

methods, although it would decrease the fraction of infeasible subproblems encountered. This can be observed with ROA-$\mathcal{L}_1$, whose fraction of infeasible NLPs is the lowest among all the tested methods. However, it was the weakest alternative considered in this chapter.

For the single-tree implementation, we observed that Remark 7.6.1 often appeared in practice, with three-quarters of the instances presenting at some point infeasible MIP-Proj problems. These infeasible problems arise from the weak LB coming from the B&B tree, leading to an average of 40% of all MIP-Proj problems being infeasible in the single-tree setting.

7.8 Conclusions and future work

This chapter presents a new solution framework for multi-tree and single-tree Outer-approximation based on regularizations for solving convex MINLP problems. We present seven different regularization methods for OA through this framework, including two that were presented earlier in [89]. These regularizations can be classified into two groups: those based on distance minimization around an incumbent solution, and those based on approximations of the Lagrangean function around that incumbent solution. The regularization approach relies on the solution of an auxiliary mixed-integer program, which, based on the objective function's choice, can be a mixed-integer linear program or a mixed-integer quadratic program. We show that the convergence proofs from [89] directly applies to these methods as well, thus, guaranteeing convergence to the optimal solution. Moreover, the regularization ideas are integrated with the LP/NLP Branch & Bound method [90] leading to a single-tree regularization algorithm for convex MINLP. The implementation of these methods was done on top of the Mixed-Integer Decomposition Toolbox for Pyomo - MindtPy [190] in open-source code. We evaluated these approaches experimentally and

compared them to OA by solving a large set of convex MINLP problems. We observed that the regularization approaches are especially well-fitted for highly nonlinear problems, achieving performance improvements compared to OA. This confirms the hypothesis that staying close to the feasible solutions ensures the integer combinations found by the linearizations to stay close to the convex set defined by the nonlinear constraints. For almost linear instances, the benefits of the regularization technique are sometimes lost due to the cost of solving auxiliary projection problem, which also aligns well with the results in [89]. However, our results demonstrate that using linearly representable regularizations do improve the average performance for all the convex MINLP instances at the benchmarking library MINLPLib, including the highly linear ones.

As future work, we consider an interesting avenue to perform updates in the Hessian of the Lagrangean estimate. As in the BFGS algorithm [272], the Hessian of the Lagrangean needs not to be computed exactly, and its approximation can be iteratively refined with the first estimate of it being a scaled identity matrix. This technique has proved extremely useful in trust-region methods for continuous NLP problems, such as Sequential Quadratic Programming (SQP) [267, 277]. Moreover, the two extra parameters introduced in the regularization methods α and d have been maintained constant throughout these experiments. These hyperparameters represent a trade-off between how trustworthy the incumbent solution is compared to the optimal solution and how much exploration far from that incumbent solution is required. One can imagine a dynamic update policy for these parameters, balancing the incumbent solutions' exploration and exploitation as a future research direction.

7.A Algorithmic description of OA and LP/NLP Branch & Bound

This section presents the algorithmic description of the Outer-approximation method [25, 30], in Algorithm 10, and the LP/NLP Branch & Bound method [90, 129], in Algorithm 11.

7.B Reformulation of Norms 1 and ∞ using Linear Programming

This section shows the valid reformulations of optimization problems with norms 1 and infinity in the objective function using auxiliary variables and linear constraints. This reformulation is exact in the sense that they preserve the local and global optima from the original problem [278]. These reformulations are particularly interesting since they allow the regularization problem MIP-Proj to be written as Mixed-Integer Linear Programming (MILP) problems, instead of Mixed-Integer Quadratic Programming (MIQP) problems, as in the work byKronqvist, Bernal, and Grossmann [89]. The manurity of MILP solution methods compared to MIQP allows these problems to be more quickly solvable in practice.

The norm-1 of a vector $\mathbf{v} \in V \subseteq \mathbb{R}^N$ whose components might be negative or positive, $\ell_1(\mathbf{v}) = \|\mathbf{v}\|_1 = \sum_{i=1}^{N} |v_i|$ can be reformulated in the case that this term appears in the objective function with a set of linear constraints. Through the addition of $2N$ non-negative slack variables $\mathbf{s}^+, \mathbf{s}^- \in \mathbb{R}^N_+$, and N linear equality constraints the following reformulation is valid:

$$\begin{aligned} \min_{\mathbf{v}} \quad & \|\mathbf{v}\|_1 \\ \text{s.t.} \quad & \mathbf{v} \in V \subseteq \mathbb{R}^N \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} \min_{\mathbf{v},\mathbf{s}^+,\mathbf{s}^-} \quad & \sum_{i=1}^{N} s_i^+ + s_i^- \\ \text{s.t.} \quad & \mathbf{s}^+ - \mathbf{s}^- = \mathbf{v} \\ & \mathbf{v} \in V \subseteq \mathbb{R}^N,\ \mathbf{s}^+ \in \mathbb{R}^N_+,\ \mathbf{s}^- \in \mathbb{R}^N_+ \end{aligned} \tag{7.18}$$

This reformulation is applied to the regularization problem MIP-Proj when considering the ℓ_1 regularization function as in (7.4), resulting in problem MIP-Proj-ℓ_1. It can also be potentially applied to the feasibility NLP problem NLP-f.

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu,\mathbf{s}^+,\mathbf{s}^-} \quad & \sum_{j=1}^{n+m} s_j^+ + s_j^- \\
\text{s.t.} \quad & s_j^+ - s_j^- = x_j - \bar{x}_j \quad \forall j = \{1,\dots,n\} \\
& s_{n+j}^+ - s_{n+j}^- = y_j - \bar{y}_j \quad \forall j = \{1,\dots,m\} \\
& \mu \leq \widehat{f_k^\star} \\
& f(\mathbf{x}^i,\mathbf{y}^i) + \nabla f(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq \mu \quad \forall i = 1,\dots,k, \\
& g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq 0 \quad \forall i = 1,\dots k, \forall j \in \mathcal{I}_i, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m,\ \mu \in \mathbb{R},\ \mathbf{s}^+,\mathbf{s}^- \in \mathbb{R}^{n+m}_+
\end{aligned}
\tag{MIP-Proj-ℓ_1}
$$

The norm-∞ of a vector $\mathbf{v} \in V \subseteq \mathbb{R}^N$ whose components might be negative or positive, $\ell_\infty(\mathbf{v}) = \|\mathbf{v}\|_\infty = \max_{i=\{1,\dots,N\}} |v_i|$ can be reformulated in the case that this term appears in the objective function with a set of linear constraints. Through the addition of one non-negative slack variable $s \in \mathbb{R}_+$, and $2N$ linear inequality constraints, the following reformulation is valid:

$$
\begin{aligned}
\min_{\mathbf{v}} \quad & \|\mathbf{v}\|_\infty \\
\text{s.t.} \quad & \mathbf{v} \in V \subseteq \mathbb{R}^N
\end{aligned}
\quad \Leftrightarrow \quad
\begin{aligned}
\min_{\mathbf{v},s} \quad & s \\
\text{s.t.} \quad & s \geq \mathbf{v} \\
& s \geq -\mathbf{v} \\
& \mathbf{v} \in V \subseteq \mathbb{R}^N,\ s \in \mathbb{R}_+
\end{aligned}
\tag{7.19}
$$

This is the usual choice for reformulating problem NLP-f, and can also be used to reformulate problem MIP-Proj with ℓ_∞ regularization objective function, as in (7.5). This

last problem formulation is:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y},\mu,s} \quad & s \\
\text{s.t.} \quad & s \geq x_j - \bar{x}_j \quad \forall j = \{1,\dots,n\} \\
& s \geq \bar{x}_j - x_j \quad \forall j = \{1,\dots,n\} \\
& s \geq y_j - \bar{y}_j \quad \forall j = \{1,\dots,m\} \\
& s \geq \bar{y}_j - y_j \quad \forall j = \{1,\dots,m\} \\
& \mu \leq \widehat{f_k^\star} \\
& f(\mathbf{x}^i,\mathbf{y}^i) + \nabla f(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq \mu \quad \forall i = 1,\dots,k, \\
& g_j(\mathbf{x}^i,\mathbf{y}^i) + \nabla g_j(\mathbf{x}^i,\mathbf{y}^i)^\top \begin{bmatrix} \mathbf{x}-\mathbf{x}^i \\ \mathbf{y}-\mathbf{y}^i \end{bmatrix} \leq 0 \quad \forall i = 1,\dots k, \forall j \in \mathcal{I}_i, \\
& \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} \leq \mathbf{b}, \\
& \mathbf{x} \in \mathbb{R}^n,\ \mathbf{y} \in \mathbb{Z}^m,\ \mu \in \mathbb{R},\ s \in \mathbb{R}_+
\end{aligned}
\tag{MIP-Proj-ℓ_∞}
$$

7.C Performance profiles for Problem Set 1

In this section of the Appendix, we present the performance profiles for the multi-tree and single-tree implementation of the methods included in this chapter when solving all 358 convex MINLP problems in Problem Set 1. Figures 7.12 and 7.13 include the time and iteration performance profiles for the multi-tree implementation, respectively. Figures 7.14 and 7.15 include the time and iteration performance profiles for the single-tree implementation, respectively. Notice that we define iterations in the single-tree context as the number of NLP-I problems solved.

Algorithm 9 An algorithm summarizing the regularized LP/NLP (RLP/NLP) method.

Define accepted optimality gap $\epsilon \geq 0$ and choose the parameter $\alpha \in (0, 1]$.

1. Initialization.
   - 1.1 Obtain a relaxed solution $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ by solving an integer relaxation of the MINLP problem.
   - 1.2 Generate cuts at $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ according to (7.1) and construct problems OA-MILP.
   - 1.3 Set node counter $k = 1$, $UB^0 = \infty$ and $LB^0 = -\infty$.
2. Begin B&B search for problem OA-MILP and continue until $UB^{k-1} - LB^{k-1} \leq \epsilon$.
   - 2.1 If a new solution $\widehat{\mathbf{x}}, \widehat{\mathbf{y}}$ is found, then generate cuts at $\widehat{\mathbf{x}}, \widehat{\mathbf{y}}$, set $\mathbf{y}^k = \widehat{\mathbf{y}}$ and update LB^k according to current B&B tree. Add the new cuts to open nodes of the B&B tree and to problem MIP-Proj.
   - 2.2 If a feasible solution has been found or provided
     - 2.2.1 Calculate the estimated optimal value $\widehat{f_k^\star}$ according to (7.2).
     - 2.2.2 Solve problem MIP-Proj to update $\mathbf{y}^k$. If the regularization problem is infeasible, keep $\mathbf{y}^k$ unchanged.
   - 2.3 Solve problem NLP-I with integer variables fixed as $\mathbf{y}^k$ to obtain $\mathbf{x}^k$ and $\lambda^{\mathbf{k}}$.
     - 2.3.1 If problem NLP-I is feasible, set $UB^k = \min\{f(\mathbf{x}^k, \mathbf{y}^k), UB^{k-1}\}$.
       - 2.3.1.1 If $f(\mathbf{x}^k, \mathbf{y}^k) \leq f(\bar{\mathbf{x}}, \bar{\mathbf{y}})$, set $\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\lambda} = \mathbf{x}^k, \mathbf{y}^k, \lambda^k$.
     - 2.3.2 If problem NLP-I is infeasible, obtain $\mathbf{x}^k$ by solving feasibility problem NLP-f.
   - 2.4 Generate cuts at $\mathbf{x}^k, \mathbf{y}^k$ according to (7.1) and add these to problem MIP-Proj, and add these as global lazy constraints to the B&B tree.
   - 2.5 (Optional) Generate no-good cuts at $\mathbf{y}^k$ and add these as global lazy constraints to the B&B tree of problem OA-MILP.
   - 2.6 Increase node counter, $k = k + 1$
3. Return $\bar{\mathbf{x}}, \bar{\mathbf{y}}$ as the optimal solution $\mathbf{x}^\star, \mathbf{y}^\star$.

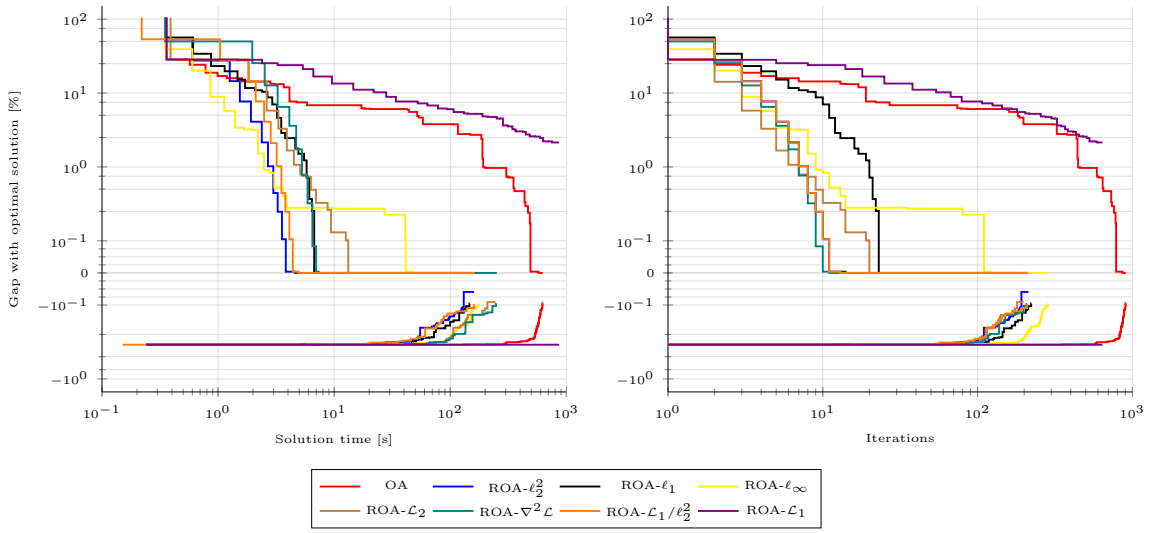


Figure 7.6: Bound profiles for instance `cvxnonsep_psig40` against (a) solution time and (b) iterations using the multi-tree ROA method as described in Algorithm 8. The figure shows the upper and lower bounds obtained by the different regularization methods.

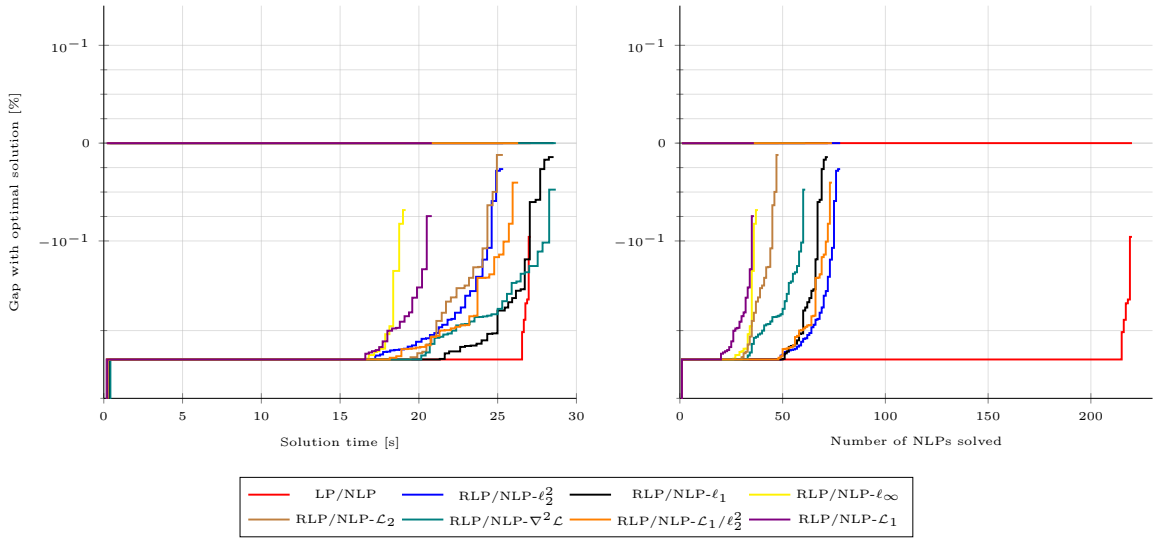


Figure 7.7: Bound profiles for instance `cvxnonsep_normcon20` against (a) solution time and (b) NLP problems solved using the single-tree RLP/NLP methods as described in Algorithm 9. The figure shows the upper and lower bounds obtained by the different regularization methods.

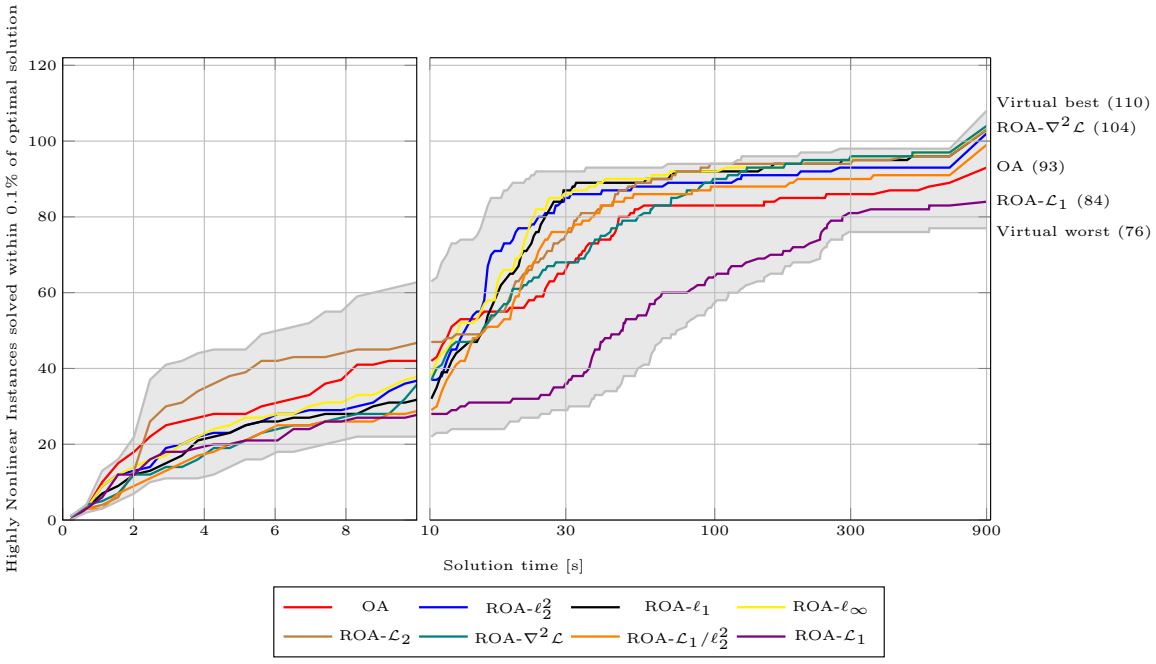


Figure 7.8: Time performance profile for highly nonlinear instances for multi-tree ROA method as described in Algorithm 8.

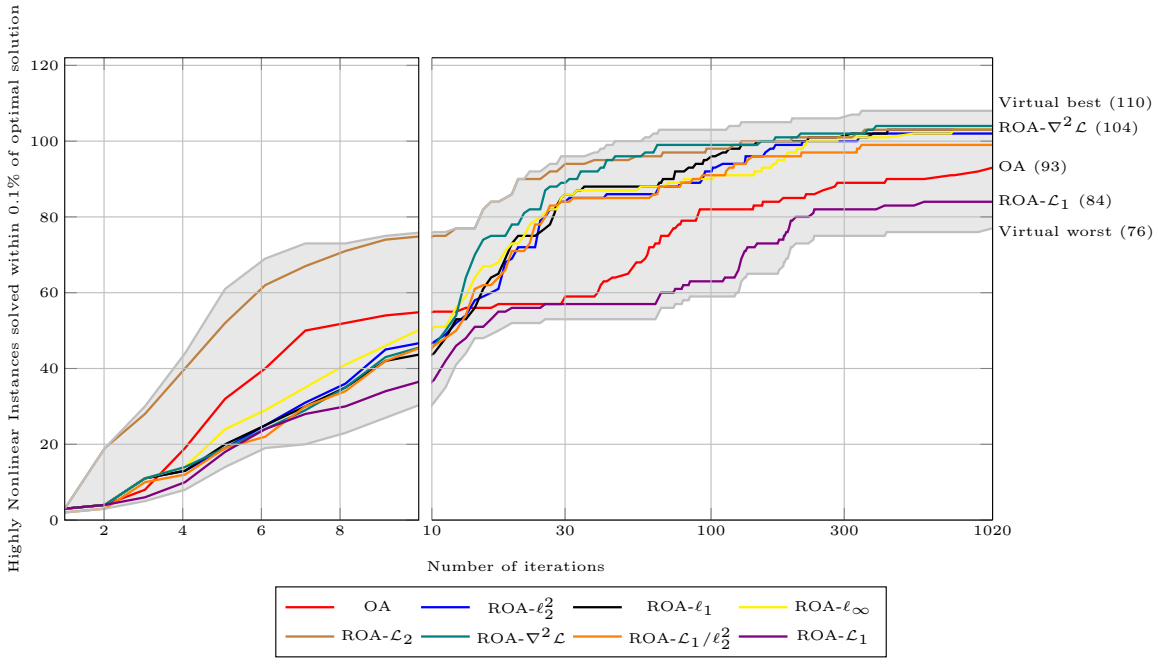


Figure 7.9: Iteration performance profile for highly nonlinear instances for multi-tree ROA method as described in Algorithm 8.

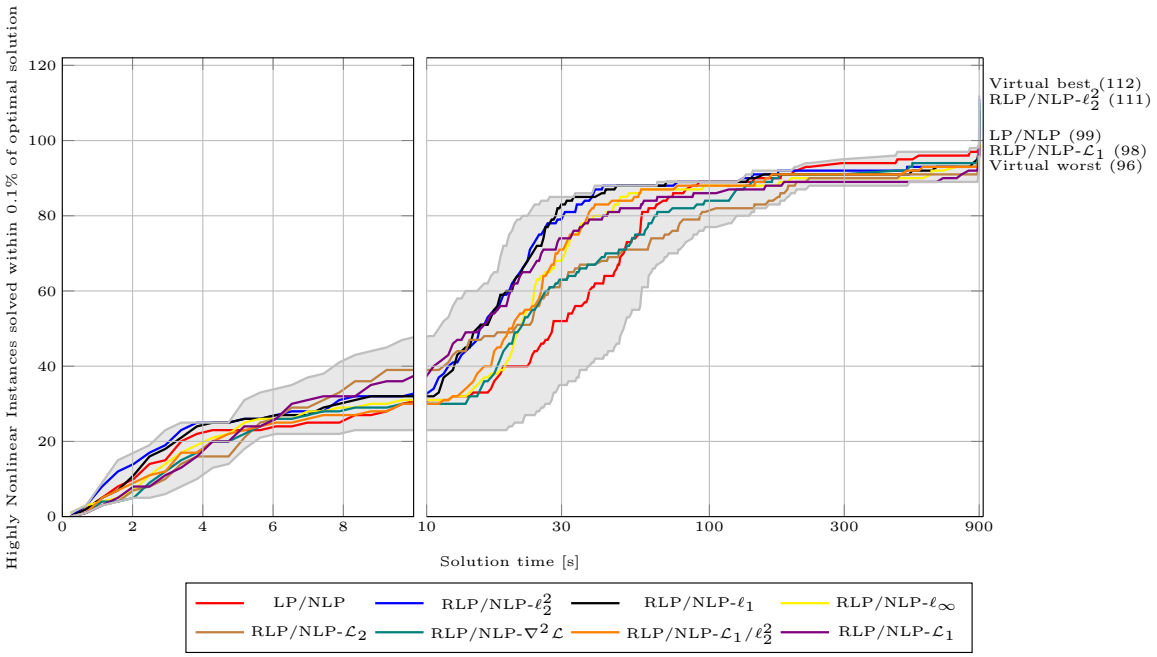


Figure 7.10: Time performance profile for highly nonlinear instances for single-tree RLP/NLP methods as described in Algorithm 9.

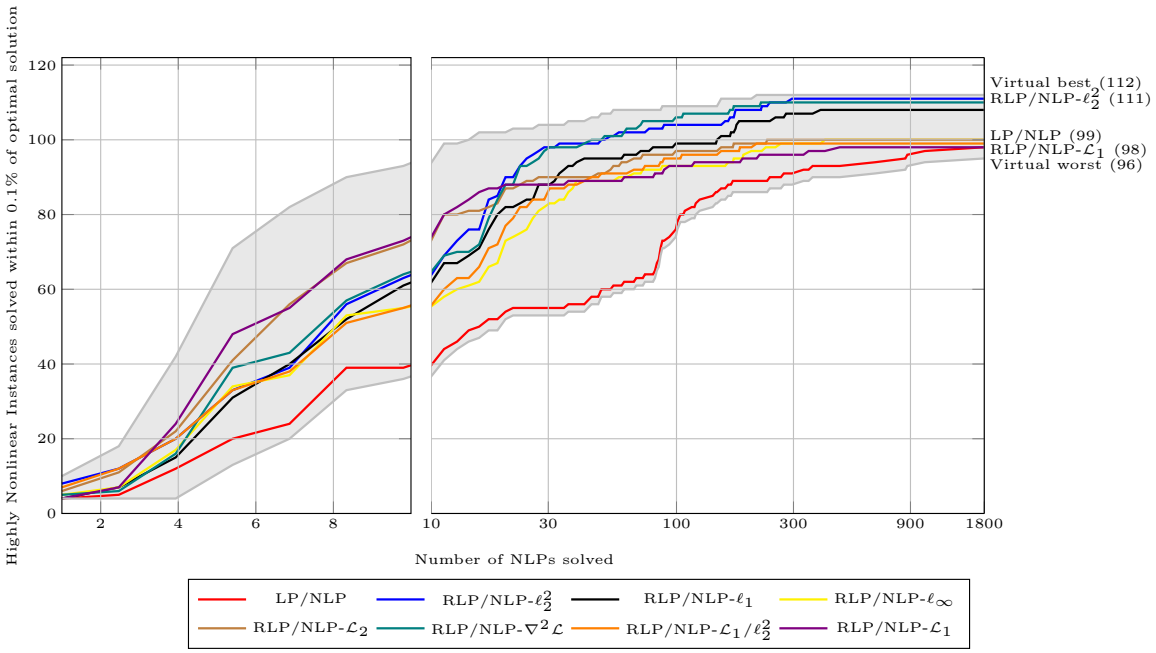


Figure 7.11: Iteration performance profile for highly nonlinear instances for single-tree RLP/NLP methods as described in Algorithm 9.

Algorithm 10 An algorithm summarizing the Outer-approximation method.

Define accepted optimality gap $\epsilon \geq 0$.

1. Initialization.
   - 1.1 Obtain a relaxed solution $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ by solving an integer relaxation of the MINLP problem.
   - 1.2 Generate cuts at $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ according to (7.1) and construct problem OA-MILP.
   - 1.3 Set iteration counter $k = 1$, $UB^0 = \infty$ and $LB^0 = -\infty$.
2. Repeat until $UB^{k-1} - LB^{k-1} \leq \epsilon$.
   - 2.1 Solve problem OA-MILP to obtain $\mathbf{y}^k$ and LB^k
   - 2.2 Solve problem NLP-I with integer variables fixed as $\mathbf{y}^k$ to obtain $\mathbf{x}^k$.
     - 2.2.1 If problem NLP-I is feasible, set $UB^k = \min\{f(\mathbf{x}^k, \mathbf{y}^k), UB^{k-1}\}$.
     - 2.2.2 If problem NLP-I is infeasible, obtain $\mathbf{x}^k$ by solving feasibility problem NLP-f and set $UB^k = UB^{k-1}$.
   - 2.3 Generate cuts at $\mathbf{x}^k, \mathbf{y}^k$ according to (7.1) and add these to problem OA-MILP.
   - 2.4 (Optional) Generate no-good cuts at $\mathbf{y}^k$ and add these to problems OA-MILP.
   - 2.5 Increase iteration counter, $k = k + 1$
3. Return the best found solution.

Algorithm 11 An algorithm summarizing the LP/NLP based Branch & Bound algorithm.

Define accepted optimality gap $\epsilon \geq 0$ and choose the parameter $\alpha \in (0, 1]$.

1. Initialization.
   - 1.1 Obtain a relaxed solution $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ by solving an integer relaxation of the MINLP problem.
   - 1.2 Generate cuts at $\widetilde{\mathbf{x}}, \widetilde{\mathbf{y}}$ according to (7.1) and construct problems OA-MILP.
   - 1.3 Set node counter $k = 1$, $UB^0 = \infty$ and $LB^0 = -\infty$.
2. Begin Branch & Bound for problem OA-MILP and terminate until $UB^{k-1} - LB^{k-1} \leq \epsilon$.
   - 2.1 If a new incumbent integer solution $\widehat{\mathbf{x}}, \widehat{\mathbf{y}}$ is found, check if $\widehat{\mathbf{y}} \in \{\mathbf{y}^i | i = 1, \ldots, k-1\}$.
     - 2.1.1 if $\widehat{\mathbf{y}} \in \{\mathbf{y}^i | i = 1, \ldots, k-1\}$, then skip this iteration and continue the Branch & Bound process.
     - 2.1.2 if $\widehat{\mathbf{y}} \notin \{\mathbf{y}^i | i = 1, \ldots, k-1\}$, set $\mathbf{y}^k = \widehat{\mathbf{y}}$ and set LB^k to the lower bound of current B&B tree.
   - 2.2 Solve problem NLP-I with integer variables fixed as $\mathbf{y}^k$ to obtain $\mathbf{x}^k$.
     - 2.2.1 If problem NLP-I is feasible, set $UB^k = \min\{f(\mathbf{x}^k, \mathbf{y}^k), UB^{k-1}\}$.
     - 2.2.2 If problem NLP-I is infeasible, obtain $\mathbf{x}^k$ by solving feasibility problem NLP-f and set $UB^k = UB^{k-1}$.
   - 2.3 Generate cuts at $\mathbf{x}^k, \mathbf{y}^k$ according to (7.1) and add these as global lazy constraints to the B&B tree of problem OA-MILP.
   - 2.4 (Optional) Generate no-good cuts at $\mathbf{y}^k$ and add these as global lazy constraints to the B&B tree of problem OA-MILP.
   - 2.5 Increase node counter, $k = k + 1$.
3. Return the best found solution.

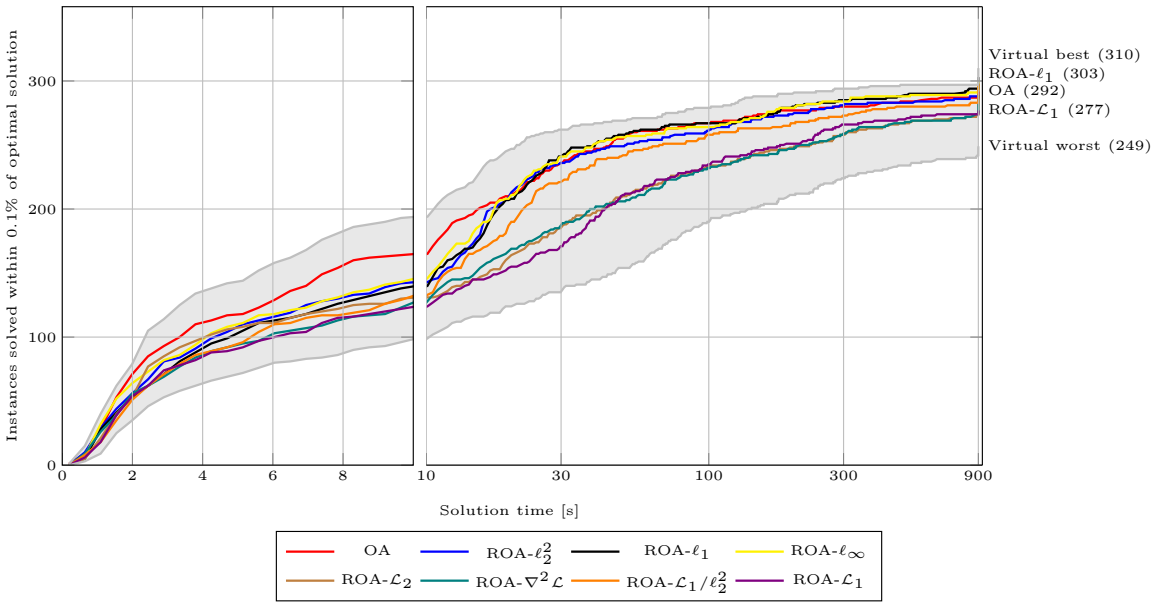


Figure 7.12: Time performance profile for multi-tree ROA method as described in Algorithm 8.

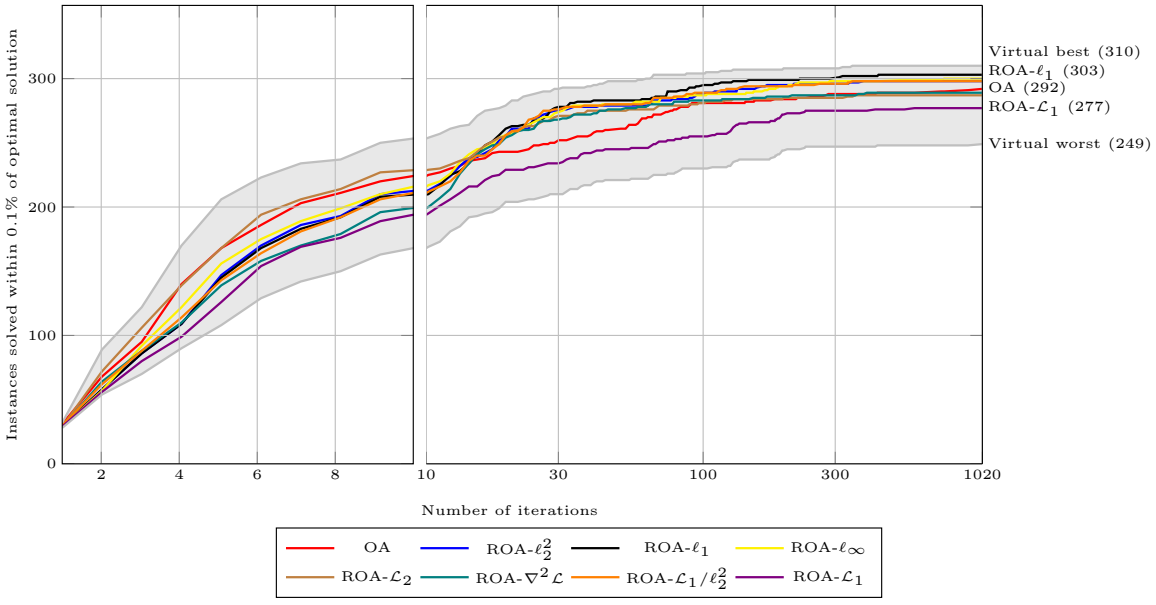


Figure 7.13: Iteration performance profile for multi-tree ROA method as described in Algorithm 8.

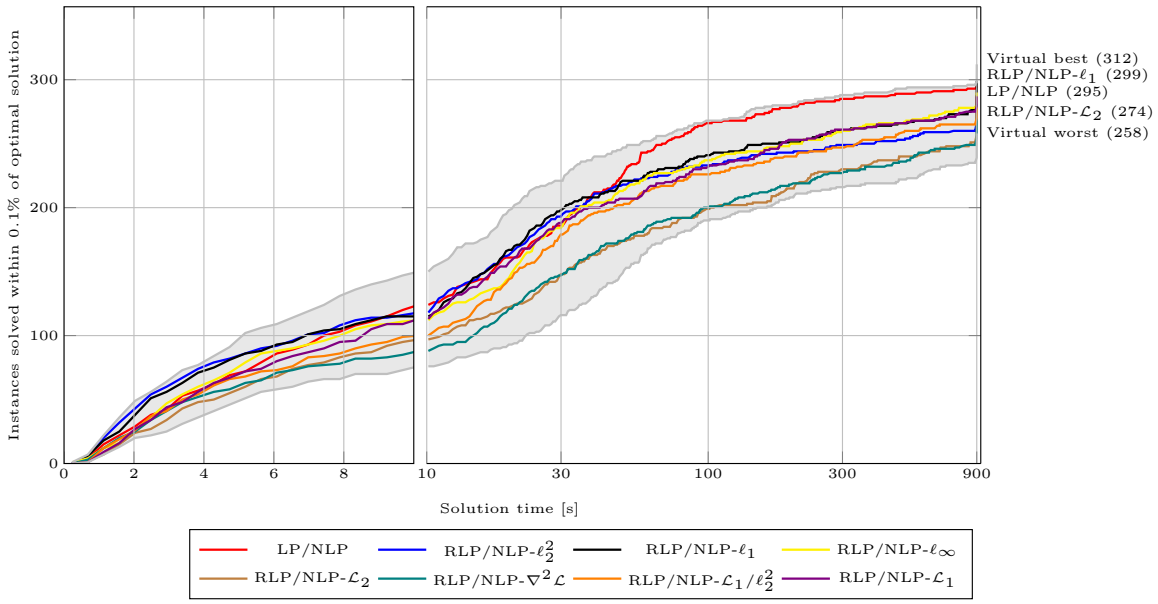


Figure 7.14: Time performance profile for single-tree RLP/NLP methods as described in Algorithm 9.

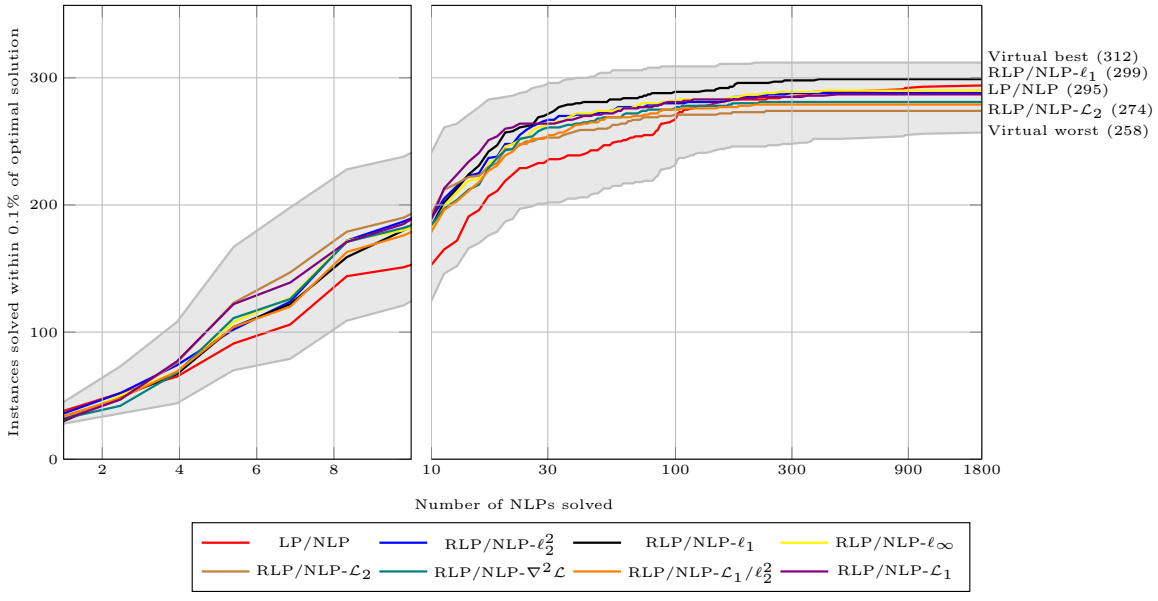


Figure 7.15: Iteration performance profile single-tree RLP/NLP methods as described in Algorithm 9

Chapter 8

Easily Solvable Convex MINLP Derived from Generalized Disjunctive Programming using Cones

8.1 Introduction

Mathematical Programming is an approach to solving optimization problems by casting their characteristics into a set of mathematical equations that include objectives that need to be optimized, subject to constraints that need to be satisfied, by modifying the values of the variables. The nature of the objectives, constraints, and variables allow the different mathematical programs to be classified. When nonlinear algebraic inequalities describe the constraints and objectives, and the variables are allowed to take either continuous or discrete values, the mathematical program becomes a Mixed-Integer Nonlinear Programming (MINLP) problem. MINLP is a problem class of great interest, both theoretical [24] and practical [60, 251]. In particular, MINLP problems *formulations* allow modeling a wide range of applications. Most industrial problems can be modeled using MINLP [23], many of those concerning the Process Systems Engineering (PSE) community. Different applications of MINLP within PSE include process control, process design, process operation, and molecular dynamics [4].

A particular class of MINLP problems is where the constraints are convex functions.

Although it is non-convex because of the nature of the discrete variables, this problem is known as convex MINLP [25, 26]. This class of MINLP is a subject of interest given the many applications that it can represent and the challenging algorithmic requirements that need to be tackled to address their problems. For a review on convex MINLP, refer to Kronqvist et al. [26].

Among the solution techniques for convex MINLP, several have been adapted from the Mixed-Integer Linear Programming (MILP), including Branch & Bound [27] and Benders Decomposition [28];. In contrast, others generalize the solutions methods for convex continuous Nonlinear Programming (NLP) problems, such as the Extended Cutting Plane methods [29]. A particularly successful approach to convex MINLP is the outer-approximation (OA) method proposed by Duran and Grossmann [30], where an iterative solution of a convex NLP and an MILP subproblems is performed. The MILP is derived through first-order Taylor approximations, or gradient-based linearizations, of the nonlinear constraints at the NLP solutions, and the NLPs stem from the problems appearing when fixing the values of the discrete variables at the MILP solution [25, 30]. Many of the current commercial tools to solve convex MINLP rely on the OA method [26].

In continuous convex programming, solutions methods have also been derived by generalizing Linear Programming (LP) notions and techniques. One of the most successful ones has been the proposal of convex optimization problems as problems defined over cones, or Conic Programming (CP) problems [31]. CP is a numerically stable alternative for convex programming [31], given that it exploits properties of the conic sets. Convex Programming problems described via algebraic convex nonlinear constraints of the form $f(x) \leq 0$ can be equivalently posed as linear transformation of the variables belonging to convex sets $\mathcal{K}$, i.e., $Ax - b \in \mathcal{K}$ [31, 32]. A generalization of CP where some variables are required to take discrete values is Mixed-Integer Conic Programming (MICP). MICP problems are highly

expressible and can represent a wide range of optimization problem [33]. Many of these applications have been gathered in the problem library CBLib [34].

The automatic identification and translation of the two equivalent descriptions of convex sets is a crucial feature for algorithmic solution software, *solvers*, development. This is since the description of problems using algebraic constraints is more natural for practitioners. However, the conic description of the problem allows taking advantage of mathematical properties such as conic duality for more stable solution procedures. Generic solvers have been designed to tackle CP problems, e.g., MOSEK [35], ECOS [36], and Hypatia [37]. This translation is not trivial [38–40]. However, it has been achieved for the quadratic case allowing for solution methods based on conic programming to be used for these problems. An alternative to translating practical optimization problems into CP is via Disciplined Convex Programming (DCP) [41], where strict rules of function definitions guarantee the problem's convexity and perform the translation such that they can be solved through generic conic solvers.

In the mixed-integer setting, solvers have been designed to take as input the MICP problem taking advantage of this form of the optimization problem structure, e.g., Mosek [35], and Pajarito [42–44]. Even for solvers that do not necessarily consider the conic representation of convex problems, identifying such structures leads to improvements in its performance, such as in SCIP [45, 46] and BARON [47]. There is a significant potential for MINLP solvers to perform automatic reformulations once they identify correct structures [48]. An example of the automatic identification of conic structures is Mixed-Integer Quadratically-constrained Quadratic Programming (MIQCQP) problems can now be tackled through Mixed-Integer Second-Order Conic Programming (MISOCP) methods in commercial solvers such as Knitro [49], Xpress [50], Gurobi [51], and CPLEX [52].

The discrete nature of the integer variables in mixed-integer programming problems

has been exploited to derive efficient solution methods for these problems. In particular, deriving sets of extra inequalities, *cutting planes* or *cuts*, has allowed a considerable speedup in the solution of these problems, see [53]. One of the key disciplines for deriving such cutting planes is Disjunctive Programming, which considers the optimization over disjunctive sets such as the one given by the domain of the discrete variables. In the convex nonlinear setting, the conic structure has been exploited to derive special cutting planes for MICP solution methods [54–56]. A source of these problems are those driven by *indicator variables*, that activate or deactivate sets of constraints [48], see a review by Bonami et al. [57].

Generalized Disjunctive Programming (GDP) was proposed by Grossmann and Lee [58] as an intuitive way of describing the logic behind applications. In this setting, sets of constraints are activated with logical variables linked to each other by logical constraints, including disjunctions. This mathematical description of the problem can be tackled directly by logic-based optimization methods [59], which generalize mixed-integer solution methods to the logical domain. Another way of solving these problems is through reformulations into mixed-integer programs, where the logical variables are mapped to binary or indicator variables. Depending on the linearity of the constraints within the GDP, the reformulations can yield a MILP or MINLP problem. The two most common reformulations are: the Big-M reformulations, where a large coefficient is added to make the constraints redundant in the case their associated indicator variable is inactive; and the Hull Reformulation (HR), where using Disjunctive Programming theory, a set of constraints in an extended space are derived such that their projection onto the space of the original variables is the convex hull of the disjunctive sets. These two reformulations yield different mixed-integer models, which can be characterized by size and tightness. The tightness of a mixed-integer model is measured through the difference of the optimal solution of the problem, ignoring the discrete constraints, known as the *continuous relaxation*, and the original problem's optimal

solution [60]. The Big-M and Hull reformulations offer a tradeoff between tightness and problem size. The HR is the tightest possible model, while the Big-M formulation does not require any additional continuous variables and constraints. Both the model size and tightness are relevant to the efficiency of solution methods of mixed-integer programs [61].

For convex GDPs, the HR requires modeling the perspective function of the convex functions in the disjunctions, which can be complex for nonlinear functions given its non-differentiability at 0 [61, 62]. Perspective functions arise in formulations of convex MINLP since they are in general part of the reformulation of disjunctive programs. Moreover, the MINLP formulations involving the perspective function can be used either directly in tight formulations of convex disjunctive programs, either in the original variable space [57, 63] or in a higher dimensional space [58, 64] , or indirectly through the generation of valid cutting-planes [65, 66]. A recent computational study shows the positive impact of perspective cuts in the MINLP framework [46]. The importance of this perspective formulations and the challenges associated with their implementation have motivated its study, where customized versions have been derived for special cases [48, 63, 67] or the proposal of ε-approximations for general convex functions [62, 64].

8.1.1 Contribution and outline

Below we list the contributions of this chapter. We propose the formulation of convex Generalized Disjunctive Programming (GDP) problems using conic inequalities leading to conic GDP problems. We then show the reformulation of conic GDPs into Mixed-Integer Conic Programming (MICP) problems through both the Big-M and Hull Reformulations. These reformulations have the advantage that they are representable using the same cones as the original conic GDP. In the case of HR, they require no approximation of the perspective function. Moreover, the MICP problems derived can be solved by specialized conic solvers

and offer a natural extended formulation amenable to both conic and gradient-based solvers. We present the closed-form of several convex functions and their respective perspectives in terms of conic sets, allowing users to formulate their conic GDP problems easily. We finally implement a large set of conic GDP examples and solve them via the traditional and conic mixed-integer reformulations. These examples include applications from Process Systems Engineering, Machine learning, and randomly generated instances. Our results show that the conic structure can be exploited to obtain solutions to these challenging MICP problems more efficiently.

The remaining of this chapter is organized as follows. Sectin 8.2 presents the necessary background for this work, including details on cones, the perspective function, Disjunctive Programming, and Generalized Disjunctive Programming. Section 8.3 introduces conic Generalized Disjunctive Programming, with its relevant reformulations into MICP problems. We show our computational results in Section 8.4, where the different instances are presented separately in two groups: those representable by quadratic cones and those by exponential cones. Finally, we conclude this chapter with the lessons learned, discussion, and future research directions stemming from this work in Section 8.5.

8.2 Background

In this chapter, we are concerned with convex Mixed-Integer Nonlinear Programming (MINLP) problems. A convex MINLP problem is defined as

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{g}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}, \\ & \mathbf{y}^l \leq \mathbf{y} \leq \mathbf{y}^u, \\ & \mathbf{x} \in \mathbb{R}^{n_x}_+, \ \mathbf{y} \in \mathbb{Z}^{n_y}, \end{aligned} \tag{MINLP}$$

where the objective function $f : \mathbb{R}^{n_x+n_y} \to \mathbb{R} \cup \{\infty\}$ is convex and the constraints $\mathbf{g} : \mathbb{R}^{n_x+n_y} \to (\mathbb{R} \cup \{\infty\})^J$ define a convex set $\mathcal{F} = \{\mathbf{x} \in \mathbb{R}^{n_x}_+, \mathbf{y} \in \mathbb{R}^{n_y} | \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0}\}$. Although it is not necessary, we will consider that each constraint, $g_j(\mathbf{x}, \mathbf{y})$ for $j \in \{1, \ldots, J\} = [\![J]\!]$, is a convex function. We consider bounded integer variables $\mathbf{y}$. Without loss of generality, we will assume that the objective function is linear, which can be achieved through the epigraph reformulation [26]. Notice that, although the continuous relaxation of the feasible region F is convex, the original convex MINLP feasible region is non-convex given the discrete nature of variables $\mathbf{y}$.

8.2.1 Cones

For a thorough discussion about convex optimization and conic programming, we refer the reader to [31]. The following definitions are required for the remainder of the chapter.

The set $\mathcal{K} \subseteq \mathbb{R}^k$ is a cone if $\forall(\mathbf{z}, \lambda) \in \mathcal{K} \times \mathbb{R}_+, \lambda\mathbf{z} \in \mathcal{K}$. The dual cone of $\mathcal{K} \subseteq \mathbb{R}^k$ is

$$\mathcal{K}^* = \left\{\mathbf{u} \in \mathbb{R}^k : \mathbf{u}^T\mathbf{z} \geq \mathbf{0}, \forall \mathbf{z} \in \mathcal{K}\right\}, \tag{8.1}$$

and it is self-dual if $\mathcal{K} = \mathcal{K}^*$. The cone is pointed if $\mathcal{K} \cap (-\mathcal{K}) = \{\mathbf{0}\}$. A cone is proper if it is closed, convex, pointed, and with non-empty interior. If $\mathcal{K}$ is proper, then its dual $\mathcal{K}^*$ is proper too. $\mathcal{K}$ induces a partial order on $\mathbb{R}^k$:

$$\mathbf{x} \succcurlyeq_{\mathcal{K}} \mathbf{y} \iff \mathbf{x} - \mathbf{y} \in \mathcal{K}, \tag{8.2}$$

which allows us to define a conic inequality as

$$\mathbf{A}\mathbf{x} \succcurlyeq_{\mathcal{K}} \mathbf{b}, \tag{8.3}$$

where $\mathbf{A} \in \mathbb{R}^{m \times k}$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathcal{K}$ a cone.

When using a cone that represents the Cartesian product of others, i.e., $\mathcal{K} = \mathcal{K}_{n_1} \times \cdots \times \mathcal{K}_{n_r}$ with each cone $\mathcal{K}_{n_i} \subseteq \mathbb{R}^{n_i}$, its corresponding vectors and matrices are partitioned conformally,

i.e.,

$$\begin{aligned}
\mathbf{x} &= (\mathbf{x_1}; \ldots; \mathbf{x_r}) && \text{where } \mathbf{x_i} \in \mathbb{R}^{n_i}, \\
\mathbf{y} &= (\mathbf{y_1}; \ldots; \mathbf{y_r}) && \text{where } \mathbf{y_i} \in \mathbb{R}^{n_i}, \\
\mathbf{c} &= (\mathbf{c_1}; \ldots; \mathbf{c_r}) && \text{where } \mathbf{c_i} \in \mathbb{R}^{n_i}, \\
\mathbf{A} &= (\mathbf{A}_1; \ldots; \mathbf{A}_r) && \text{where } \mathbf{A} \in \mathbb{R}^{m \times n_i}.
\end{aligned} \tag{8.4}$$

Furthermore, if each cone $\mathcal{K}_{n_i} \subseteq \mathbb{R}^{n_i}$ is proper, then $\mathcal{K}$ is proper too.

A Conic Programming (CP) problem is then defined as:

$$\begin{aligned}
\min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x} \\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\
& \mathbf{x} \in \mathcal{K} \subseteq \mathbb{R}^k.
\end{aligned} \tag{CP}$$

Examples of proper cones are:

- The non-negative orthant

$$\mathbb{R}^k_+ = \left\{ \mathbf{z} \in \mathbb{R}^k : \mathbf{z} \geq \mathbf{0} \right\}. \tag{8.5}$$

- The positive semi-definite cone

$$\mathbb{S}^k_+ = \left\{ Z \in \mathbb{R}^{k \times k} : Z = Z^T, \lambda_{min}(Z) \geq 0 \right\}, \tag{8.6}$$

  where $\lambda_{min}(Z)$ denotes the smallest eigenvalue of Z.

- The second-order cone, Euclidean norm cone, or Lorentz cone

$$Q^k = \left\{ \mathbf{z} \in \mathbb{R}^k : z_1 \geq \sqrt{\sum_{i=2}^{k} z_i^2} \right\}. \tag{8.7}$$

- The exponential cone [279]

$$\begin{aligned}
\mathcal{K}_{exp} &= \text{cl}\left\{ (z_1, z_2, z_3) \in \mathbb{R}^3 : z_1 \geq z_2 e^{z_3/z_2}, z_1 \geq 0, z_2 > 0 \right\} \\
&= \left\{ (z_1, z_2, z_3) \in \mathbb{R}^3 : z_1 \geq z_2 e^{z_3/z_2}, z_1 \geq 0, z_2 > 0 \right\} \bigcup \mathbb{R}_+ \times \{\mathbf{0}\} \times (-\mathbb{R}_+) \\
&= \left\{ (z_1, z_2, z_3) \in \mathbb{R}^3 : z_1 \geq z_2 e^{z_3/z_2}, z_2 \geq 0 \right\}.
\end{aligned} \tag{8.8}$$

Of these cones, the only one not being self-dual or symmetric is the exponential cone.

Other cones that are useful in practice are

- The rotated second-order cone or Euclidean norm-squared cone

$$Q_r^k = \left\{ \mathbf{z} \in \mathbb{R}^k : 2z_1 z_2 \geq \sqrt{\sum_{i=3}^{k} z_i^2}, z_1, z_2 \geq 0 \right\}, \tag{8.9}$$

  This cone can be written as a rotation of the second-order cone, i.e., $\mathbf{z} \in Q^k \iff R_k \mathbf{z} \in Q_r^k$ with $R_k := \begin{bmatrix} \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ \sqrt{2}/2 & -\sqrt{2}/2 & 0 \\ 0 & 0 & I_{k-2} \end{bmatrix}$, or by a linear transformation of the second-order cone, i.e., $Q_r^k = \left\{ \mathbf{z} \in \mathbb{R}^k : (z_1 + z_2, z_1, \ldots, z_k) \in Q^{k+1} \right\}$.

- The power cone, with $l < k, \sum_{i \in [\![l]\!]} \alpha_i = 1$,

$$\mathcal{P}_k^{\alpha_1,\ldots,\alpha_n} = \left\{ \mathbf{z} \in \mathbb{R}^k : \prod_{i=1}^{l} z_i^{\alpha_i} \geq \sqrt{\sum_{i=l+1}^{k} z_i^2}, \quad z_i \geq 0 \quad i \in [\![l]\!] \right\}. \tag{8.10}$$

  This cone can be decomposed using a second-order cone and $l-1$ three-dimensional power cones

$$\mathcal{P}_3^{\alpha} = \left\{ (z_1, z_2, z_3) \in \mathbb{R}^3 : z_1^{\alpha} z_2^{1-\alpha} \geq |z_3|, \quad z_1, z_2 \geq 0 \right\}, \tag{8.11}$$

  through $l-1$ additional variables $(u, v_1, \ldots, v_{l-2})$,

$$\mathbf{z} \in \mathcal{P}_k^{\alpha_1,\ldots,\alpha_n} \iff \begin{cases} (u, z_{l+1}, \ldots, z_k) \in Q^{k-l+1}, \\ (z_1, v_1, u) \in \mathcal{P}_3^{\alpha_1}, \\ (z_i, v_i, v_{i-1}) \in \mathcal{P}_3^{\bar{\alpha}_i}, \quad i = 2, \ldots, l-1, \\ (z_{l-1}, z_l, v_{l-2}) \in \mathcal{P}_3^{\bar{\alpha}_{l-1}}, \end{cases} \tag{8.12}$$

  where $\bar{\alpha}_i = \alpha_i / (\alpha_i + \cdots + \alpha_n)$ for $i = 2, \ldots, l-1$. $\mathcal{P}_3^{\alpha}$ can be represented using linear and exponential cone constraints, i.e., $\lim_{\alpha \to 0} (z_1, z_2, z_2 + \alpha z_3) \in \mathcal{P}_3^{\alpha} = (z_1, z_2, z_3) \in \mathcal{K}_{exp}$

Most, if not all, applications-related convex optimization problems can be represented by conic extended formulations using the these standard cones [35], i.e., in problem CP, the cone $\mathcal{K}$ is a product $\mathcal{K}_1 \times \cdots \times \mathcal{K}_r$, where each $\mathcal{K}_i$ is one of the recognized cones mentioned above. Equivalent conic formulations for more exotic convex sets using unique cones can be formulated with potential advantages for improved solution performance [37].

As mentioned in the introduction, an alternative to a convex optimization problem's algebraic description as in problem MINLP is the following Mixed-Integer Conic Programming (MICP) problem:

$$\begin{aligned} \min_{\mathbf{z},\mathbf{y}} \quad & \mathbf{c}^T\mathbf{z} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{y} = \mathbf{b}, \\ & \mathbf{y}^l \leq \mathbf{y} \leq \mathbf{y}^u, \\ & \mathbf{z} \in \mathcal{K} \subseteq \mathbb{R}^k,\ \mathbf{y} \in \mathbb{Z}^{n_y}, \end{aligned} \tag{MICP}$$

where $\mathcal{K}$ is a closed convex cone.

Without loss of generality, integer variables need not be restricted to cones, given that corresponding continuous variables can be introduced via equality constraints. Notice that for an arbitrary convex function $f : \mathbb{R}^k \to \mathbb{R} \cup \{\infty\}$, one can define a closed convex cone using its recession,

$$\mathcal{K}_f = \text{cl}\{(\mathbf{z}, \lambda, t) : \lambda f(\mathbf{z}/\lambda) = \widetilde{f}(\mathbf{z}, \lambda) \leq t, \lambda > 0\}, \tag{8.13}$$

where the function $\widetilde{f}(\mathbf{z}, \lambda)$ is the perspective function of function $f(\mathbf{z})$, and whose algebraic representation is a central piece of this work. Closed convex cones can also be defined as the recession of convex sets. On the other hand, a conic constraint is equivalent to a convex inequality,

$$\mathbf{A}\mathbf{x} \succcurlyeq_{\mathcal{K}} \mathbf{b} \iff \mathbf{g}(\mathbf{x}) \leq \mathbf{0}, \tag{8.14}$$

for appropriately chosen smooth convex functions $\mathbf{g}(\mathbf{x})$ [54, 280].

We can therefore reformulate problem MINLP in the following parsimonious manner [42]:

$$
\begin{aligned}
\min_{\substack{\mathbf{x},\mathbf{y},s_{[\![J]\!]} \\ \mathbf{x}_f,\mathbf{y}_f,t_f, \\ \mathbf{x}_{[\![J]\!]},\mathbf{y}_{[\![J]\!]}}} \quad & t_f \\
\text{s.t.} \quad & \mathbf{x} = \mathbf{x_f}, \mathbf{y} = \mathbf{y_f}, \\
& ((\mathbf{x_f};\mathbf{y_f}), 1, t_f) \in \mathcal{K}_f, \\
& \mathbf{x} = \mathbf{x_j}, \mathbf{y} = \mathbf{y_j}, \\
& \Big((\mathbf{x_j};\mathbf{y_j}), 1, s_j\Big) \in \mathcal{K}_{g_j}, s_j \in \mathbb{R}_+, j \in [\![J]\!], \\
& \mathbf{y}^l \leq \mathbf{y} \leq \mathbf{y}^u, \\
& \mathbf{x} \in \mathbb{R}^{n_x}_+, \ \mathbf{y} \in \mathbb{Z}^{n_y},
\end{aligned}
\tag{8.15}
$$

where copies of the original variables $\mathbf{x}$ and $\mathbf{y}$ are introduced for the objective function and each constraints, $\mathbf{x_f}, \mathbf{y_f}, \mathbf{x_j}, \mathbf{y_j}, j \in [\![J]\!]$, such that each belongs to the recession cone of each constraint defined as in (8.13). Each conic set requires the introduction of an epigraph variable t and a recession variable λ. The epigraph variable from the objective function, t_f, is used in the new objective, and the ones corresponding to the constraints are set as non-negative slack variables s_j. The recession variables λ in (8.13) are fixed to one in all cases.

Notice that problem 8.15 is in MICP form with $\mathcal{K} = \mathbb{R}^{n_x+J}_+ \times \mathcal{K}_f \times \mathcal{K}_{g_1} \times \cdots \times \mathcal{K}_{g_J}$. As mentioned above, the case when $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_r$ where each $\mathcal{K}_i$ is a recognized cone is more useful from practical purposes. Lubin et al. [42] showed that all the convex MINLP instances at the benchmark library MINLPLib [100] could be represented with these recognized cones.

8.2.2 Perspective function

For a convex function $h(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ its perspective function $\widetilde{h}(\mathbf{x}, \lambda) : \mathbb{R}^{n+1} \to \mathbb{R} \cup \{\infty\}$ is defined as

$$\widetilde{h}(\mathbf{x}, \lambda) = \begin{cases} \lambda h(\mathbf{x}/\lambda) & \text{if } \lambda > 0 \\ \infty & \text{otherwise} \end{cases} \tag{8.16}$$

The perspective of a convex function is convex, but not closed. Hence, consider the closure of the perspective function $(\mathrm{cl}\widetilde{h})(\mathbf{x}, \lambda)$ defined as

$$\left(\mathrm{cl}\,\widetilde{h}\right)(\mathbf{x}, \lambda) = \begin{cases} \lambda h(\mathbf{x}/\lambda) & \text{if } \lambda > 0 \\ h'_\infty(\mathbf{x}) & \text{if } \lambda = 0 \\ \infty & \text{otherwise} \end{cases}, \tag{8.17}$$

where $h'_\infty(\mathbf{x})$ is the recession function of function $h(\mathbf{x})$[281, Section B Proposition 2.2.2], and which in general does not have a closed-form.

The closure of the perspective function of a convex function is relevant for convex MINLP on two ends. On the one hand, it appears when describing the closure of the convex hull of disjunctive sets. On the other hand, as seen above, it can be used to define closed convex cones $\mathcal{K}$, that determine the feasible region of conic programs. Relying on amenable properties of convex cones, conic programs can be addressed with specialized algorithms allowing for more efficient solution methods.

The closure of the perspective function presents a challenge when implementing it for nonlinear optimization models, given that it is not defined at $\lambda = 0$. Modeling this function becomes necessary when writing the convex hull of the union of convex sets, as seen below. This difficulty has been addressed by several authors in the literature through ε-approximations. The first proposal was made by Lee and Grossmann [64], where

$$\left(\mathrm{cl}\,\widetilde{h}\right)(\mathbf{x}, \lambda) \approx (\lambda + \varepsilon) h\left(\frac{\mathbf{x}}{\lambda + \varepsilon}\right). \tag{8.18}$$

This approximation is exact when $\varepsilon \to 0$. However, it requires values for ε, which are small enough to become numerically challenging when implemented in a solution algorithm.

Furman, Sawaya, and Grossmann [62] propose another approximation for the perspective function such that

$$\left(\mathrm{cl}\,\widetilde{h}\right)(\mathbf{x},\lambda) \approx ((1-\varepsilon)\lambda+\varepsilon)h\left(\frac{\mathbf{x}}{(1-\varepsilon)\lambda+\varepsilon}\right) - \varepsilon h(0)(1-\lambda), \tag{8.19}$$

which is exact for values of $\lambda = 0$ and $\lambda = 1$, is convex for $h(\mathbf{x})$ convex, and is exact when $\varepsilon \to 0$ as long as $h(\mathbf{0})$ is defined. Using this approximation in the set describing the system of equations of the closed convex hull of a disjunctive set also has properties that are beneficial for mathematical programming.

This approximation is used in software implementations when reformulating a disjunctive set using its hull relaxation [59, 282]. Notice that even with its desirable properties, the approximation introduces some error for values $\varepsilon > 0$; hence it is desirable to circumvent its usage. As shown in [61] and the Section 8.4 below, using a conic constraint to model the perspective function allows for a more efficient solution of convex MINLP problems.

8.2.3 Disjunctive Programming

Optimization over disjunctive sets is denoted as Disjunctive Programming [2, 283]. A disjunctive set is given by the system of inequalities joined by logical operators of conjunction ($\wedge$, "and") and disjunction ($\vee$, "or"). These sets are non-convex and represent usually the union of convex sets. The main reference on Disjunctive Programming is the book by Balas [2].

Consider the following disjunctive set

$$C = \left\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \in \bigvee_{i\in I} C_i\right\} = \bigcup_{i\in I}\{\mathbf{x} \in \mathbb{R}^n : \mathbf{x} \in C_i\}, \tag{8.20}$$

where $|I|$ is finite. Each set defined as $C_i := \{\mathbf{x} \in \mathbb{R}^n | \mathbf{h}_i(\mathbf{x}) \leq \mathbf{0}\}$ is a convex, bounded, and non-empty set defined by a vector valued function $\mathbf{h}_i : \mathbb{R}^n \to (\mathbb{R} \cup \{\infty\})^{J_i}$. Notice that is it sufficient for C_i to be convex that each component of $\mathbf{h}_i$, $h_{ij}, j \in \{1, \cdots, J_i\}$, is a proper closed convex function, although it is not a necessary condition.

Ceria and Soares [284] characterize the closure of the convex hull of C, cl conv(C), with the following result.

Theorem 8.2.1. [284] Let $C_i = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{h}_i(\mathbf{x}) \leq \mathbf{0}\} \neq \emptyset$, assume that each component of $\mathbf{h}_i$, $h_{i[\![J_i]\!]}$, is a proper closed convex function, and let

$$\mathcal{H} = \left\{ \begin{array}{ll} \mathbf{x} = \sum_{i \in I} \mathbf{v}_i, & \\ \sum_{i \in I} \lambda_i = 1, & \\ \left(\mathrm{cl}\, \widetilde{\mathbf{h}}_i\right)(\mathbf{v}_i, \lambda_i) \leq 0, & i \in I, \\ \mathbf{v}_i \in \mathbb{R}^n, & i \in I, \\ \lambda_i \in \mathbb{R}_+, & i \in I \end{array} \right\}. \tag{8.21}$$

Then cl conv$(\bigcup_{i \in I} C_i) = \mathrm{proj}_{\mathbf{x}}(\mathcal{H})$.

Proof. See [284, Theorem 1] and [57, Theorem 1]. □

Theorem 8.2.1 provides a description of cl conv(C) in a higher dimensional space, an *extended formulation*. This Theorem generalizes the result by [2, 283, 285, 286] where all the convex sets C_i are polyhedral. Even though the extended formulations induce growth in the size of the optimization problem, some of them have shown to be amenable for MINLP solution algorithms [42, 269, 287, 288].

A similar formulation was derived by Stubbs and Mehrotra [65] in the context of a Branch-and-cut method for Mixed-binary convex programs. These authors notice that the

extended formulation might not be computationally practical, hence they derive linear inequalities or cuts from this formulation to be later integrated into the solution procedure. Similar ideas have been explored in the literature [66]. In particular cases, the dimension of the extended formulation can be reduced to the original size of the problem, e.g., when there are only two terms in the disjunction, i.e., $|I| = 2$, and one of the convex sets C_i is a point [61]. A description in the original space of variables has also been given for the case when one set C_1 is a box and the constraints defining the other C_2 is defined by the same bounds as the box and nonlinear constraints being isotone [63]. This has been extended even further by Bonami et al. [57] with complementary disjunctions. In other words the activation of one disjunction implies that the other one is deactivated, in the case that the functions that define each set $\mathbf{h}_{\{1,2\}}$ are isotone and share the same indices on which they are non-decreasing. The last two cases present the formulation in the original space of variables by paying a prize of exponentially many constraints required to represent cl conv(C).

In the case that C_i is compact, its recession cone is the origin, i.e., $C_{i\infty} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{h}'_{i\infty}(\mathbf{x}) \leq \mathbf{0}\} = \{\mathbf{0}\}$ [281, Section A, Proposition 2.2.3]. This fact, together with (8.17) and Theorem 8.2.1, forces that for a compact C_i, a value of $\lambda_i = 0$ implies $\mathbf{v}_i = 0$. This fact has been used to propose mixed-integer programming formulations for expressing the disjunctive choice between convex sets, by setting the interpolation variables to be binary $\lambda_i \in \{0,1\}, i \in I$ [64, 289], i.e.,

$$\mathcal{H}_{\{0,1\}} = \left\{ \begin{array}{ll} \mathbf{x} = \sum_{i \in I} \mathbf{v}_i, & \\ \sum_{i \in I} \lambda_i = 1, & \\ \left(\mathrm{cl}\, \widetilde{\mathbf{h}}_i\right)(\mathbf{v}_i, \lambda_i) \leq 0, & i \in I, \\ \mathbf{v}_i \in \mathbb{R}^n, & i \in I, \\ \lambda_i \in \{0,1\}, & i \in I \end{array} \right\}. \tag{8.22}$$

An interesting observation is that using the approximation of the closure of the perspective function from Furman, Sawaya, and Grossmann [62], for any value of $\varepsilon \in (0,1)$, $\text{proj}_{\mathbf{x}}(\mathcal{H}_{\{0,1\}}) = C$ when $\mathbf{h}_i(\mathbf{0})$ is defined $\forall i \in I$ and

$$\{\mathbf{x} \in \mathbb{R}^n : \mathbf{h}_i(\mathbf{x}) - \mathbf{h}_i(\mathbf{0}) \leq \mathbf{0}\} = \{\mathbf{0}\}, \forall i \in I \tag{8.23}$$

see [62, Proposition 1].

The condition on (8.23) is required to ensure that if $\lambda_i = 1$, then $\mathbf{v}_{i'} = 0, \forall i' \in I \setminus \{i\}$. This condition is not valid in general for a disjunctive set C, but it is sufficient to have a bounded range on $\mathbf{x} \in C_i, i \in I$. Moreover, when these conditions are satisfied, $C \subseteq \text{proj}_{\mathbf{x}}(\mathcal{H})$ using the approximation in (8.19) for $\varepsilon \in (0,1)$, with cl conv$C = \text{proj}_{\mathbf{x}}(\mathcal{H})$ in the limit when $\varepsilon \to 0$[62, Proposition 3].

8.2.4 Generalized Disjunctive Programming

The framework of Generalized Disjunctive Programming (GDP) was introduced by Raman and Grossmann [290]. This modeling paradigm extends the usual mathematical programming paradigm by allowing Boolean variables, logical constraints, and disjunctions to appear in the optimization problem formulation. We define a GDP as follows:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{Y}} \; & f(\mathbf{x}) \\
\text{s.t.} \; & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ \mathbf{h}_{ik}(\mathbf{x}) \leq \mathbf{0} \end{bmatrix}, \quad k \in K \\
& \veebar_{i \in D_k} Y_{ik}, \quad k \in K \\
& \Omega(\mathbf{Y}) = True \\
& \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u \\
& \mathbf{x} \in \mathbb{R}^n \\
& Y_{ik} \in \{False, True\}, \quad k \in K, i \in D_k,
\end{aligned}
\tag{GDP}
$$

where constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ are called global constraints, the set K represents the possible disjunctions in the problem, and each element i of the set D_k represents a disjunctive term, also called disjunct, in that disjunction. In the disjunction $k \in K$, each disjunct $i \in D_k$ has a set of constraints $\mathbf{h}_{ik}(\mathbf{x}) \leq \mathbf{0}$ which are activated when a Boolean variable associated with the disjunct is equals to $True$, i.e., $Y_{ik} = True$. Each disjunct may contain a different number of constraints J_i, i.e., $\mathbf{h}_{ik}(\mathbf{x}) = (h_{ik1}(\mathbf{x}), \ldots, h_{ikJ_i}(\mathbf{x})) = (h_{ik[\![J_i]\!]}(\mathbf{x}))$. These constraints define set $C_{ik} = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{h}_{ik}(\mathbf{x}) \leq \mathbf{0}\}$, to which the point $\mathbf{x}$ belongs to when the disjunct is active, i.e., $Y_{ik} = True$. The disjuncts within the disjunction are related through an inclusive-or operator $\vee$, which means that at least one Boolean variable in every disjunction, $Y_{ik}, k \in K$, is set to $True$. Each disjunction defines a disjunctive set, like the ones introduced in the previous section. $\Omega(\mathbf{Y})$ represent logical propositions in terms of the Boolean variables $\mathbf{Y}$. These logical constraints can be written in Conjunctive Normal Form (CNF), i.e., $\Omega(\mathbf{Y}) = \bigwedge_{t \in [\![T]\!]} \left[\bigvee_{Y_{ik} \in R_t} (Y_{ik}) \bigvee_{Y_{ik} \in Q_t} (\neg Y_{ik}) \bigvee \right]$ where for each logical clause $t \in [\![T]\!]$, the subset $R_t \subseteq \mathbf{Y}$ are non-negated Boolean variables and the subset $Q_t \subseteq \mathbf{Y}$ are the negated Boolean variables. We

assume that the exclusive-or operators among the Boolean variables for each disjunction $k \in K$, i.e., $\veebar_{i \in D_k} Y_{ik}$, are included in $\Omega(\mathbf{Y}) = True$ [233, 291]. It has been proved that GDP is equivalent to Disjunctive programming in the case that the constraints are linear [292] and convex [293].

Besides offering a more intuitive modeling paradigm of discrete problems through disjunctions, a GDP model can be used to inform computational solution tools, i.e., solvers, of the original problem's underlying structure, thus leading to improved solving performance. The tailored solution methods for GDP are usually based on generalizing algorithms for MINLP, where the optimization problems are decomposed, so the discrete variables are fixed and allow to solve the problem only in terms of the continuous variables. Different methods are used to select the combination of these discrete variables, including branching across the different values the discrete variables can take, i.e., Branch & Bound (B&B), or solving a linear approximation of the original problem [26]. For GDP algorithms, contrary to the case in MINLP, these Nonlinear Programming (NLP) subproblems only include the constraints that concern the logical variable combinations. We encounter the Logic-based Branch & Bound (LBB) and the Logic-based Outer-approximation (LOA) among these tailored algorithms. For more information on general GDP algorithms, refer to [59].

Another route to solve these problems is through the reformulation to Mixed-integer problems, where binary variables $\mathbf{y} \in \{0,1\}^{\sum_{k \in K} |D_k|}$ are added to the problem in exchange of the Boolean variables and constraints within the disjunction are enforced subject to the binary variables' value. Notice that these reformulations yield problems of the form MINLP. The logical propositions $\Omega(\mathbf{Y}) = True$ can be easily reformulated as a set of linear inequality constraints, $O\mathbf{y} \leq \mathbf{o}$, in terms of the binary variables [290, 291, 294]. In the case that $\Omega(\mathbf{Y})$ is written in CNF, this reformulation is simply $\sum_{y_{ik} \in R_t} y_{ik} + \sum_{y_{ik} \in Q_t} (1 - y_{ik}) \geq 1, t \in [\![T]\!]$. An example is the exclusive-or constraint $\veebar_{i \in D_k} Y_{ik}$ reformulated as a partitioning constraint

$\sum_{i \in D_k} y_{ik} = 1, k \in K$. These approaches take advantage of the more mature Mixed-integer solvers available commercially.

The Big-M reformulation is among the best-known reformulation for GDP problems. In this case, each disjunction's constraints are relaxed by adding a large term, M, if its corresponding binary variable is equal to zero. The formulation of the Big-M reformulation is as follows:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y}} \ & f(\mathbf{x}) \\
\text{s.t.} \ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
& h_{ikj}(\mathbf{x}) \leq M_{ikj}(1 - y_{ik}), \quad k \in K, i \in D_k, j \in [\![J_i]\!], \\
& \sum_{i \in D_k} y_{ik} = 1, \quad k \in K \\
& O\mathbf{y} \leq \mathbf{o} \\
& \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u \\
& \mathbf{x} \in \mathbb{R}^n \\
& y_{ik} \in \{0,1\}, \quad k \in K, i \in D_k,
\end{aligned}
\tag{Big-M}
$$

where the coefficient M_{ikj} has to be large enough to guarantee the enforcement of the original GDP logic, i.e., $y_{ik} = 1 \rightarrow \mathbf{h}_{ik}(\mathbf{x}) \leq \mathbf{0}$, but small enough to avoid numerical problem related to solving accuracy [60]. This can be accomplished by setting $M_{ikj} = \max_{\mathbf{x} \in \{\mathbf{x}: \mathbf{h}_{ik} \leq \mathbf{0}\}} h_{ikj}(\mathbf{x}), j \in [\![J_i]\!]$. Although traditionally used, the Big-M reformulation is well-known for its weak continuous relaxation gap, i.e., the difference in the optimal objective function when solving the problem considering $y_{ik} \in [0,1] \subset \mathbb{R}, k \in K, i \in D_k$ compared to the original problem's optimal objective; even with the tightest values for the M coefficients. This is particularly important for solution methods based on B&B, where this continuous relaxation gives the first node in the search tree.

Another valid transformation of problem GDP into a mixed-integer problem is the Hull

Reformulation (HR). This reformulation uses the same mapping of Boolean into binary variables as in Big-M. On the other hand, it introduces copies of the $\mathbf{x}$ variables, $\mathbf{v}_{ik}$ for each disjunct $k \in K, i \in D_k$ and uses the closure of the perspective function to enforce the constraints when their corresponding binary variable is active. The formulation for the HR of a GDP is as follows:

$$
\begin{aligned}
&\min_{\mathbf{x},\mathbf{v},\mathbf{y}} f(\mathbf{x}) \\
&\text{s.t. } \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
&\quad \mathbf{x} = \sum_{i \in D_k} \mathbf{v}_{ik}, \quad k \in K \\
&\quad \left(\operatorname{cl} \widetilde{\mathbf{h}}_{ik}\right)(\mathbf{v}_{ik}, y_{ik}) \leq 0, \quad k \in K, i \in D_k \\
&\quad \sum_{i \in D_k} y_{ik} = 1, \quad k \in K \\
&\quad O\mathbf{y} \leq \mathbf{o} \\
&\quad \mathbf{x}^l y_{ik} \leq \mathbf{v}_{ik} \leq \mathbf{x}^u y_{ik} \\
&\quad \mathbf{x} \in \mathbb{R}^n \\
&\quad \mathbf{v}_{ik} \in \mathbb{R}^n, \quad k \in K, i \in D_k \\
&\quad y_{ik} \in \{0,1\}, \quad k \in K, i \in D_k.
\end{aligned}
\tag{HR}
$$

The problem formulation HR is derived by replacing each disjunction with set $\mathcal{H}_{\{0,1\}}$ (8.22), presented in Section 8.2.3. Notice that in order to guarantee the validity of the formulation, the condition on (8.23) is enforced implicitly by having the bounds over $\mathbf{x}$ included in each disjunct, leading to constraint $\mathbf{x}^l y_{ik} \leq \mathbf{v}_{ik} \leq \mathbf{x}^u y_{ik}$.

In general, for GDP, no convexity assumptions are made for the functions $f, \mathbf{g}, \mathbf{h}_{ik}$ or the sets within the disjunctions C_i. This means that the continuous relaxation of either Big-M or HR might not have convex feasible regions. We refer the interested reader to the review by Ruiz and Grossmann [295] that covers the techniques to solve these challenging

optimization problems.

In order to use the theory from Conic Programming and Disjunctive programming, covered in Sections 8.2.1 and 8.2.3, respectively, we assume here that functions $f, \mathbf{g}, \mathbf{h}_{ik}$ are convex, hence the sets C_i are convex too. These are known as convex GDP problems [296].

For a literature review on GDP, we refer the reader to the chapter by Grossmann and Ruiz [291].

8.3 Conic Generalized Disjunctive Programming

The first step towards defining easily solvable convex MINLP problems via conic programming is to define a GDP with conic constraints. As mentioned in Section 8.2.1, we can use the tautological reformulation in (8.13) to write any convex GDP of form GDP as follows:

$$
\begin{aligned}
&\min_{\mathbf{x},\mathbf{Y}} \ f(\mathbf{x}) \\
&\text{s.t.} \ \ \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
&\qquad \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ \mathbf{A}_{ik}\mathbf{x} \succcurlyeq_{\mathcal{K}_{ik}} \mathbf{b}_{ik} \end{bmatrix}, \quad k \in K \\
&\qquad \Omega(\mathbf{Y}) = True \\
&\qquad \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u \\
&\qquad \mathbf{x} \in \mathbb{R}^n \\
&\qquad Y_{ik} \in \{False, True\}, \quad k \in K, i \in D_k.
\end{aligned}
\tag{GDP-Cone}
$$

Since the objective function $f(\mathbf{x})$ and the global constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ are convex we can reformulate them to a conic program via (8.13) as in problem 8.15. The sets defined within each disjunct

$$
P_{ik} := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}_{ik}\mathbf{x} \succcurlyeq_{\mathcal{K}_{ik}} \mathbf{b}_{ik}\} \tag{8.24}
$$

are convex sets, where for every disjunct $\mathbf{A}_{ik} \in \mathbb{R}^{m_i \times n}$, $\mathbf{b}_{ik} \in \mathbb{R}^{m_i}$, and $\mathcal{K}_{ik}$ is a proper cone.

Although the derivation of specific solution algorithms for problem GDP-Cone is a subject of active research, we focus on the reformulation of the given problem into Mixed-integer Programming problems. These convex GDP problems can be reformulated into a convex MINLP problem, which in turn can be written down as a MICP problem.

The first trivial reformulation is the Big-M reformulation, which yield the following problem:

$$
\begin{aligned}
&\min_{\mathbf{x},\mathbf{y}} \ f(\mathbf{x}) \\
&\text{s.t.} \ \ \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
&\quad \mathbf{A}_{ik}\mathbf{x} \succcurlyeq_{\mathcal{K}_{ik}} \mathbf{b}_{ik} + M_{ik}(1 - y_{ik}), \quad k \in K, i \in D_k, \\
&\quad \sum_{i \in D_k} y_{ik} \leq 1, \quad k \in K \\
&\quad O\mathbf{y} \leq \mathbf{o} \\
&\quad \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u \\
&\quad \mathbf{x} \in \mathbb{R}^n \\
&\quad y_{ik} \in \{0, 1\}, \quad k \in K, i \in D_k,
\end{aligned}
\tag{Big-M-Cone}
$$

To derive the Hull Reformulation of GDP-Cone, we need to characterize the convex hull of the disjunctive set (8.20) in the case that each convex and bounded set is defined using cones as in 8.24.

Theorem 8.3.1. [56] Let $\mathcal{P}_i = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}_i\mathbf{x} \succcurlyeq_{\mathcal{K}_i} \mathbf{b}_i\}$ for $i \in I$, where $\mathbf{A}_i \in \mathbb{R}^{m_i \times n}$, $\mathbf{b}_i \in \mathbb{R}^{m_i}$, and $\mathcal{K}_i$ is a proper cone, and let

$$\mathcal{P} = \left\{ \begin{array}{ll} \mathbf{x} = \sum_{i \in I} \mathbf{v}_i, & \\ \sum_{i \in I} \lambda_i = 1, & \\ \mathbf{A}_i \mathbf{v}_i \succcurlyeq_{\mathcal{K}_i} \lambda_i \mathbf{b}_i, & i \in I, \\ \mathbf{v}_i \in \mathbb{R}^n, & i \in I, \\ \lambda_i \in \mathbb{R}_+, & i \in I \end{array} \right\}. \tag{8.25}$$

Then $\text{conv}(\bigcup_{i \in I} \mathcal{P}_i) \subseteq \text{proj}_{\mathbf{x}}(\mathcal{P})$ and:

1. if $\mathcal{P}_i \neq \emptyset, \forall i \in I$, then $\text{proj}_{\mathbf{x}}(\mathcal{P}) \subseteq \text{cl conv}(\bigcup_{i \in I} \mathcal{P}_i)$
2. if $\mathcal{P}_i = \mathcal{S}_i + \mathcal{W}, \forall i \in I$, where $\mathcal{S}_i, i \in I$ is a closed, bounded, convex, non-empty set and $\mathcal{W}$ is a convex closed set, then

$$\text{conv}\left(\bigcup_{i \in I} \mathcal{P}_i\right) = \text{proj}_{\mathbf{x}}(\mathcal{P}) = \text{cl conv}\left(\bigcup_{i \in I} \mathcal{P}_i\right).$$

Proof. See [31, Proposition 2.3.5]. □

Using the characterization of the convex hull of the union of convex sets defined by cones, we can define the Hull Reformulation of the GDP-Cone as follows:

The formulation for the HR of a GDP is as follows:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{v},\mathbf{y}} \; & f(\mathbf{x}) \\
\text{s.t.} \; & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
& \mathbf{x} = \sum_{i \in D_k} \mathbf{v}_{ik}, \quad k \in K \\
& \mathbf{A}_{ik}\mathbf{v}_{ik} \succcurlyeq_{\mathcal{K}_{ik}} y_{ik}\mathbf{b}_{ik}, \quad k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ik} = 1, \quad k \in K \\
& O\mathbf{y} \leq \mathbf{o} \\
& \mathbf{x}^l y_{ik} \leq \mathbf{v}_{ik} \leq \mathbf{x}^u y_{ik} \\
& \mathbf{x} \in \mathbb{R}^n \\
& \mathbf{v}_{ik} \in \mathbb{R}^n, \quad k \in K, i \in D_k \\
& y_{ik} \in \{0,1\}, \quad k \in K, i \in D_k.
\end{aligned}
\tag{HR-Cone}
$$

This problem is of the form of MICP, and more notably uses the same cones within the disjunctions, $\mathcal{K}_{ik}$ in the extended formulation. Contrary to problem HR, problem HR-Cone does not require an approximation of the perspective function. Considering the HR reformulation as an optimization problem defined over convex cones allows exploiting the tight continuous relaxation of these problems while efficiently addressing the perspective reformulation's exact form.

To show several functions that appear in the normal context of convex MINLP can be reformulated as the standard cones described in Section 8.2.1, as well as their perspective function, we include Table 8.1. The conic representations at Table 8.1 are not unique and are given as a practical guide for implementing convex constraints using cones. Notice that applying the perspective reformulation, we recover the results found by several authors on stronger formulations for convex constraints activated through indicator variables. Such

examples include the epigraph of quadratic functions [48] and the epigraph of power functions with positive rational exponents [297]. The conic reformulation gives a natural and systematic procedure to perform extended reformulations [42], which have proved to be helpful in solution methods for mixed-integer convex programs [63, 269].

To use the HR reformulation of GDP using conic constraints, it suffices to perform the take the perspective on its cones, i.e., for variables $\mathbf{z}$ defined over the cone $\mathcal{K}$ its perspective becomes $(y\frac{\mathbf{z}}{y}) \in \mathcal{K}$. This has a considerable advantage, given that the HR reformulation is representable in the same cones like the ones used within the disjunctions.

8.4 Computational results

The computational results in this chapter include the comparison of different mixed-integer reformulations of GDP problems. The sources of these GDP problems are applications in Process Systems Engineering (PSE) and Machine Learning (ML), besides some randomly generated instances to benchmark the different solution methods. Each different reformulation was tackled using MINLP solvers. All the problems were implemented in the General Algebraic Modeling Software GAMS [248] 28.2. The solvers used for this comparison are BARON [269] 19.7, CPLEX [52] 12.9, and KNITRO [49] 11.1 for convex MINLP. We also use as a MICP solver MOSEK [35] 9.0.98, using two different algorithms implemented within it for solving relaxations of the conic problems, either an interior-point solution or through an outer-approximation approach (`MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION` set as `MSK_OFF` or `MSK_ON`), denoted MOSEK-IP and MOSEK-OA, respectively. Given the sophistication of these solvers, the effects of the different problem formulations can be shadowed by the use of heuristics within them. To better observe the performance difference given by the problem formulation, we use the Simple Branch & Bound SBB [300] implementation

Table 8.1: Common convex constraints $\mathbf{h}(\mathbf{z}) \leq \mathbf{0}$ and perspective functions $\widetilde{\mathbf{h}}(\mathbf{z}, y) \leq \mathbf{0}$ with conic reformulation.

$\mathbf{h}(\mathbf{z}) \leq \mathbf{0}$	$\mathbf{z} \in \mathcal{K}_{\mathbf{h}}$	$\widetilde{\mathbf{h}}(\mathbf{z}, y) \leq \mathbf{0}$	$(\mathbf{z}, y) \in \mathcal{K}_{\widetilde{\mathbf{h}}} = (\frac{\mathbf{z}}{y}) \in \mathcal{K}_{\mathbf{h}}$	Notes
$A\mathbf{x} + \mathbf{b} \leq 0$	$(A\mathbf{x} + \mathbf{b}) \in \mathbb{R}^m_+$	$A\mathbf{x} + \mathbf{b}y \leq 0$	$(A\mathbf{x} + \mathbf{b}y) \in \mathbb{R}^m_+$	Union of polyhedra [2, 283]
$x^2 - t \leq 0$	$(0.5, t, x) \in \mathcal{Q}^3_r$	$x^2 - ty \leq 0$	$(0.5y, t, x) \in \mathcal{Q}^3_r$	
$\mathbf{x}^\top \mathbf{x} - t \leq 0$	$(0.5, t, \mathbf{x}) \in \mathcal{Q}^{n+2}_r$	$\mathbf{x}^\top \mathbf{x} - ty \leq 0$	$(0.5y, t, \mathbf{x}) \in \mathcal{Q}^{n+2}_r$	
$1/x - t \leq 0, x > 0$	$(x, t, \sqrt{2}) \in \mathcal{Q}^3_r$	$y^2 - tx \leq 0, x > 0$	$(x, t, \sqrt{2}y) \in \mathcal{Q}^3_r$	
$\lvert x\rvert^p - t \leq 0, p > 1$	$(t, 1, x) \in \mathcal{P}^{1/p, 1-1/p}_3$	$\lvert x\rvert^p / y^{p-1} - t \leq 0, p > 1$	$(t, y, x) \in \mathcal{P}^{1/p}_3$	
$1/x^p - t \leq 0, x > 0, p > 1$	$(t, x, 1) \in \mathcal{P}^{1/(1+p), p/(1+p)}_3$	$y^{1+p}/x^p - t \leq 0, x > 0, p > 1$	$(t, x, y) \in \mathcal{P}^{1/(1+p)}_3$	
$x^{a/b} - t \leq 0, x > 0, a \geq b > 0, a, b \in \mathbb{Z}$	$(t, 1, x) \in \mathcal{P}^{b/a, 1-b/a}_3$	$x^a - t^b y^{a-b} \leq 0, x > 0, a \geq b > 0, a, b \in \mathbb{Z}$	$(t, y, x) \in \mathcal{P}^{b/a}_3$	Cut strengthening for rational power constraints with indicators[67]
$e^x - t \leq 0$	$(t, 1, x) \in \mathcal{K}_{exp}$	$ye^{x/y} - t \leq 0$	$(t, y, x) \in \mathcal{K}_{exp}$	
$t - \log(x) \leq 0$	$(x, 1, t) \in \mathcal{K}_{exp}$	$t - y\log(x/y) \leq 0$	$(x, y, t) \in \mathcal{K}_{exp}$	
$1/\log(x) - t \leq 0, x > 1$	$(u, t, \sqrt{2}) \in \mathcal{Q}^3_r$, $(x, 1, u) \in \mathcal{K}_{exp}$	$y/\log(x/y) - t \leq 0, x > 1$	$(u, t, \sqrt{2}y) \in \mathcal{Q}^3_r$, $(x, y, u) \in \mathcal{K}_{exp}$	
$xe^x - t \leq 0, x \geq 0$	$(0.5, u, x) \in \mathcal{Q}^3_r$, $(t, x, u) \in \mathcal{K}_{exp}$	$xe^{x/y} - t \leq 0, x \geq 0$	$(0.5y, u, x) \in \mathcal{Q}^3_r$, $(t, x, u) \in \mathcal{K}_{exp}$	
$a_1^{x_1} \cdots a_n^{x_n} - t \leq 0, a_1 > 0$	$(t, 1, \sum_{i \in [\![n]\!]} x_i \log a_i) \in \mathcal{K}_{exp}$	$ya_1^{x_1/y} \cdots a_n^{x_n/y} - t \leq 0, a_1 > 0$	$(t, y, \sum_{i \in [\![n]\!]} x_i \log a_i) \in \mathcal{K}_{exp}$	
$\log(1 + e^x) - t \leq 0$	$(u, 1, x - t) \in \mathcal{K}_{exp}$, $(v, 1, -t) \in \mathcal{K}_{exp}$, $u + v \leq 1$	$y\log(1 + e^{x/y}) - t \leq 0$	$(u, y, x - t) \in \mathcal{K}_{exp}$, $(v, y, -t) \in \mathcal{K}_{exp}$, $u + v \leq y$	
$-\log(1/(1 + e^{-\theta^\top \mathbf{x}})) - t \leq 0$	$(u, 1, -\theta^\top \mathbf{x} - t) \in \mathcal{K}_{exp}$, $(v, 1, -t) \in \mathcal{K}_{exp}$, $u + v \leq 1$	$-y\log(1/(1 + e^{-\theta^\top \mathbf{x}/y})) - t \leq 0$	$(u, y, -\theta^\top \mathbf{x} - t) \in \mathcal{K}_{exp}$, $(v, y, -t) \in \mathcal{K}_{exp}$, $u + v \leq y$	Logistic cost function
$x\log(x) + t \leq 0$	$(1, x, t) \in \mathcal{K}_{exp}$	$x\log(x/y) + t \leq 0$	$(y, x, t) \in \mathcal{K}_{exp}$	Entropy and relative entropy [298]
$\log(1 + 1/x) - t \leq 0, x > 0$	$(x + 1, u, \sqrt{2}) \in \mathcal{Q}^3_r$, $(1 - u, 1, t) \in \mathcal{K}_{exp}$	$y\log(1 + y/x) - t \leq 0, x, y > 0$	$(x, y + x, u) \in \mathcal{K}_{exp}$, $(x + y, x, v) \in \mathcal{K}_{exp}$, $t + u + v = 0$	
$\lVert A\mathbf{x} + \mathbf{b}\rVert_2 - \mathbf{c}^\top \mathbf{x} + d \leq 0$	$(\mathbf{c}^\top \mathbf{x} + d, A\mathbf{x} + \mathbf{b}) \in \mathcal{Q}^{m+1}$	$\lVert A\mathbf{x} + \mathbf{b}\rVert_2 - \mathbf{c}^\top \mathbf{x} + dy$	$(\mathbf{c}^\top \mathbf{x} + dy, A\mathbf{x} + \mathbf{b}) \in \mathcal{Q}^{m+1}$	Robust constraint with ellipsoidal uncertainty set [299]
$\log\left(\sum_{i \in [\![n]\!]} e^{x_i}\right) - t \leq 0$	$(u_i, 1, x_i - t) \in \mathcal{K}_{exp}, i \in [\![n]\!]$, $\sum_{i \in [\![n]\!]} u_i \leq 1$	$y\log\left(\sum_{i \in [\![n]\!]} e^{x_i/y}\right) - t \leq 0$	$(u_i, y, x_i - t) \in \mathcal{K}_{exp}, i \in [\![n]\!]$, $\sum_{i \in [\![n]\!]} u_i \leq y$	Log-sum-exp
$\lVert\mathbf{x}\rVert_1 - t = \sum_{i \in [\![n]\!]} \lvert x_i\rvert - t \leq 0$	$(u_i, x_i) \in \mathcal{Q}^2, i \in [\![n]\!]$, $t = \sum_{i \in [\![n]\!]} u_i$	$\lVert\mathbf{x}\rVert_1 - t = \sum_{i \in [\![n]\!]} \lvert x_i\rvert - t \leq 0$	$(u_i, x_i) \in \mathcal{Q}^2, i \in [\![n]\!]$, $t = \sum_{i \in [\![n]\!]} u_i$	ℓ_1-norm epigraph
$\lVert\mathbf{x}\rVert_2 - t = (\sum_{i \in [\![n]\!]} x_i^2)^{1/2} - t \leq 0$	$(t, \mathbf{x}) \in \mathcal{Q}^{n+1}$	$\lVert\mathbf{x}\rVert_2 - t = (\sum_{i \in [\![n]\!]} x_i^2)^{1/2} - t \leq 0$	$(t, \mathbf{x}) \in \mathcal{Q}^{n+1}$	ℓ_2-norm epigraph
$\lVert\mathbf{x}\rVert_p - t = (\sum_{i \in [\![n]\!]} x_i^p)^{1/p} - t \leq 0, p > 1$	$(u_i, t, x_i) \in \mathcal{P}^{1/p}_3, i \in [\![n]\!]$, $t = \sum_{i \in [\![n]\!]} u_i$	$\lVert\mathbf{x}\rVert_p - t = (\sum_{i \in [\![n]\!]} x_i^p)^{1/p} - t \leq 0, p > 1$	$(u_i, t, x_i) \in \mathcal{P}^{1/p}_3, i \in [\![n]\!]$, $t = \sum_{i \in [\![n]\!]} u_i$	ℓ_p-norm epigraph
$\lVert\mathbf{x}\rVert_{-1} - t = n(\sum_{i \in [\![n]\!]} x_i^{-1})^{-1} - t \leq 0, \mathbf{x} > \mathbf{0}$	$(u_i, x_i, t) \in \mathcal{Q}^3_r, i \in [\![n]\!]$, $nt/2 = \sum_{i \in [\![n]\!]} u_i$	$\lVert\mathbf{x}\rVert_{-1} - t = n(\sum_{i \in [\![n]\!]} x_i^{-1})^{-1} - t \leq 0, \mathbf{x} \geq \mathbf{0}$	$(u_i, x_i, t) \in \mathcal{Q}^3_r, i \in [\![n]\!]$, $nt/2 = \sum_{i \in [\![n]\!]} u_i$	Harmonic mean ℓ_{-1}-norm epigraph
$(\prod_{i \in [\![n]\!]} x_i)^{1/n} - \lvert t\rvert \leq 0, \mathbf{x} > \mathbf{0}$	$(u_i, x_i, u_{i+1}) \in \mathcal{P}^{1-1/i}_3, i \in \{2, \ldots, n\}$, $u_1 = x_1, u_{n+1} = t$	$(\prod_{i \in [\![n]\!]} x_i)^{1/n} - \lvert t\rvert \leq 0, \mathbf{x} > \mathbf{0}$	$(u_i, x_i, u_{i+1}) \in \mathcal{P}^{1-1/i}_3, i \in \{2, \ldots, n\}$, $u_1 = x_1, u_{n+1} = t$	Geometric mean
$\prod_{i \in [\![n]\!]} x_i^{\alpha_i} - \lvert t\rvert \leq 0, \mathbf{x} > \mathbf{0}$, $\alpha > \mathbf{0}, \sum_{i \in [\![n]\!]} \alpha_i = 1$	$(u_i, x_i, u_{i+1}) \in \mathcal{P}^{1-\beta_i}_3$, $\beta_i = \alpha_i / (\sum_{j \in [\![i]\!]} \alpha_j), i \in \{2, \ldots, n\}$, $u_1 = x_1, u_{n+1} = t$	$\prod_{i \in [\![n]\!]} x_i^{\alpha_i} - \lvert t\rvert \leq 0, \mathbf{x} > \mathbf{0}$, $\alpha > \mathbf{0}, \sum_{i \in [\![n]\!]} \alpha_i = 1$	$(u_i, x_i, u_{i+1}) \in \mathcal{P}^{1-\beta_i}_3$, $\beta_i = \alpha_i / (\sum_{j \in [\![i]\!]} \alpha_j), i \in \{2, \ldots, n\}$, $u_1 = x_1, u_{n+1} = t$	Weighted Geometric Mean

Symbols a, b, c, d, p stand for scalar parameters, α, β, θ are vector parameters, $\mathbf{x} \in \mathbb{R}^n$ and $t \in \mathbb{R}$ are variables, and u, v stand for additional variables added for the conic reformulation. $y \in [0, 1]$ is the perspective variable. This table is heavily influenced by the Conic Modeling Cleansheet [35]

in GAMS and solve the respective continuous subproblems using gradient-based interior-point NLP solver KNITRO [49] 11.1, and MOSEK [35] 9.0.98 for the conic subproblems. All experiments were run on a single thread of an Intel® Xeon® CPU (24 cores) 2.67 GHz server with 128GB of RAM running Ubuntu. The termination criteria were a time limit of 3600 seconds or a relative optimality gap of $\epsilon_{rel} = 10^{-5}$. Unless otherwise stated, the conic reformulation of the constraints was written explicitly, meaning that the auxiliary variables required by the reformulation were introduced to the problem directly. This is a weakness identified in the conic programming interface in GAMS, where the conic structure identification is not made automatically. The definition of the cones, although trivial, had to be done manually.

For all these GDP problems, the Big-M and HR reformulations are presented. When neccesary, the conic representations for both cases, i.e., Big-M-Cone and HR-Cone, are presented separately from the algebraic description, i.e., Big-M and HR. The algebraic description of the HR included the ε-approximation (8.19) proposed in [62] to avoid numerical difficulties, denoted HR-ε. We use the recommended value of $\varepsilon = 10^{-4}$ for all the cases presented herein. We also implemented the perspective function directly and used the ε-approximation (8.18). However, the results proved that, in general, the numerical challenges associated with the perspective function were better handled using the approximation in (8.19). Hence, we do not include the results of the direct implementation of the perspective function or the approximation given by (8.18), and only present those from using the approximation in (8.19) in this chapter. However, the interested reader can find the complete results in the online repository.

The mixed-integer Big-M and Hull reformulations of some of these instances are present in the benchmarking libraries MINLPLib [100] and MINLP.org [301]. They have been widely used for MINLP solver benchmarks [26, 62, 129, 263]. This applies in particular for the

PSE applications, Constrained Layout (`CLay*`), Process Networks (`proc*`), and Retrofit Synthesis instances (`RSyn*` and `Syn*`). This motivates the study on these well-known instances.

Moreover, there has been recent interest from the Machine Learning (ML) community in using rigorous methods for non-convex optimization, contrary to heuristics based on convex relaxations. Even considering the performance cost of the rigorous methods, the optimal solution to the original non-convex optimization problem is informative and valuable within an ML framework [302]. Finding the optimal values of the parameters of a probability distribution such that a likelihood estimator is maximized, i.e., training, is known as Expectation-Maximization (EM) in ML [303]. When the data labels are incomplete, the general problem can be stated as learning from weakly labeled data [304]. While performing the training, the assignment of the labels is naturally representable through disjunctions, giving rise to mixed-integer programs. For example, there has been a recent interest in tackling the clustering problem using mixed-integer programming [302]. Optimally guaranteed solutions to a problem similar to 8.29 leads to better results measured by the performance of the ML model arising from the clustering compared to local-optimization approaches to the EM problem. The ML instances on k-mean clustering (`kClus*`) and logistic regression (`LogReg*`) are inspired on problems proposed in the literature but are randomly generated for this chapter.

The following results are presented in two subsections, one considering "quadratic" problems that can be formulated using second-order and rotated second-order cones, and the second one with problems modeled through the exponential cone. Each formulation includes linear constraints, which can be managed by both gradient-based and conic mixed-integer convex programming solvers. All the results from this chapter are available in an

open-access repository*.

It is worth mentioning that we report the nodes required by each solver. The definition of a node might vary for every solver, and a detailed description of each case is not widely available. To better control these reports, we compare SBB as a central manager for the branching procedures. In this last procedure, we can guarantee that each node is the solution to a continuous convex optimization problem.

8.4.1 Quadratic problems

The three families of instances presented herein are the Constrained Layout problem, a k-mean clustering optimization problem, and randomly generated instances. All these problems share the characteristic that the constraints within the disjunctions are representable via second-order and rotated second-order cones.

The mixed-integer reformulations of these problems were implemented as in Big-M and HR, both the HR-ε and HR-Cone. Notice that in the case of second-order cone, the explicit definition of the cone can be replaced by the inequality [48]

$$x^2 - ty \leq 0 \iff \|(2x, y-t)\|_2 \leq y + t, \tag{8.26}$$

that avoids the variable multiplication ty and improves the performance of gradient-based solvers like IPOPT and KNITRO. When implementing this alternative to the exact representation of the perspective function, it improves the performance of KNITRO slightly, at the expense of a significant decrease in BARON's performance. Therefore, the implementation results are left out of this chapter, although they are included in the repository for reference.

The examples in this section had constraints in their disjunctions directly identified as a cone by MOSEK in the GAMS interface. This might not be the general case, with the cones

*https://github.com/bernalde/conic_disjunctive

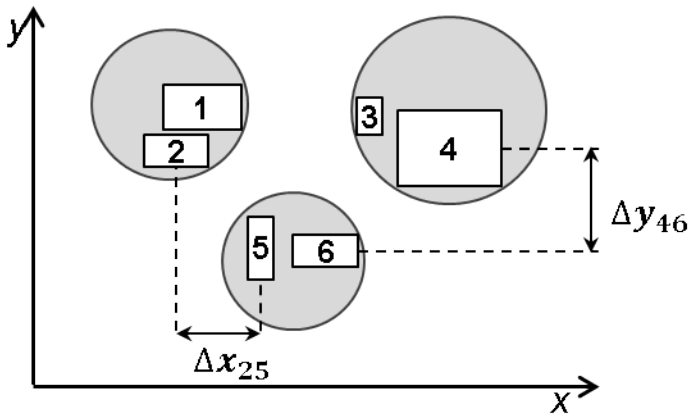


Figure 8.1: Schematics of Constrained Layout Problem

needing to be explicitly written for MOSEK to process them. This allowed the Big-M instances to be written in their algebraic form. Simultaneously, the HR reformulation required the explicit introduction of additional constraints for the conic form to be accepted by the GAMS-MOSEK interface. CPLEX, on the other hand, can identify and transform certain general quadratic constraints into general and rotated second-order cones automatically.

Below we present the examples considered as convex quadratic GDPs.

8.4.1.1 Constrained layout problem

The constrained layout problem is concerned with the minimization of the connection costs among non-overlapping rectangular units. These units need to be packed within a set of fixed circles. Figure 8.1 illustrates the constrained layout problem. It can be formulated as the following convex GDP [233]:

$$
\begin{aligned}
\min_{\delta\mathbf{x},\delta\mathbf{y},\mathbf{x},\mathbf{y},\mathbf{W},\mathbf{Y}} & \sum_{i,j\in N} c_{ij}(\delta x_{ij}+\delta y_{ij}) \\
\text{s.t. } & \delta x_{ij} \geq x_i - x_j && i,j\in N, i<j \\
& \delta x_{ij} \geq x_j - x_i && i,j\in N, i<j \\
& \delta y_{ij} \geq y_i - y_j && i,j\in N, i<j \\
& \delta y_{ij} \geq y_j - y_i && i,j\in N, i<j \\
& \begin{bmatrix} Y_{ij}^1 \\ x_i + L_i/2 \leq x_j - L_j/2 \end{bmatrix} \veebar \begin{bmatrix} Y_{ij}^2 \\ x_j + L_j/2 \leq x_i - L_i/2 \end{bmatrix} \\
& \veebar \begin{bmatrix} Y_{ij}^3 \\ y_i + H_i/2 \leq y_j - H_j/2 \end{bmatrix} \veebar \begin{bmatrix} Y_{ij}^4 \\ y_j + H_j/2 \leq y_i - H_i/2 \end{bmatrix} && i,j\in N, i<j \\
& \veebar_{t\in T} \begin{bmatrix} W_{it} \\ (x_i + L_i/2 - xc_t)^2 + (y_i + H_i/2 - yc_t)^2 \leq r_t^2 \\ (x_i + L_i/2 - xc_t)^2 + (y_i - H_i/2 - yc_t)^2 \leq r_t^2 \\ (x_i - L_i/2 - xc_t)^2 + (y_i + H_i/2 - yc_t)^2 \leq r_t^2 \\ (x_i - L_i/2 - xc_t)^2 + (y_i - H_i/2 - yc_t)^2 \leq r_t^2 \end{bmatrix} && i\in N \qquad (8.27) \\
& Y_{ij}^1 \veebar Y_{ij}^2 \veebar Y_{ij}^3 \veebar Y_{ij}^4 && i,j\in N, i<j \\
& \veebar_{t\in T} W_{it} && i\in N \\
& 0 \leq x_i \leq x_i^u && i\in N \\
& 0 \leq y_i \leq y_i^u && i\in N \\
& \delta x_{ij}, \delta y_{ij} \in \mathbb{R}_+ && i,j\in N, i<j \\
& x_i, y_i \in \mathbb{R} && i\in N \\
& Y_{ij}^1, Y_{ij}^2, Y_{ij}^3, Y_{ij}^4 \in \{False, True\} && i,j\in N, i<j \\
& W_{it} \in \{False, True\} && i\in N, t\in T
\end{aligned}
$$

where the coordinate centers of each rectangle $i \in N$ are represented through variables x_i, y_i, the distance between two rectangles $i, j \in N, i < j$ is given by variables δx_{ij} and δy_{ij}, and c_{ij} is the cost associated with it. The first disjunction allows for the non-overlapping of the rectangles, while the second one ensures that each rectangle is inside of one of the circles $t \in T$, whose radius is given by r_t and center specified by coordinates (xc_t, yc_t).

The constraints in the second disjunction are representable through a quadratic cone as follows:

$$\begin{aligned} &(x_i \pm L_i/2 - xc_t)^2 + (y_i \pm H_i/2 - yc_t)^2 \leq r_t^2 \\ &\iff (r_t, x_i \pm L_i/2 - xc_t, y_i \pm H_i/2 - yc_t) \in Q^3. \end{aligned} \tag{8.28}$$

Seven different problem instances are defined through the variation of the number of circular areas to fit in the rectangle $|T|$ and the number of possible rectangles N, which each instance being denoted `CLay|T||N|`.

8.4.1.2 k-means clustering

The k-means clustering problem is an optimization problem that appears in unsupervised learning. This problem minimizes the total distance of a set of points to the center of k clusters, varying the center's position and the assignment of which center determines the distance to each point. This problem is solved usually through heuristics, although these approaches do not guarantee the quality of the solution.

Recently, Papageorgiou and Trespalacios [305] proposed a GDP formulation for the

k-mean clustering problem, also used in [306]. The problem formulation reads as follows:

$$\begin{aligned}
&\min_{\mathbf{c},\mathbf{d},\mathbf{Y}} \sum_{i\in N} d_i \\
&\text{s.t.} \quad c_{k-1,1} \le c_{k,1}, \quad k \in \{2,\dots,|K|\} \\
&\bigvee_{k\in K} \begin{bmatrix} Y_{ik} \\ d_i \ge \sum_{j\in D} (p_{ij} - c_{kj})^2 \end{bmatrix}, \quad i \in N, k \in K \\
&\veebar_{k\in K} Y_{ik}, \quad i \in N \\
&\mathbf{d} \in \mathbb{R}_+^{|N|} \\
&\mathbf{c} \in \mathbb{R}^{|K|\times|D|} \\
&Y_{ik} \in \{False, True\}, \quad i \in N, k \in K,
\end{aligned} \tag{8.29}$$

where N is the set of points given in $|D|$ dimensions, whose coordinates are given by $\mathbf{p} \in \mathbb{R}^{|N|\times|D|}$. The variables are the center coordinates $\mathbf{c}$, and the squared distances of each point to its closest center are denoted by $\mathbf{d}$. The first constraint is a symmetry-breaking constraint. An arbitrary increasing ordering in the first dimension is taken for the centers. The disjunctions determine with which center k will the distance to point i be computed, given that $Y_{ik} = True$.

The constraint for each disjunction $i \in N, k \in K$ is naturally representable as a rotated second-order cone

$$d_i \ge \sum_{j\in D} (p_{ij} - c_{kj})^2 \iff (0.5, d_i, p_{i1} - c_{k1}, \dots, p_{i|D|} - c_{k|D|}) \in Q_r^{2+|D|}. \tag{8.30}$$

We vary the number of clusters $|K| \in \{3,5\}$, the number of given points $|N| \in \{10,20\}$, and the dimensions of those points $|D| \in \{2,3,5\}$ leading to instance `kClus_|K|_|N|_|D|_x`. `x` in this case denotes one of the random instances generated. For this problem, we include 10 instances for each case varying the point coordinates $p_{ij} \in U[0,1], i \in N, k \in K$.

8.4.1.3 Random examples

We generate random quadratic GDP problems to test further the reformulations proposed in this chapter. The random quadratic GDP problems are of the form,

$$
\begin{aligned}
&\min_{\mathbf{x},\mathbf{Y}} \mathbf{c}^\top \mathbf{x} \\
&\text{s.t.} \quad \bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ \sum_{j \in [\![n]\!]} \left(a'_{ijk} x_j^2 + a''_{ijk} x_j \right) + a'''_{ik} \leq 1 \end{bmatrix}, \quad k \in K \\
&\quad \veebar_{i \in D_k} Y_{ik}, \quad k \in K \\
&\quad \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u \\
&\quad \mathbf{x} \in \mathbb{R}^n \\
&\quad Y_{ik} \in \{False, True\}, \quad k \in K, i \in D_k,
\end{aligned}
\tag{SOCP-rand-GDP}
$$

where the upper and lower bounds of variables $\mathbf{x}$, $\mathbf{x}^l$ and $\mathbf{x}^u$, are set at -100 and 100, respectively.

The constraint in each disjunct is representable as a rotated second-order cone,

$$
\begin{aligned}
&\sum_{j \in [\![n]\!]} \left(a'_{ijk} x_j^2 + a''_{ijk} x_j \right) + a'''_{ik} \leq 1 \\
&\iff \begin{cases} \left(0.5, t, \sqrt{a'_{ijk}} x_j, \ldots, \sqrt{a'_{ink}} x_n \right) \in Q_r^{n+2} \\ t + \sum_{j \in D} a''_{ijk} x_j + a'''_{ik} \leq 1 \end{cases},
\end{aligned}
\tag{8.31}
$$

The different random instances were generated by varying the number of disjunctions $|K| \in \{5, 10\}$, the number of disjunctive terms at each disjunction $|D_k| \in \{5, 10\}$, and the dimensions of the $\mathbf{x}$ variables $n \in \{5, 10\}$ leading to instance `socp_random_|K|_|`D_k`|_n_x`. `x` denotes the index of the random variable generated. 10 instances are generated for each case varying the parameters within bounds $[l, u]$ by sampling the random uniform distributions $U[l, b]$ as follows: $a'_{ijk} \in U[0.01, 1], a''_{ijk} \in U[-1, 1], a'''_{ik} \in U[-1, 1], c_j \in U[-1000, 1000], i \in D_k, k \in$

$K, j \in [\![n]\!]$. We also include instances `socp_random_2_2_2_x`, that represent the illustrative example in [305].

Notice that the k-Mean clustering formulation is a particular case of these randomly generated GDPs. In particular, if we set $\mathbf{a}' = 1, \mathbf{a}'' = 2\mathbf{p}, \mathbf{a}''' = \mathbf{p}^\top\mathbf{p}, \mathbf{c} = 1$ we recover the k-Mean clustering problem.

8.4.1.4 Results

We generate a total of 217 GDP problems, which are transformed through a Big-M and HR, this last using both HR-ε and HR-Cone. The main results are presented in Table 8.2, where the solution times and nodes for the Big-M, HR, and HR-Cone reformulations using different commercial solvers are included. Consider that the HR-ε formulation introduces non-linearities in the formulation, preventing CPLEX and MOSEK from addressing it.

In general, we can observe that CPLEX applied to the Big-M reformulation has the best performance for the `CLay*` and `kClus*` instances when considering runtime. BARON applied to the same Big-M formulation returns the optimal solution with the least number of explored nodes for the constrained layout problems. This shows how the mature solvers for mixed-integer programming have implemented useful heuristics to work with Big-M formulation; the ubiquity of these formulations among practitioners motivates their development of techniques to work with these problems efficiently. An example is that CPLEX identifies the Big-M formulation and internally treats its constraints through specialized branching rules derived from indicator constraints [52]⋆.

When comparing only the Big-M formulation solution, BARON solves all the problems with the least number of nodes for all instances. This corresponds to the main focus of BARON on solving more "meaningful" nodes for the problem. However, it might

⋆IBM documentation

incur a performance cost [47]. This observation also appears when comparing all the HR formulation results, where BARON required the fewest number of nodes. This applied to the HR-ε and the HR-Cone formulations.

In terms of runtime, when comparing the HR formulations, we observe that CPLEX is the most performing solver for the `CLay*` instances, while MOSEK-IP is the one for the `kClus*` and `socp_random*` problems. Notice that either of these solvers could be applied to the HR-ε formulation, proving that using a conic formulation of the HR problem opens the possibility of using solvers that can better exploit the problem structure. Even for general nonlinear solvers, such as BARON, the use of the conic reformulation provides a performance improvement, given that the lifted reformulation can be exploited for tighter relaxations within the solver [269]. On the other hand, solvers based on nonlinear B&B, where each node is solved with a general NLP algorithm such as interior-point methods, such as KNITRO, can worsen their performance when using the conic reformulation. The non-differentiability of the cones, together with the larger subproblem sizes, can cause such a negative impact. This can be alleviated by taking advantage of the conic structure, something that KNITRO has implemented as part of their presolve capabilities [49]*.

A better view of the general performance of the different solvers is given in Figures 8.2 and 8.3. These figures present performance profiles accounting for the number of problems solved to a given gap of the optimal solution (0.1% in this case) within a time or node limit. In general, as seen in Figure 8.3, the performance concerning nodes is superior for all solvers when using the HR, except for BARON. This is expected given the tightness of this formulation. Moreover, in terms of solution time, both algorithms used in MOSEK improve their performance when using a HR compared to the Big-M case. This shows that when modeling disjunctive conic programs, the Hull reformulation is preferable for this solver.

*KNITRO v11 presentation

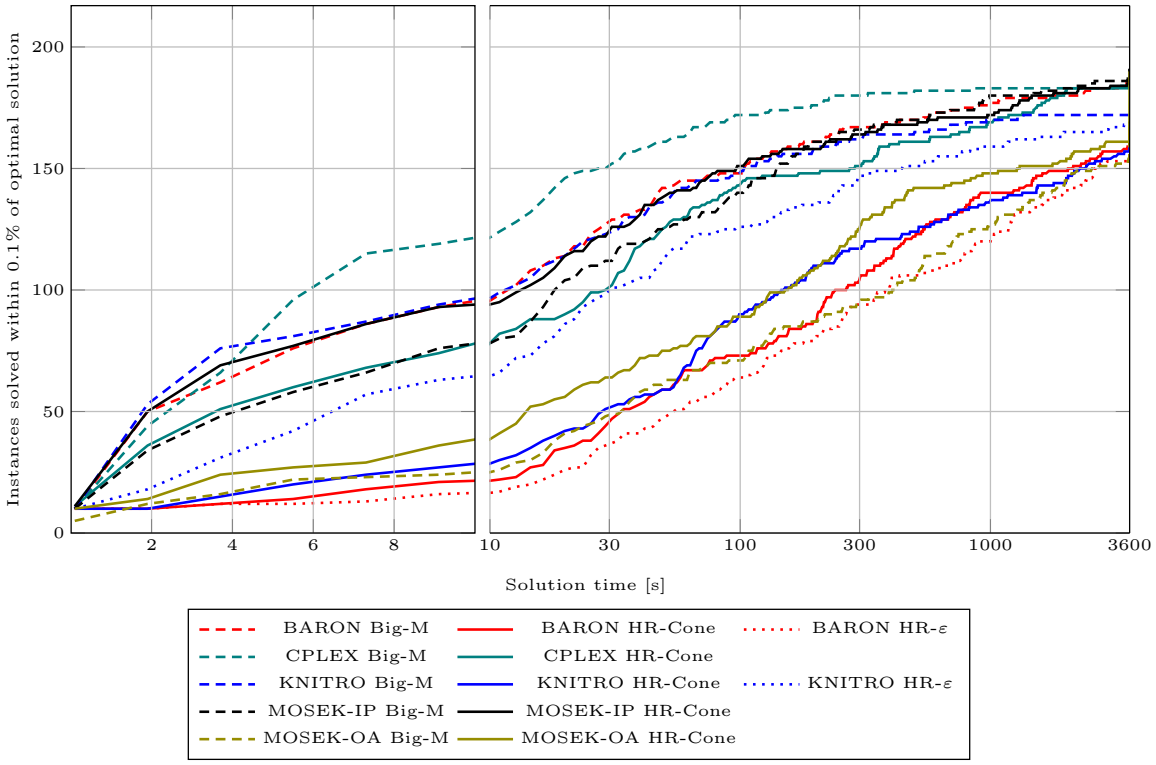


Figure 8.2: Time performance profile for quadratic instances using the different GDP reformulations and commercial solvers.

The other solvers worsen their performance when using the extended formulations in terms of solution time.

Of the total 217 instances, the solver that solved the most instances to within 0.1% of the best-known solution was MOSEK-IP with 191, both using the Big-M and HR-Cone formulations. The alternative that solved the fewest instances was KNITRO applied to the HR-Cone formulation, solving only 160.

8.4.2 Exponential problems

As examples of problems representable using the exponential cone $\mathcal{K}_{exp}$, we present four families of problems: Process networks, Retrofit Synthesis Problems, Logistic Regression,

Table 8.2: Results for Quadratic GDPs using different mixed-integer reformulations and solvers. The least time and fewest nodes results for each instance within a reformulation are italicized. The best results overall are bolded.

	Big-M									
	BARON		CPLEX		KNITRO		MOSEK-OA		MOSEK-IP	
Instance	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes
CLay0203	0.49	***9***	0.35	56	0.80	175	0.61	145	0.78	261
CLay0204	0.73	*4*	***0.58***	557	7.19	1921	4.74	2589	10.88	2357
CLay0205	5.78	*61*	***3.96***	3972	102.76	16540	27.96	15559	177.25	20115
CLay0303	1.43	***79***	0.61	203	2.22	347	1.36	367	1.07	271
CLay0304	5.98	***155***	***1.71***	842	15.33	1647	15.19	3313	59.30	6865
CLay0305	7.84	***41***	***4.04***	4629	99.68	14878	112.42	38589	139.41	12601
CLay0405	7.96	***41***	***4.07***	4629	97.83	14878	112.22	38589	138.11	12601
kClus_3_10_2_*	4.02±1.68	*186±102*	***0.54±0.32***	1053±507	1.74±.70 (.07%)	698±308	18.51±9.41	1846±554	2.56±0.56	916±206
kClus_3_10_3_*	9.87±8.32	*412±407*	***0.83±0.28***	1997±961	4.10±2.14 (.05%)	1613±928	88.93±36.88	3445±1093	5.17±2.06	1589±641
kClus_3_10_5_*	28.46±13.80	*880±511*	***4.36±3.62***	7555±3218	15.37±5.62 (.04%)	5119±1893	558.13±234.97	8354±2927	18.33±4.96	4322±1189
kClus_3_20_2_*	50.86±25.61	*1537±948*	***5.92±3.17***	19326±11194	91.70±52.41 (.11%)	20219±10165	1686.86±714.65	66645±26505	65.65±40.45	13118±8208
kClus_3_20_3_*	564.54±641.67	*12481±16401*	***85.29±87.60 (.51%)***	261404±268823	1179.82±824.21 (12.48%)	130637±71286	3600+ (156.09%)	61030±12564	812.72±976.49	133093±161988
kClus_3_20_5_*	3302.28±591.02 (67.71%)	*27529±17966*	***516.73±49.54 (42.03%)***	1000000+	2950.01±41.10 (85.39%)	200000+	3600+ (571.42%)	24085±4234	3594.30±17.97 (35.91%)	454679±8911
kClus_5_10_2_*	23.81±13.96	*1766±1040*	***5.22±3.18***	13507±9764	31.21±13.15 (.94%)	7382±3068	383.21±174.41	11391±3369	21.33±6.70	4999±1580
kClus_5_10_3_*	65.26±54.41	*3768±2988*	***15.00±16.73***	25142±23951	57.23±25.19 (.36%)	12898±5686	1337.58±623.21	17742±6985	51.54±31.30	10204±6176
kClus_5_10_5_*	439.59±331.61	*16507±12604*	***64.80±58.28***	80939±42792	406.47±243.50 (.14%)	57827±30849	3600+ (430.70%)	15037±2151	302.44±174.68	42001±24008
kClus_5_20_2_*	2607.50±1251.24 (93.15%)	*86762±48654*	***226.88±108.77 (9.39%)***	550460±281643	2830.47±519.31 (560.88%)	191837±24425	3600+ (853.71%)	47549±9311	1758.49±915.35 (3.33%)	208111±110067
kClus_5_20_3_*	3600+ (509.02%)	*51192±12127*	***591.02±277.76 (126.18%)***	1000000+	3232.92±63.71 (514.94%)	200000+	3600+ (∞)	19394±3531	3400.14±600.12 (67.67%)	335881±60762
kClus_5_20_5_*	3600+	11893±3233	***795.61±73.33 (559.21%)***	1000000+	3600+ (674.52%)	189226±2349	3600+ (∞)	7832±1295	3600+ (233.82%)	255259±3240
socp_random_10_10_10_*	106.84±81.75	*3483±2927*	482.81±926.53	160209±293928	**33.98±12.48**	4233±1625	2772.27±829.39 (4.87%)	29369±8324	127.57±51.60	5273±2222
socp_random_10_10_5_*	***8.98±5.99***	*325±357*	14.25±4.61	3078±1477	12.55±3.01	2233±581	776.76±540.49	22688±15947	141.69±69.42	4601±2092
socp_random_10_5_10_*	12.55±8.14	*455±371*	17.23±6.77	3749±1562	***7.94±5.10***	1777±1175	684.04±352.24	9112±4511	17.12±7.81	1734±736
socp_random_10_5_5_*	***1.36±.72***	*35±36*	5.73±1.35	494±311	2.05±.85	559±305	52.39±28.37	1912±865	8.37±3.45	836±347
socp_random_2_2_2_*	0.03±0.01	*1±0*	0.02±.01	1±1	*0.02±0.00*	4±1	0.11±0.02	5±1	0.04±0.01	5±1
socp_random_5_10_10_*	7.38±3.31	*305±202*	6.18±1.80	1370±635	***3.35±1.96***	680±499	129.11±42.08	3138±939	10.66±3.80	1034±403
socp_random_5_10_5_*	1.41±.81	*56±58*	3.95±1.56	378±221	*1.42±.46*	330±156	34.08±9.66	1526±523	5.19±2.00	548±251
socp_random_5_5_10_*	***1.29±.64***	*39±32*	3.95±.88	219±130	***0.72±0.20***	160±64	17.45±5.00	569±192	1.18±.43	208±78
socp_random_5_5_5_*	0.56±0.27	*19±16*	1.91±.66	66±32	*0.53±0.14*	137±68	3.79±0.92	344±84	0.90±0.24	185±66

	HR-Cone										HR ε-approximation			
	BARON		CPLEX		KNITRO		MOSEK-OA		MOSEK-IP		BARON		KNITRO	
Instance	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes
CLay0203	2.46	109	***0.33***	127	4.27	239	0.14* (91.4%)	*69*	0.64	199	2.34	287	4.23	179
CLay0204	*2.26*	***1***	*0.68*	842	13.88	1775	2.00	1395	5.41	1579	2.29	7	18.73	1771
CLay0205	18.03	***45***	*6.38*	7552	187.79	13111	32.51	16855	67.20	9433	12.38	177	246.84	15902
CLay0303	4.70	251	***0.41***	*186*	15.32	257	0.79	329	1.76	303	11.85	4297	6.27	223
CLay0304	48.16	2351	*1.89*	*1046*	65.81	1713	5.42	1801	49.08	6081	68.00	13185	47.28	1457
CLay0305	30.11	*133*	*13.78*	10590	515.58	15955	48.19	21269	97.22	10093	24.74	231	295.43	12227
CLay0405	30.10	*133*	*13.78*	10590	515.51	15955	48.21	21269	95.99	10093	24.88	231	292.45	12227
kClus_3_10_2_*	29.22±15.33	525±483	*1.31±0.28*	936±290	56.26±17.84 (.13%)	756±337	8.17±2.76	1187±390	2.51±.77	551±177	22.75±11.80	*357±278*	6.02±1.96 (.07%)	789±276
kClus_3_10_3_*	87.10±48.70	1148±898	*7.17±2.24*	1529±567	65.00±24.18 (.06%)	1630±809	34.90±15.15	2365±800	*6.01±1.79*	1124±368	58.53±27.51	*655±450*	15.52±7.11 (.07%)	1731±916
kClus_3_10_5_*	479.73±194.92	4220±2191	45.97±15.44	5135±1642	135.97±40.76 (.04%)	4009±1351	197.68±59.99	6381±2538	*22.92±9.46*	3146±1377	287.92±137.92	*2086±1451*	48.30±12.29 (.04%)	4015±1147
kClus_3_20_2_*	821.89±582.53	6773±5222	*39.23±16.94*	19176±9424	1028.58±445.35 (.11%)	21647±10121	415.52±204.68	28496±15069	90.01±29.74	10441±3507	783.84±538.69	*5515±5066*	318.99±125.66 (.11%)	20706±8089
kClus_3_20_3_*	3496.43±254.15 (129.69%)	13494±3443	1568.36±1137.08 (6.31%)	163189±110030	3224.30±590.94 (79.77%)	50106±7991	3370.83±478.28 (39.17%)	107636±18035	*1088.85±940.14 (2.56%)*	107190±100179	3236.45±578.69 (129.32%)	*9690±3092*	2540.24±1190.91 (42.88%)	103881±43238
kClus_3_20_5_*	3600+ (452.53%)	5760±1573	3600+ (74.97%)	123015±24631	3600+ (151.18%)	36059±1622	3600+ (226.97%)	42462±9595	*3600+ (53.20%)*	266698±18613	3600+ (471.50%)	*3495±1601*	3600+ (116.23%)	101891±1345
kClus_5_10_2_*	224.28±99.44	5250±2504	25.22±33.11	5857±1821	288.74±104.15 (.84%)	8166±3467	94.17±30.15	8665±3343	*23.47±6.75*	3887±1179	177.28±84.11	*3685±1884*	74.39±37.66 (.97%)	6781±3438
kClus_5_10_3_*	774.57±534.53	13334±9416	67.89±35.31	12002±6653	530.88±275.35 (.36%)	12517±7735	287.76±101.44	11438±4146	*48.41±17.54*	*6214±2503*	516.15±245.79	7823±4470	170.73±74.84 (.39%)	12432±5611
kClus_5_10_5_*	3289.68±664.31 (66.12%)	23297±5042	886.60±517.13	76632±48317	2681.71±754.70 (5.88%)	48680±13897	2771.39±852.71 (28.37%)	42357±14044	*386.01±266.65*	32660±24156	2911.30±734.97 (176.34%)	*17516±6177*	1237.42±768.45 (.14%)	60538±36374
kClus_5_20_2_*	3600.17±.06 (511.63%)	19732±3677	*1782.80±1019.48 (20.57%)*	447233±275642	3600+ (656.11%)	37293±2459	3600+ (176.55%)	120790±23599	2089.76±984.05 (8.21%)	154490±78837	3600+ (749.47%)	*16490±3378*	3582.12±54.81 (694.41%)	118488±4948
kClus_5_20_3_*	3600+ (∞)	7425±2841	3600+ (222.05%)	169688±72270	3600+ (850.78%)	15355±1968	3600+ (729.79%)	39462±6099	*3600+ (73.28%)*	207274±16444	3600+ (947.78%)	*3976±1893*	3600+ (683.11%)	88012±2561
kClus_5_20_5_*	3600+ (∞)	2200±936	3600+ (881.16%)	*41878±19519*	3600+ (∞)	4885±939	3600+ (∞)	12659±2086	*3600+ (303.19%)*	121861±5742	3600+ (∞)	654±271	3600+ (759.28%)	54139±1713
socp_random_10_10_10_*	2365.50±889.74 (4.66%)	1531±1406	454.42±291.88	7948±6799	1967.52±1065.66	1643±1919	2344.96±1082.43 (4.06%)	2271±1004	*54.33±43.63*	1097±855	3469.74±391.88 (100.28%)	**356±146**	185.97±164.91	2776±3025
socp_random_10_10_5_*	231.09±126.89	***191±198***	27.22±17.03	735±355	131.87±73.17	539±312	201.58±147.33	1916±1260	*10.07±4.74*	429±250	1161.88±360.83	242±209	27.41±11.94	516±330
socp_random_10_5_10_*	1021.48±733.14	1038±1325	50.15±24.51	927±505	92.10±29.22	**390±172**	434.27±315.97	1037±837	*9.83±3.85*	497±215	2502.07±754.56 (2.82%)	996±603	20.08±6.33	440±195
socp_random_10_5_5_*	42.83±36.83	44±88	3.92±2.44	166±97	25.92±20.49	218±315	14.20±6.28	272±136	*1.96±.73*	150±86	95.10±70.92	***28±49***	8.22±6.27	258±370
socp_random_2_2_2_*	0.05±0.02	1±0	***0.01±0.00***	***0±0***	0.03±0.00	1±0	0.04±0.02	0±1	0.04±.01	0±1	0.05±0.02	1±1	0.03±0.00	4±2
socp_random_5_10_10_*	248.35±59.30	***146±216***	43.59±18.64	944±580	92.75±76.83	199±291	238.98±181.32	549±585	*4.33±2.99*	161±145	1011.75±306.25	192±160	16.19±7.57	205±307
socp_random_5_10_5_*	36.89±22.51	***29±31***	3.38±2.06	106±78	20.91±9.01	*78±77*	13.09±6.68	266±194	***1.31±.55***	80±64	150.07±112.83	30±28	5.74±1.99	73±58
socp_random_5_5_10_*	40.86±16.23	***21±34***	6.17±5.28	126±119	9.25±3.27	47±26	24.11±10.54	97±51	*0.98±0.32*	55±31	143.50±109.49	39±31	3.37±.87	55±37
socp_random_5_5_5_*	8.38±3.39	***5±6***	1.83±1.53	42±31	4.55±1.11	41±28	2.72±.73	59±25	***0.38±0.12***	30±15	19.59±12.41	7±6	1.91±.22	40±27

Results corresponding to k-mean clustering kClus and random instances socp_random are the average and standard deviation over 10 randomly generated instances. Values with * denote solver failure, and values with + denote the termination for reaching time or iteration limits. A gap of ∞ denotes that no feasible solution was found.

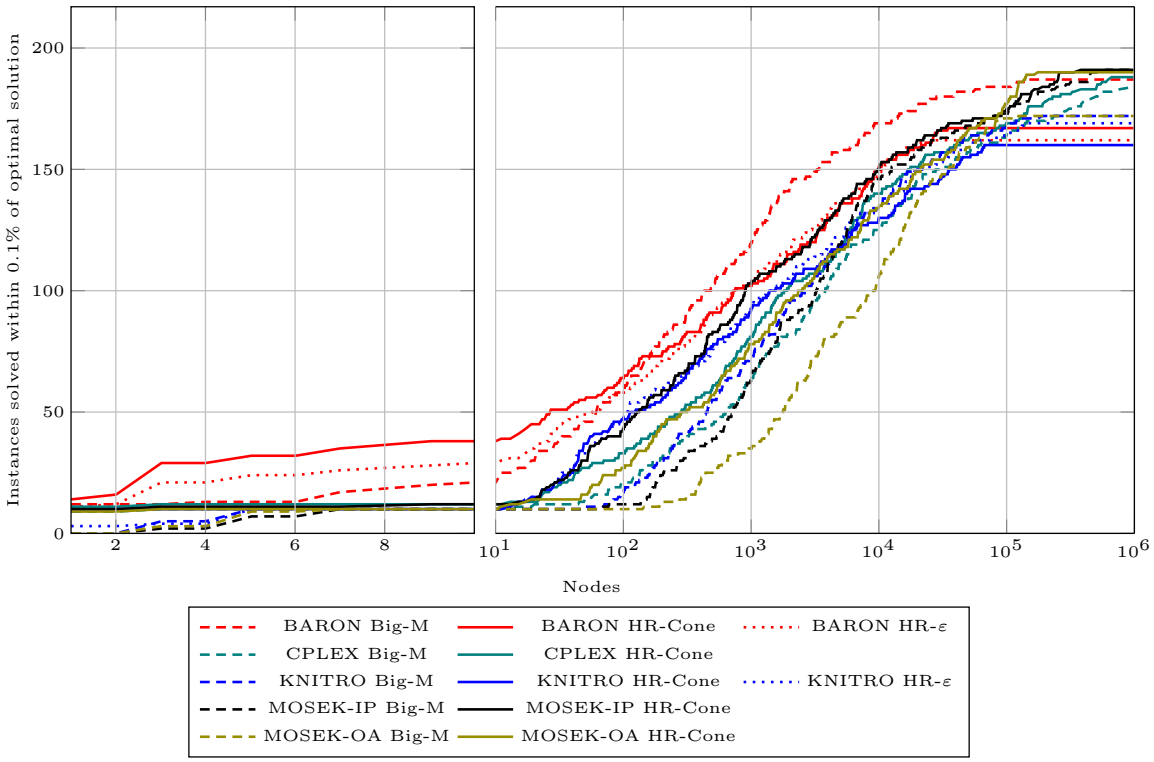


Figure 8.3: Node performance profile for quadratic instances using the different GDP reformulations and commercial solvers.

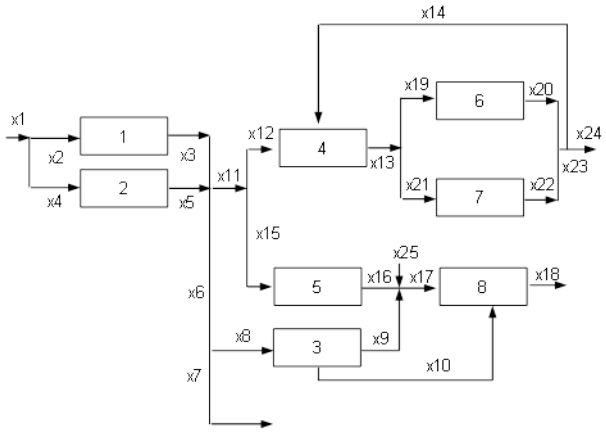


Figure 8.4: Schematic of process networks with 8 possible processes [307]

and randomly generated instances. The GAMS-MOSEK interface does not directly identify the exponential cone; therefore, we include algebraic and extended conic formulations, Big-M and Big-M-Cone respectively for Big-M. HR and HR-Cone are also tested for these problems, denoted as HR-ε using the approximation in (8.19) and HR-Cone formulation through the extended formulation required by MOSEK for the exponential cones to be correctly identified. Contrary to the previous section, the solver CPLEX was not used for these experiments since it has no capabilities to handle general nonlinear constraints beyond quadratics or exponential cones.

8.4.2.1 Process Networks

In the process network problem, we seek to maximize the profit from a process by deciding the equipment to be installed to fabricate some valuable product subject to material flows between the equipment pieces. The total cost is computed from the cost of raw materials and equipment subtracted from the product's sales. Alternative equipment pieces might induce a trade-off in terms of cost and production, defining the problem's constraints. This classical problem in process design usually considers complex models describing each

equipment piece. For this simplified case [293, 308], we assume input-output correlations for each equipment described by an exponential function. This is a simplification that still accounts for the non-linearity inherent to chemical processes. Figure 8.4 illustrates the superstructure for a process with potentially eight units presented by [307]. The problem can be modeled through the following convex GDP:

$$
\begin{aligned}
&\min_{\mathbf{c},\mathbf{x},\mathbf{Y}} \sum_{k\in K} c_k + \sum_{j\in J} p_j x_j \\
&\text{s.t.} \quad \sum_{j\in J} r_{jn} x_j \le 0, \quad n \in N \\
&\qquad \bigvee_{i\in D_k} \begin{bmatrix} Y_{ik} \\ \sum_{j\in J_{ik}} d_{ijk}(e^{x_j/t_{ijk}} - 1) - \sum_{j\in J_{ik}} s_{ijk} x_j \le 0 \\ c_k = \gamma_{ik} \end{bmatrix}, \quad k \in K \\
&\qquad \veebar_{i\in D_k} Y_{ik}, \quad k \in K \\
&\qquad \Omega(\mathbf{Y}) = True \\
&\qquad c_k, x_j \in \mathbb{R}_+, \quad j \in J_{ik}, i \in D_k, k \in K \\
&\qquad Y_{ik} \in \{False, True\}, \quad i \in D_k, k \in K.
\end{aligned}
\tag{Proc}
$$

In problem Proc, c_k is the cost associated to the equipment chosen in disjunction $k \in K$. The flow quantity x_j is defined for each possible stream $j \in J$, with an associated profit. The global mass balances are described for each node in the process $n \in N$ by the linear constraint $\sum_{j\in J} r_{jn} x_j \le 0$, where r_{jn} is the coefficient of the mass balance for flow j. Each disjunction $k \in K$ presents the choice between $i \in D_k$ equipment alternatives. When choosing each alternative ($Y_{ik} = True$) the corresponding input-output constraint in terms of the flows $j \in J_{ik}$ and parameters $d_{ijk}, t_{ijk}, s_{ijk}$ is active, and the cost associated to that disjunction c_k takes the value γ_{ik}. The topology of the superstructure and extra logical constraints are included in $\Omega(\mathbf{Y}) = True$.

Using Figure 8.4 as an example, there are two disjunctions $k \in \{A, B\}$, where for the first one the set in the disjuncts $D_A = \{1, 2\}$, such that either equipment 1 (Y_{1A}) or equipment 2 (Y_{2A}) is chosen. Similarly $D_B = \{6, 7\}$.

An interesting alternative is where the sets D_k yield a single element, and there is a Disjunction for every equipment piece. This yields the following formulation:

$$
\begin{aligned}
&\min_{\mathbf{c},\mathbf{x},\mathbf{Y}} \sum_{k\in K} c_k + \sum_{j\in J} p_j x_j \\
\text{s.t.}\quad & \sum_{j\in J} r_{jn} x_j \le 0, \quad n \in N \\
& \begin{bmatrix} Y_k \\ \sum_{j\in J_k} d_{jk}(e^{x_j/t_{jk}} - 1) - \sum_{j\in J_k} s_{jk} x_j \le 0 \\ c_k = \gamma_k \end{bmatrix} \vee \begin{bmatrix} \neg Y_k \\ x_j = 0, \quad j \in J_k \\ c_k = 0 \end{bmatrix}, \quad k \in K \qquad \text{(Procb)} \\
& \Omega(\mathbf{Y}) = True \\
& c_k, x_j \in \mathbb{R}_+, \quad j \in J_k, k \in K \\
& Y_k \in \{False, True\}, \quad k \in K.
\end{aligned}
$$

This case allows several pieces of equipment to be built within each alternative as long as the objective is maximized. The fact that it represents the disjunction of a convex set and a single point means that the HR formulation will yield the convex hull of the union of these sets without requiring an extended formulation [48, Corollary 1].

The exponential input-output constraint can be formulated in conic form as follows:

$$
\sum_{j\in J} d_j(e^{x_j/t_j} - 1) - \sum_{j\in J} s_j x_j \le 0 \iff \begin{cases} \sum_{j\in J} d_j u_j - \sum_{j\in J} s_j x_j \le 0, \\ (t_j u_j + 1, t_j, x_j) \in \mathcal{K}_{exp} \end{cases} \tag{8.32}
$$

We include 5 variants of the process problem with $|K| \in \{21, 31, 36, 48, 100\}$ possible units. The first four cases are taken from [233, 293, 296]. The last case was generated for this

chapter, given that commercial solvers can trivially solve the smaller cases. The instances are denoted `process|K|` or `process|K|b` when implementing problems Proc and Procb, respectively. For the new instance, the parameters are chosen from the uniform distributions $d_{ijk} \in U[1, 1.2], t_{ijk} \in U[1, 1.3], s_{ijk} \in U[0.8, 1.2], \gamma_{ik} \in U[2, 3]$

8.4.2.2 Simultaneous Retrofit and Synthesis problems

A generalization of the process network problem is the simultaneous retrofit and synthesis problem. In this problem, there is an existing process network that needs to be upgraded. To do so, one can consider either installing new equipment or improving the existing one. The potential of this process is to be maximized given a budget constraint. This problem was first proposed by Jackson and Grossmann [309] and its GDP implementation was done by Sawaya [233]. In the synthesis problem, the problem is equivalent to Proc with an extra index for the time periods when the problem is considered. The retrofit synthesis problem contains additional linear constraints and disjunctions to represent the conditions associated with retrofitting the existing process units. The complete formulation is available in [62].

The instances solved here are parametric to the number of synthesis processes $|S| \in \{5, 10, 15, 30, 40\}$, the number of retrofit units $|R| \in \{8\}$ and the number of time periods considered $|T| \in 1, 2, 3, 4$, leading to instances `Syn|S|M|T|` and `RSyn|R||S|M|T|`.

8.4.2.3 Logistic Regression

Logistic regression is a training technique for binary classification. In this training task, given a set of D-dimensional points $\mathbf{p}_i \in \mathbb{R}^D, i \in I$ we will assign a binary classifier $y \in \{0, 1\}$ to each point in the case that they lie above or below a hyperline given by $\theta^\top \mathbf{p}$. This line needs to be determined such that the logistic cost function is minimized. The logistic cost function $\log(1/(1 + e^{-\theta^\top \mathbf{p} + \theta_0}))$ can be interpreted as the probability of a point belonging to the class

given by $y = 1$. This problem can be modeled as a GDP by encoding the binary classifier y in a Boolean variable Y and writing the constraints within the disjunctions as follows:

$$
\begin{aligned}
&\min_{\theta,\mathbf{t}} \sum_{i\in I} t_i \\
&\text{s.t.} \begin{bmatrix} Y_i \\ t_i \geq \log\left(1+e^{-\theta^\top \mathbf{p}+\theta_0}\right) \\ \theta^\top \mathbf{p} \geq 0 \end{bmatrix} \vee \begin{bmatrix} \neg Y_i \\ t_i \geq \log\left(1+e^{\theta^\top \mathbf{p}+\theta_0}\right) \\ \theta^\top \mathbf{p} \leq 0 \end{bmatrix}, \quad i \in I \\
&\Omega(\mathbf{Y}) = True \\
&t_i \in \mathbb{R}_+, \quad i \in I \\
&\theta_0 \in \mathbb{R} \\
&\theta_j \in \mathbb{R}, \quad j \in [\![D]\!] \\
&Y_k \in \{False, True\}, \quad i \in I,
\end{aligned}
\tag{LogReg}
$$

where the logical constraints $\Omega(\mathbf{Y}) = True$ can enforce symmetry-breaking constraints to help in the solution process or other additional constraints related to the regression task.

The logistic regression constraint can be expressed as the following conic inequality:

$$
t \geq \log\left(1+e^{\theta^\top \mathbf{p}+\theta_0}\right) \iff \begin{cases} u+v \leq 1, \\ x = \theta^\top \mathbf{p} + \theta_0, \\ (v, 1, -t) \in \mathcal{K}_{exp}, \\ (v, 1, x-t) \in \mathcal{K}_{exp}, \end{cases} \tag{8.33}
$$

and equivalently for the complementary disjunction.

In the examples presented herein, we generate ten random instances for each one of the following settings. We set the value of the points' dimensions within $D \in \{2, 5, 10\}$, $|I| = 20$, and we choose to generate 2 clusters of normally distributed points being at a Mahalanobis

distance, i.e., a distance metric between points and distributions, such that the points are at most $\sigma \in \{1,2\}$ standard deviations away from the center of the distributions. This is computed via an inverse χ-squared distribution with D degrees of freedom computed at probabilities $\{0.68,0.95\}$ corresponding to Mahalanobis distances of $\sigma \in \{1,2\}$ in the one-dimensional case. This distance is then divided in $2\sqrt{D}$, such that we place the centers of the distributions at opposite corners of the D-dimensional hypercube. As mentioned in [302], a natural advantage of the mathematical programming approach to the training tasks in ML, compared to the heuristics, is that additional constraints can be enforced through the problem formulation. In this case, within $\Omega(\mathbf{Y}) = True$, we force the split between the data points to be within 45% and 55% and also force the farthest two points in the set from the origin to belong to opposite classes as a symmetry breaking constraint. Instances generated by this method are denominated `LogReg_D_|I|_`σ`_x`.

8.4.2.4 Random examples

Besides the applications-related instances listed above, we generate random instances whose disjunctive constraints can be represented using $\mathcal{K}_{exp}$. The form of these GDP is:

$$
\begin{aligned}
&\min_{\mathbf{x},\mathbf{Y},z} \mathbf{c}^\top \mathbf{x} \\
&\text{s.t.} \bigvee_{i\in D_k} \begin{bmatrix} Y_{ik} \\ a_{ik}^{'} \exp \sum_{j\in[\![n]\!]} a_{ijk}^{''''} x_j \le a_{ik}^{''} z + a_{ik}^{'''} \end{bmatrix}, \quad k \in K \\
&\quad \veebar_{i\in D_k} Y_{ik}, \quad k \in K \\
&\quad \mathbf{x}^l \le \mathbf{x} \le \mathbf{x}^u \\
&\quad z \le z^u \\
&\quad \mathbf{x} \in \mathbb{R}^n \\
&\quad z \in \mathbb{R} \\
&\quad Y_{ik} \in \{False, True\}, \quad k \in K, i \in D_k,
\end{aligned}
\tag{EXP-rand-GDP}
$$

where the upper and lower bounds of variables $\mathbf{x}$, $\mathbf{x}^l$ and $\mathbf{x}^u$, are set at 0 and 10, respectively. An upper bound for z is given by

$$
z^u = \max_{i\in D_k, k\in K} \left[\frac{a_{ik}^{'} \exp \sum_{j\in[\![n]\!]} a_{ijk}^{''''} x_j^l - a_{ik}^{'''}}{(a_{ik}^{''})^2} \right]. \tag{8.34}
$$

The exponential constraint can be written equivalently as a logarithmic constraint and in a conic form as follows:

$$
\begin{aligned}
& a_{ik}^{'} \exp \sum_{j\in[\![n]\!]} a_{ijk}^{''''} x_j \le a_{ik}^{''} z + a_{ik}^{'''} \\
& \iff \log(a_{ik}^{'}) + \sum_{j\in[\![n]\!]} a_{ijk}^{''''} x_j \le \log(a_{ik}^{''} z + a_{ik}^{'''}) \\
& \iff \begin{cases} a_{ik}^{'} v_{ik} \le a_{ik}^{''} z + a_{ik}^{'''} \\ \left(v_{ik}, 1, \sum_{j\in[\![n]\!]} a_{ijk}^{''''} x_j\right) \in \mathcal{K}_{exp}. \end{cases}
\end{aligned}
\tag{8.35}
$$

The generation of the random exponential GDPs use the same parameters as the random quadratic GDPs, i.e., $|K| \in \{5, 10\}, D_k \in \{5, 10\}$, and $n \in 5, 10$. Ten instances, denoted `exp_random_|K|_|`D_k`|_n_x`, are generated for each combination, besides a simple case with `exp_random_2_2_2_x` and the extra parameters are drawn from uniform distributions as $a'_{ik} \in U[0.01, 1], a''_{ik} \in U[0.01, 1], a'''_{ik} \in U[0.01, 1], a''''_{ijk} \in U[0.01, 1], c_j \in U[-1, -0.01], i \in D_k, k \in K, j \in [\![n]\!]$.

8.4.2.5 Results

We solve 208 GDP instances that are representable through the exponential cone. These instances are transformed through Big-M and HR. Since the exponential cone is not automatically identified through the constraints defining them, the explicit description of the cone was required, giving rise to two different versions of each reformulation. The Big-M results are summarized in Table 8.3 and the HR results are included in Table 8.4.

Depending on the family of instances, a given combination of solver and reformulation was the best in runtime. For the `LogReg*` and `RSyn*` instances, MOSEK-OA was the best solver when applied to the HR-Cone formulation. The other algorithm for MOSEK, MOSEK-IP, was the best performance solver for the `proc*` instances, with the outstanding solution of the `proc_100` problems in less than 5 seconds when most other approaches could not solve it within the 1-hour time limit. The closest non-conic approach was BARON applied to the original Big-M formulation. A $\approx$ 80x and 6x speedup was obtained with instances `proc_100` and `proc_100b`, respectively. BARON applied to the Big-M formulation was the best among all solvers for the `Syn*` instances. This approach was the fastest for the `exp_random*` instances, with a pretty similar performance achieved when applied to the Big-M-Cone formulation. This was not the case in general, where the conic formulation of the Big-M problem led to considerable performance degradation for BARON when solving

the `LogReg*` and `Syn*` instances. When considering the Big-M-Cone formulation, we see that both KNITRO and MOSEK-IP time out for most instances.

When considering the HR, using a conic formulation severely affected the performance of BARON and KNITRO. This was a sign of the challenges that gradient-based methods encounter when facing exponential constraints such as the ones appearing in the conic reformulation. As an example, in instance `RSyn0805M02`, the HR-Cone formulation led to KNITRO failing to evaluate the gradients at every B&B node, given function overloads by the evaluation of exponential functions. BARON could not find a solution to this problem either, while a solver that takes advantage of the exponential cone such as MOSEK solved the problem in 2 seconds.

As with the quadratic instances, the most efficient solver in terms of nodes explored to find the optimal solution is BARON, both in the Big-M and HR.

As with the quadratic instances, performance profiles are presented in Figures 8.5 and 8.6 for the exponential instances. In the time performance profile in Figure 8.5, we observe a clear dominance of both MOSEK algorithms applied to the HR-Cone formulation, particularly within the first seconds. Towards the end of the time limit, BARON applied to both the Big-M and HR formulations solves more instances to optimality. BARON applied to the HR-ε approximation can solve all the exponential problems within the time limit. Except for BARON, all solvers improve their performance when comparing the Big-M and HR formulations. Having mentioned that, both BARON and KNITRO have difficulties when solving the HR-Cone formulation, with the extreme case of BARON failing for all instances.

When observing the node performance profile for the exponential instances in Figure 8.6, the HR formulations require fewer nodes than the Big-M formulations, except for BARON. BARON proves that it generates strong relaxation nodes, requiring fewer to solve the problems, clearly dominating in this sense the other solvers. A similar observation was

Table 8.3: Results for Exponential GDPs using the Big-M reformulation and different solvers. The least time and fewest nodes results for each instance within a reformulation are italicized. The best results overall are bolded.

	Big-M				**Big-M-Cone**							
	BARON		KNITRO		BARON		KNITRO		MOSEK-OA		MOSEK-IP	
Instance	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes
LogReg_10_20_1_*	***.28±.05***	***1±0***	436.44±41.76	45384±2261	471.02±76.56	13699±3061	720.82±33.83	46478±2163	113.76±20.31	21941±1742	244.55±20.07	23759±1513
LogReg_10_20_2_*	***.42±.53***	***2±2***	441.82±43.23	45418±2507	453.59±47.54	13884±2365	719.01±44.70	45368±2353	111.68±18.63	22454±1443	238.62±17.22	24278±1820
LogReg_2_20_1_*	*14.79±6.55*	***71±26***	612.61±419.51	73711±45473	22.54±10.84	69±28	710.94±453.25	73709±45475	157.14±100.41	80143±48546	358.59±228.81	75566±44999
LogReg_2_20_2_*	*9.91±3.74*	***51±21***	453.38±270.08	53247±29832	37.34±19.98	180±228	690.44±421.71	53239±29840	120.22±79.25	59030±34633	272.45±158.24	57411±31825
LogReg_5_20_1_*	*26.61±8.56*	***187±81***	39.69±11.01	6891±1668	12.55±3.23	64±26	98.81±20.07	7111±1694	33.50±7.50	8834±1534	62.72±7.09	7864±1066
LogReg_5_20_2_*	24.90±8.71	***172±75***	36.92±10.21	6673±1893	12.47±3.71	68±38	98.16±23.75	6586±1842	34.56±12.30	9171±2629	57.55±12.03	7134±1746
RSyn0805	*0.83*	35	104.75	14427	0.92	33	94.49	12229	0.61	665	5.17	1645
RSyn0805M02	*8.73*	*59*	2278.77	87511	17.22	157	1995.42	75301	37.81	19339	178.07	17687
RSyn0805M03	*12.63*	*51*	3600+ (46%)	86418	15.29	57	3600+ (52%)	81914	108.12	40451	182.13	12213
RSyn0805M04	*14.08*	*37*	3600+ (64%)	59738	17.05	41	3600+ (66%)	51530	80.36	19417	630.2	27821
RSyn0810	*0.16*	***1***	299.27	34021	*0.16*	*1*	439.74	45863	0.42	307	15.52	3969
RSyn0810M02	*10.61*	*57*	3600+ (314%)	98341	42.75	419	3600+ (337%)	88431	263.01	103759	2075.97	177545
RSyn0810M03	*21.75*	*55*	3600+ (142%)	69420	139.25	698	3600+ (134%)	60807	2129.23	540975	3345.5	176379
RSyn0810M04	*20.18*	*39*	3600+ (120%)	39870	71	225	3600+ (121%)	33496	255.91	51883	3600+ (16%)	121298
RSyn0815	*0.78*	***15***	327.21	35507	1.31	17	430.84	42155	0.92	699	56.1	13117
RSyn0815M02	*8.64*	*49*	3600+ (∞)	99192	94.94	1418	3600+ (744%)	86741	77.64	26931	3600+ (10%)	304799
RSyn0815M03	*16.85*	*53*	3600+ (87%)	64196	32.47	93	3600+ (91%)	55528	254.47	44941	3600+ (16%)	199725
RSyn0815M04	*30.96*	*49*	3600+ (∞)	32893	724.02	1816	3600+ (∞)	30276	3600+ (5%)	574357	3600+ (45%)	135070
RSyn0820	*1.58*	*41*	3008.63	189045	3.42	73	*3217.97*	194571	5.96	4275	214.49	39649
RSyn0820M02	*14.27*	*67*	3600+ (∞)	79117	24.56	229	3600+ (∞)	64384	458.06	150715	3600+ (79%)	249620
RSyn0820M03	*27.41*	*83*	3600+ (609%)	50557	93.13	317	3600+ (613%)	42199	3601.3 (12%)	861705	3600+ (62%)	171127
RSyn0820M04	*42.33*	*75*	3600+ (∞)	30632	160.87	294	3600+ (∞)	27670	3600+ (29%)	471300	3600+ (121%)	114152
RSyn0830	*1.58*	*45*	3600+ (646%)	179714	4.9	113	3600+ (660%)	165011	2.01	1133	289.17	38547
RSyn0830M02	*8.6*	***33***	3600+ (∞)	66766	37.42	177	3600+ (∞)	54687	53.55	11553	3600+ (153%)	192187
RSyn0830M03	*27.19*	***61***	3600+ (∞)	33975	187.37	597	3600+ (1%)	30226	2610.51	262461	3600+ (132%)	115968
RSyn0830M04	*67.81*	*117*	3600+ (∞)	20713	466.32	1001	3600+ (∞)	19862	3600+ (6%)	292737	3600+ (143%)	75392
RSyn0840	*1.88*	*39*	3600+ (∞)	165386	4.3	75	3600+ (∞)	150704	0.87	367	1332.52	143695
RSyn0840M02	*12.23*	***45***	3600+ (∞)	53553	235.49	1209	3600+ (∞)	43142	259.71	38513	3600+ (190%)	122054
RSyn0840M03	*22.25*	*43*	3600+ (334%)	25769	210.43	738	3600+ (337%)	21384	762.7	65449	3600+ (55%)	97644
RSyn0840M04	*63.98*	*131*	3600+ (812%)	16402	3600+ (4.08%)	6134	3600+ (∞)	14389	3600+ (22%)	225035	3600+ (199%)	41952
Syn05	**0.04**	***1***	0.03	9	0.04	3	0.04	9	0.07	5	0.06	5
Syn05M02	***0.08***	***1***	0.31	55	0.13	*1*	0.33	47	0.13	5	0.12	7
Syn05M03	0.12	***1***	0.45	85	*0.11*	*1*	0.47	79	0.19	5	0.15	9
Syn05M04	*0.09*	***1***	0.9	105	0.14	*1*	0.89	75	0.24	7	0.25	11
Syn10	*0.04*	***1***	0.15	63	*0.04*	*1*	0.15	63	0.09	7	0.12	17
Syn10M02	*0.11*	***1***	6.39	1217	0.61	17	5.55	819	0.35	97	0.66	155
Syn10M03	*0.17*	***1***	18.91	2353	1.02	23	24.21	2497	0.47	125	1.62	281
Syn10M04	*0.24*	***1***	74.95	5727	2	35	96.06	5573	1.16	541	4.12	607
Syn15	*0.05*	***1***	0.32	101	0.22	9	0.35	101	0.19	17	0.32	79
Syn15M02	*0.16*	***1***	13	1563	0.85	11	14.93	1319	0.31	47	2.28	399
Syn15M03	*0.22*	***1***	46.28	3219	1.81	17	58.14	3225	1.27	405	10.29	1543
Syn15M04	*0.36*	***1***	294.81	13857	3.46	23	374.43	13659	1.66	419	38.91	4445
Syn20	*0.1*	***1***	2.05	727	0.34	13	2.61	727	0.23	89	1.75	443
Syn20M02	*0.27*	***1***	641.76	59013	1.15	11	1434.96	104925	0.47	143	19.64	3285
Syn20M03	*0.49*	***1***	3600+ (2%)	164828	5.67	43	3600+ (6%)	137094	3.32	963	361.29	44127
Syn20M04	*0.87*	***1***	3600+ (195%)	114242	12.45	85	3600+ (207%)	95068	20.36	6527	-	-
Syn30	*0.31*	***5***	155.74	33457	2.38	61	206.97	33831	0.27	19	10.57	2189
Syn30M02	*1.44*	*7*	3600+ (559%)	159527	10.35	129	3600+ (∞)	146231	1.5	299	567.27	59585
Syn30M03	*3.07*	*7*	3600+ (514%)	108728	20.39	163	3600+ (507%)	92989	6.03	841	-	-
Syn30M04	*5.67*	*5*	3600+ (585%)	68862	65.62	456	3600+ (577%)	45120	25.68	2803	-	-
Syn40	*0.34*	*5*	1412.67	154355	5.38	147	2078.01	176433	0.4	73	109.23	18233
Syn40M02	*1.63*	*5*	3600+ (248%)	128057	40.68	481	3600+ (267%)	109363	2.58	603	-	-
Syn40M03	*5.3*	*11*	3600+ (∞)	74752	244.77	1815	3600+ (∞)	58161	54.77	6411	-	-
Syn40M04	*14.24*	*39*	3600+ (∞)	-	493.84	2686	-	-	-	-	-	-
exp_random_10_10_10_*	***.66±.39***	***1±0***	29.62±20.08	4553±3144	1.15±.25	***1±0***	95.01±51.82	7648±4298	8.49±2.68	5924±2505	71.84±19.71	3103±713
exp_random_10_10_5_*	.69±.24	***1±0***	14.87±7.20	2792±1485	***.68±.24***	***1±0***	43.22±18.51	3155±1460	2.74±1.54	1684±1321	50.14±34.17	2062±1341
exp_random_10_5_10_*	.67±.18	***1±0***	5.33±1.80	1048±381	***.54±.16***	***1±0***	8.90±3.11	1020±424	1.21±.37	1084±371	8.69±4.54	721±444
exp_random_10_5_5_*	*.25±.10*	***1±0***	4.14±1.32	1146±400	***.32±.04***	***1±0***	8.38±2.75	1124±411	1.00±.30	691±322	5.04±2.74	423±188
exp_random_2_2_2_*	*.03±.02*	***1±1***	.03±.01	5±1	***.04±.02***	***1±0***	.03±.00	5±1	.05±.01	0±0	.07±.02	0±0
exp_random_5_10_10_*	.66±.20	***1±0***	3.27±1.10	594±201	***.65±.15***	***1±0***	5.46±1.37	651±187	1.01±.25	651±340	6.12±1.79	501±142
exp_random_5_10_5_*	*.23±.12*	***1±0***	1.62±.42	362±134	***.28±.09***	***1±0***	3.32±1.04	413±159	.62±.15	358±153	3.84±3.22	332±261
exp_random_5_5_10_*	.27±.02	***1±0***	.81±.24	196±80	***.23±.05***	***1±0***	1.27±.36	204±70	.40±.06	244±153	.78±.23	157±67
exp_random_5_5_5_*	.15±.03	***1±0***	.67±.32	171±140	***.16±.03***	***1±0***	1.17±.48	146±63	.29±.07	102±45	.47±.26	86±49
proc_100	3600+ (627.33%)	*976*	3165.79 (637%)	200000+	3600+ (484.9%)	229898	3600+ (741%)	161504	2682.55 (292%)	702218	3600+ (664%)	230134
proc_100b	3600+ (579.56)	*952*	2756.28 (711%)	200000+	3600+ (530.6%)	249969	3600+ (853%)	176762	2225.72 (254%)	632605	3600+ (486%)	230521
proc_21	1.27	*49*	4.6	921	1.21	55	5.57	863	0.91	491	2.57	447
proc_21b	*0.83*	*25*	43.95	11575	2.2	123	64.47	11833	1.19	1355	7.21	1629
proc_31	0.6	*9*	3.59	595	1.22	*9*	5.02	555	0.52	139	3.09	273
proc_31b	*1.3*	31	1206.42 (1%)	200000+	1.52	*19*	1837.92 (10%)	200000+	5.51	4439	56.14	10329
proc_36	1	*11*	5.4	779	1.45	9	8.06	719	0.87	373	4.02	549
proc_36b	4.5	*67*	1741.1 (42%)	200000+	3.32	*31*	2446.1 (37%)	200000+	10.96	14387	206.17	39771
proc_48	*1.3*	13	45.59	5999	1.85	*13*	74.83	5899	1.95	885	13.22	1331
proc_48b	*7.79*	*37*	2070.89 (81%)	200000+	8.24	*25*	2627.41 (78%)	200000+	48.14	43739	994.5	145623

Table 8.4: Results for Exponential GDPs using the Hull reformulation and different solvers. The least time and fewest nodes results for each instance within a reformulation are italicized. The best results overall are bolded.

	HR-ϵ						HR-Cone							
	BARON		KNITRO		BARON		KNITRO		MOSEK-QA		MOSEK-IP			
Instance	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes	Time [s] (Remaining gap %)	Nodes		
LogReg-10_20_1_*	.29±.07	**1±0**	451.32±41.09	45384±2261	3600+	29195±11436	-	-	.74±.26	96±46	1.01±.32	72±45		
LogReg-10_20_2_*	.32±.76	2±2	448.81±44.10 (.00%)	45418±2607	3600+	29965±7302	-	-	1.26±.61	142±103	.86±.39	69±42		
LogReg-2_20_1_*	20.31±8.78	71±26	622.69±467.89	73711±45473	3600+	35415±27159	-	-	*1.27±.33*	231±36	1.97±.41	162±40		
LogReg-2_20_2_*	13.00±5.07	51±21	462.20±288.83	53247±29832	3600+	34596±24189	-	-	*1.16±.35*	168±68	2.20±.48	178±45		
LogReg-5_20_1_*	32.11±11.15	***187±81***	40.33±11.30 (.00%)	6891±1668	3600+	33706±18346	-	-	*3.18±1.10*	929±256	9.88±2.15	879±190		
LogReg-5_20_2_*	29.64±9.88	***172±75***	38.80±10.72 (.00%)	6673±1893	3600+	33054±15143	-	-	*3.48±1.36*	944±244	9.71±2.69	879±245		
RSyn0805	0.51	3	1.62	117	2182.86 (1%)	1000000+	1.45	137	***0.12***	***0***	0.3	15		
RSyn0805M02	2.84	5	39.57	81	3600+ (∞)	368888	251.6 (∞)	100000	***0.55***	19	2.08	93		
RSyn0805M03	5.45	5	17.91	103	3600+ (1%)	441998	654.83 (∞)	100000	***0.72***	19	1.28	23		
RSyn0805M04	8.28	5	5.34	41	3600+ (1%)	275166	845.01 (∞)	100000	***0.9***	7	1.53	11		
RSyn0810	0.14	*1*	1.76	111	3400.57 (∞)	1000000+	241.11 (5%)	199766	***0.36***	7	0.44	9		
RSyn0810M02	14.93	81	31.83	175	3600+ (∞)	266400	360.38 (∞)	100000	***0.57***	7	1.92	87		
RSyn0810M03	40.54	121	*42.97*	489	3600+ (∞)	161105	598.03 (∞)	100000	***1.67***	89	8.2	299		
RSyn0810M04	56.86	101	22.93	71	3600+ (∞)	57581	912.28 (∞)	100000	***1.19***	27	10.18	249		
RSyn0815	6.08	89	4.31	139	3600 (∞)	67106	158.09 (∞)	199986	***0.23***	9	0.46	23		
RSyn0815M02	16.33	55	13.14	67	3600+ (∞)	231668	343.19 (∞)	100000	***0.66***	29	4.86	217		
RSyn0815M03	69.43	141	46.48	469	3600+ (∞)	112176	756.75 (∞)	100000	***1.39***	***99***	12.41	359		
RSyn0815M04	49.77	37	39.35	171	3600+ (∞)	56238	1117.5 (∞)	100000	***1.24***	5	2.52	23		
RSyn0820	12.21	159	2.44	127	3514.55 (8%)	1000000+	167.94 (∞)	199989	***0.31***	***17***	0.64	11		
RSyn0820M02	27.95	77	98.68	299	3600+ (∞)	76882	368.17 (∞)	100000	***0.72***	57	4.16	151		
RSyn0820M03	81.48	*117*	249.24	2171	3600+ (∞)	76882	746.46 (∞)	100000	***1.88***	119	16.58	455		
RSyn0820M04	157.7	131	72.75	395	3600+ (∞)	30904	1363.63 (∞)	100000	***1.67***	***45***	11.49	155		
RSyn0830	9.57	105	4.27	162	3600+ (∞)	504387	203.8 (∞)	200000+	***0.38***	49	0.85	***45***		
RSyn0830M02	117.24	311	21.39	213	3600+ (∞)	125094	3600+ (∞)	2002	***0.96***	*57*	5.64	183		
RSyn0830M03	249.05	235	85.91	445	3600+ (∞)	62648	3600+ (∞)	1498	***2.22***	179	10.38	203		
RSyn0830M04	3600+ (1%)	1027	259.27	1265	3600+ (∞)	39280	2298.94 (∞)	100000	***2.95***	*119*	55.61	769		
RSyn0840	4.92	29	1.41	27	3600+ (∞)	420832	234.28 (∞)	200000+	***0.33***	***17***	0.67	*15*		
RSyn0840M02	221	223	12.52	103	3600+ (∞)	93859	1279.09 (∞)	100000	***0.98***	37	4.02	95		
RSyn0840M03	2034.36	454	72.92	323	3600+ (∞)	40607	1398.34 (∞)	100000	*1.65*	*41*	10.26	149		
RSyn0840M04	3600+ (2%)	413	657.44	2339	3600+ (∞)	27908	2710.11 (∞)	100000	3.95	153	73.35	823		
Syn05	0.06	*1*	0.03	3	819.94 (∞)	1000000+	0.1	3	0.04	0	0.05	5		
Syn05M02	0.12	*1*	0.05	3	1164.47 (87%)	1000000+	0.04	3	0.12	3	0.13	3		
Syn05M03	0.28	*1*	0.06	3	3047.17 (∞)	1000000+	0.05	3	0.14	3	0.1	3		
Syn05M04	0.15	*1*	0.08	3	3600+ (∞)	785490	0.06	3	0.19	3	0.12	3		
Syn10	0.05	*1*	0.03	3	1321.8 (∞)	1000000+	0.04	3	*0.07*	*0*	*0.01*	*0*		
Syn10M02	0.44	*1*	0.14	9	2103.28 (6%)	1000000+	0.08	5	0.22	3	0.15	3		
Syn10M03	0.71	**5**	0.29	7	3357.55 (1%)	1000000+	0.15	7	0.33	3	*0.16*	3		
Syn10M04	2.1	15	0.27	7	3600+ (2%)	635620	0.22	7	0.35	3	0.26	3		
Syn15	0.2	5	0.05	3	1900.43 (∞)	1000000+	*4.85* (2%)	16387	***0.08***	3	0.11	3		
Syn15M02	0.26	3	0.21	5	3600+ (5%)	919619	0.09	3	0.15	3	0.16	3		
Syn15M03	1.06	3	0.21	5	3600+ (7%)	540753	0.25	5	0.22	3	0.24	3		
Syn15M04	1.9	5	0.35	6	3600+ (1%)	296378	0.31	3	0.2	0	0.39	3		
Syn20	0.58	11	0.06	5	2537.75 (7%)	1000000+	0.08	5	0.18	3	0.11	3		
Syn20M02	1.42	5	0.35	13	3600+ (3%)	975067	258.36 (∞)	199997	0.23	3	0.2	5		
Syn20M03	7.26	21	1.02	27	3600+ (∞)	300067	932.34 (∞)	199997	0.31	3	*0.44*	13		
Syn20M04	16.82	21	1.75	27	3600+ (∞)	208090	839.52 (∞)	199978	0.44	5	0.68	13		
Syn30	2.1	15	0.13 (0%)	7	2735.21 (∞)	1000000+	0.14 (0%)	7	0.18	7	0.17	7		
Syn30M02	6.11	19	0.42	11	3600+ (∞)	292942	0.68	11	*0.36*	11	0.38	9		
Syn30M03	65.88	49	1.56	23	3600+ (∞)	148036	605.33 (∞)	199998	*0.69*	21	0.82	21		
Syn30M04	85.14	47	3.56	37	3600+ (∞)	99419	4.43	37	*1.07*	37	1.83	27		
Syn40M02	3.27	15	0.28	13	3600+ (∞)	827854	113.53 (∞)	200000+	0.23	*11*	0.22	*11*		
Syn40M03	32.11	71	1.39	23	3600+ (∞)	251756	512.42 (∞)	199998	*0.58*	31	0.69	15		
Syn40M04	276.42	155	8.4	99	3600+ (∞)	115088	1479 (0%)	199997	*1.17*	57	1.79	39		
exp-random1_10_10_*	1108.16	317	64.61	210	3600+ (∞)	62600	1773.96 (∞)	200000+	*1.97*	127	6.18	115		
exp-random1_10_10_-*	*15.43±8.40*	1±0	46.25±36.45	398±357	3600+	75022±11349	3600+ (∞)	200000+±0	27.98±12.71	264±183	22.48±10.59	319±180		
exp-random1_10_5_*	12.92±4.21	1±0	15.96±5.60	133±128	3600+	23103±20532	507.71±4.95	200000+±0	*8.39±6.46*	457±750	7.07±3.20	155±89		
exp-random1_10_5_10_*	7.32±4.84	1±0	14.33±6.53	154±99	3600+	22952±16504	367.52±3.68	200000+±0	*5.37±1.82*	107±70	3.80±1.49	105±57		
exp-random1_10_5_-*	2.30±1.34	1±0	7.12±2.66	155±102	3600+	45152±33605	235.56±.85	200000+±0	*1.89±.60*	73±51	3.16±1.32	139±76		
exp-random2_2_2_*	*.03±.01*	1±0	.03±.00	1±0	1053.56±124.36	1000000+±0	-	-	*.04±.02*	*0±0*	*.04±.02*	1±1		
exp-random5_10_10_*	10.99±5.87	3±5	7.17±3.33	35±23	3600+	32351±42747	-	-	5.60±2.24	44±40	2.54±.99	44±28		
exp-random5_10_5_*	2.11±1.51	1±0	3.22±1.47	18±11	3600+	56131±57215	-	-	*1.65±1.01*	61±134	*1.35±.50*	44±29		
exp-random5_5_10_*	1.97±2.18	1±0	2.46±1.12	30±24	3600+	66430±56737	-	-	*1.72±.63*	23±15	*.94±.24*	33±22		
exp-random5_5_5_*	.72±.53	1±0	1.10±.59	19±12	3511.73±114.18	972676±47392	-	-	.61±.19	16±10	*.59±.13*	28±16		
proc100	129.79	287	1.25	21	3600+ (484.9%)	259774	474.53 (∞)	200000+	16.12	1831	***1.47***	***9***		
proc100b	31.33	7	5.46	87	3600+ (530.6%)	204643	472 (∞)	200000+	32.23	5199	***4.97***	***37***		
proc21	2.91	6	1.15	79	1000000+	1000000+	-	-	*0.36*	55	***0.59***	***47***		
proc21b	2.16	5	2.33	225	3600+ (∞)	89.93 (∞)	3600+	200000+	*0.24*	97	***0.47***	***91***		
proc31	1.32	1	0.18 (0%)	9	1832.16 (∞)	1000000+	150.63 (∞)	200000+	0.28	3	***0.13***	3		
proc31b	1.55	1	0.31 (0%)	17	1424.67 (∞)	1000000+	142.84 (∞)	200000+	0.62	3	***0.38***	3		
proc36	3.61	3	0.18	7	2123.32 (∞)	1000000+	166.94 (∞)	200000+	0.46	5	***0.29***	5		
proc36b	4.36	1	0.6	27	1940.37 (∞)	1000000+	161.01 (∞)	200000+	0.67	19	***0.22***	3		
proc48	3.37	1	0.28	11	2618.29 (∞)	1000000+	241.47 (∞)	200000+	0.99	7	***0.35***	5		
proc48b	5.08	3	1.23	57	2444.62 (∞)	864362	223.77 (∞)	200000+	1.41	27	***0.8***	***29***		

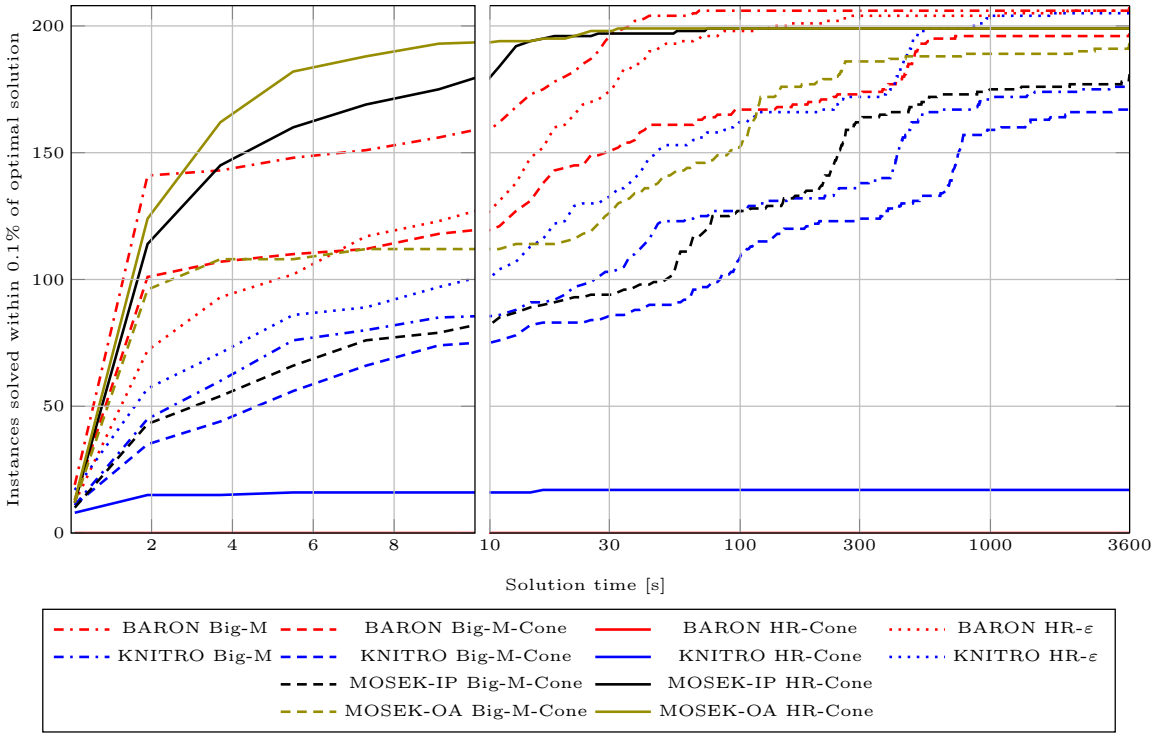


Figure 8.5: Time performance profile for exponential instances using the different GDP reformulations and commercial solvers.

made regarding the quadratic instances.

8.4.3 Controlling the Branch & Bound search

The implementations of modern solvers include an arsenal of heuristic methods to tackle more efficiently the challenging optimization problems at hand. Although this leads to performance improvements, it shades the effect of better formulations when solving the optimization problems. To that end, we consider using the Simple Branch& Bound (SBB) implementation in GAMS and solve the subproblems using both KNITRO and MOSEK. These subproblems are continuous optimization problems, while SBB manages the discrete variables' exploration. We present below two performance profiles in Figures 8.7 and 8.8

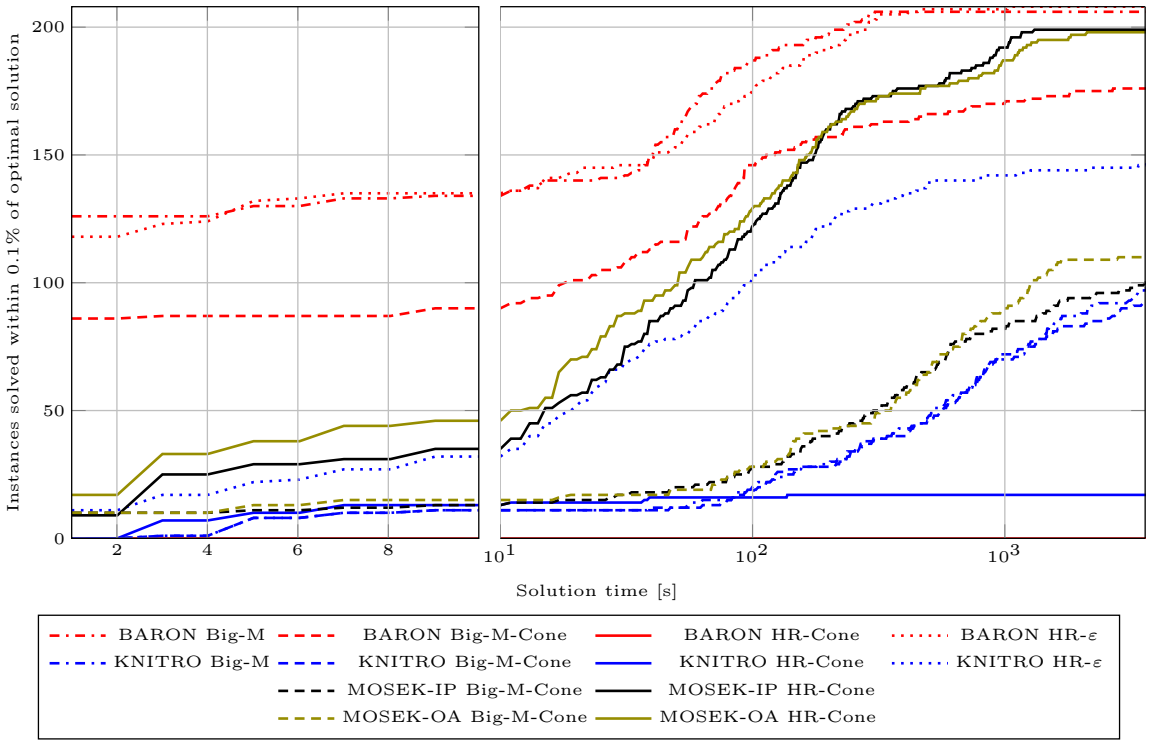


Figure 8.6: Node performance profile for exponential instances using the different GDP reformulations and commercial solvers.

for all the problems solved in this chapter, mainly including results of SBB-KNITRO and SBB-MOSEK.

In Figure 8.7 we observe the performance profiles of the SBB implementation against the number of continuous convex subproblems solved. The first observation is that the HR tight formulation allows a more efficient exploration of the subproblems solves than the Big-M formulation. The conic formulation of HR affects the performance of KNITRO when addressing the subproblems, leading to poor performance in this case. Moreover, given the same branching rules, the Big-M and HR formulations require approximately the same number of subproblems solved using the original or the extended formulations arising from the conic description of the problems. This is an expected result given that the extended formulation does not require additional binary variables.

Although the number of solved subproblems is similar, the time required to solve them varies depending on the chosen solver, as observed in Figure 8.8. In this figure, we include the time performance profiles for the SBB alternatives. For reference, we include the best commercial alternative to each reformulation. This corresponds to BARON for the Big-M and HR-ε and MOSEK-IP for the HR-Cone formulations. The solver that solved the most instances was BARON applied to the Big-M formulation, solving 393 out of the 425 problems, followed by MOSEK-IP applied to the HR-Cone formulation solving 390 problems. In general, MOSEK is more efficient at solving the convex subproblems compared to KNITRO. The difference is more exacerbated in the HR formulation. An interesting observation is that the gap in time performance between SBB and the best alternative is smaller for HR-ε than for HR-Cone. This indicates that the efficient exploitation of the conic constraints, in this case from MOSEK, can yield considerable performance advantages together with a tight reformulation of disjunctive constraints.

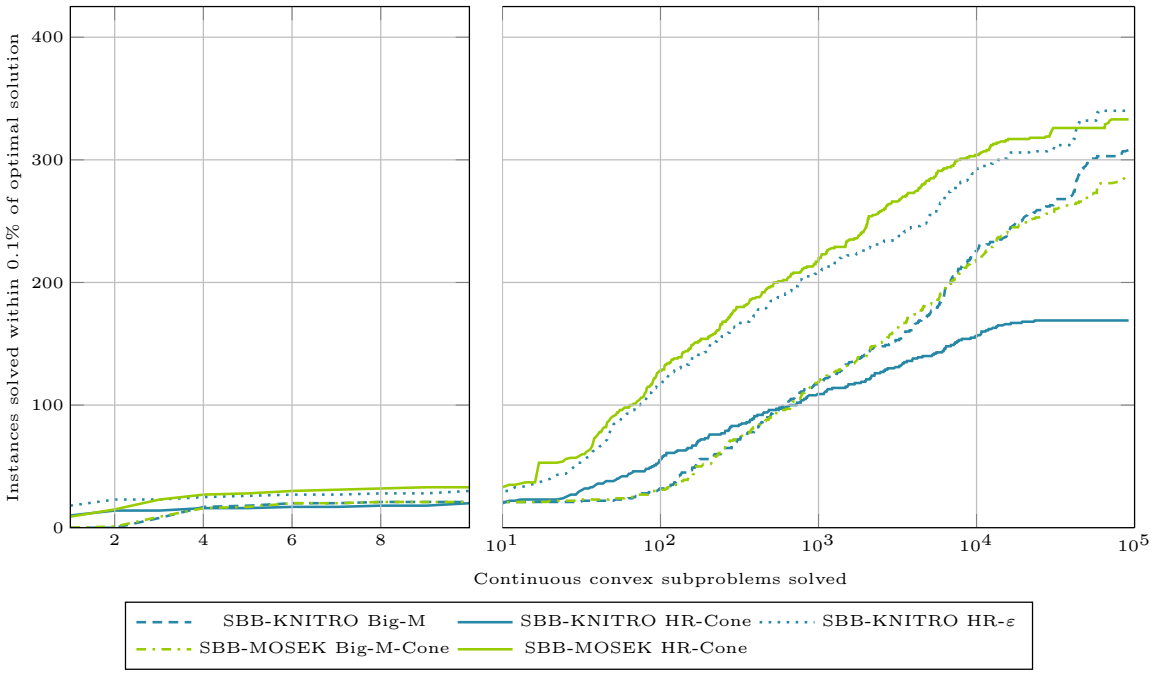


Figure 8.7: Solved subproblems performance profile for all instances using the different GDP reformulations and solvers through SBB.

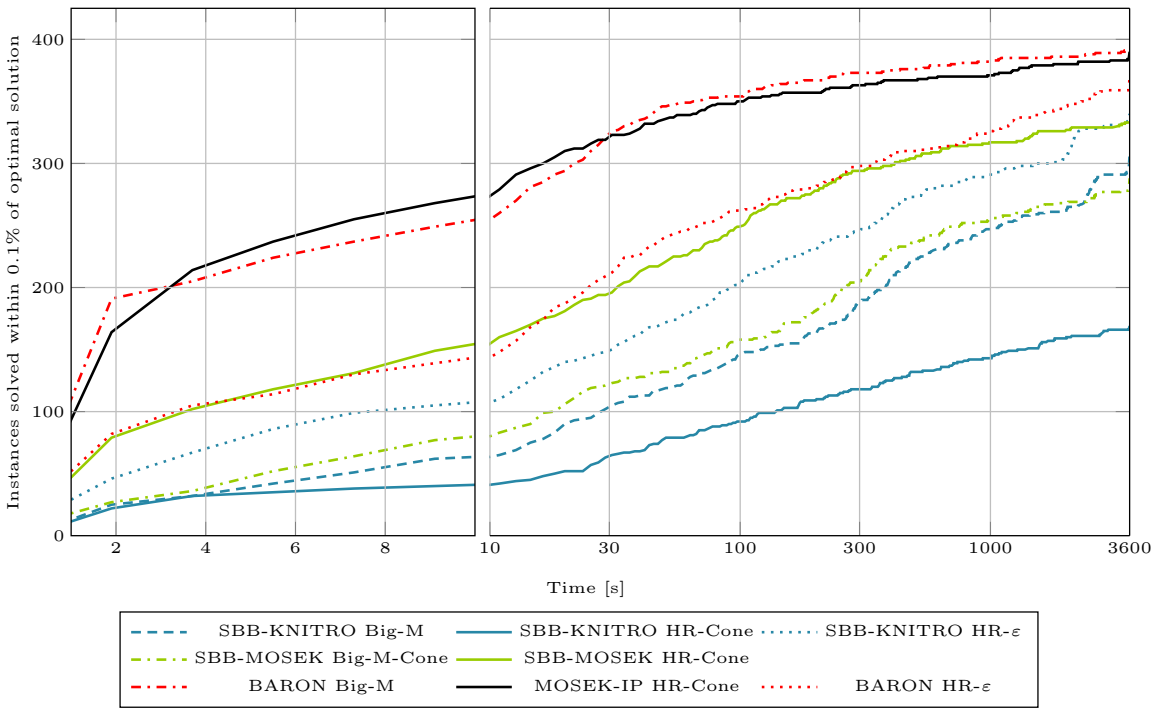


Figure 8.8: Time performance profile for all instances using the different GDP reformulations and solvers through SBB. We include the best performing commercial solver results for each reformulation.

8.5 Conclusions, discussion and future work

This work presents the formulation of convex Generalized Disjunctive Programming (GDP) problems using conic sets. The convex GDP problem can be solved through a reformulation into convex Mixed-Integer Nonlinear Programming (MINLP) problems. Two of those reformulations are covered in this chapter, the Big-M and Hull Reformulations. The Hull reformulation of a convex GDP problem requires implementing a perspective function, whose algebraic form is challenging for gradient-based nonlinear optimization solvers. We present the Big-M and Hull reformulations into Mixed-Integer Conic Programming (MICP) problems through the conic formulation of the problem. The MICP problems can be efficiently tackled using specialized conic solvers, which take advantage of the properties of the conic programs. We provide a guide to reformulate common convex constraints through conic programming. If those constraints appear inside disjunctions, we also provide a conic representation of its perspective, allowing the exact representation of the Hull reformulation.

These reformulations were tested using a large set of convex GDP problems stemming from Process Systems Engineering, Machine Learning, and randomly generated instances. These instances were classified as quadratic and exponential and solved through different reformulation alternatives and solvers. Our results show how the conic reformulation gives a systematic and natural extended formulation of the convex MINLP problems stemming from GDP. These can be exploited by solvers, allowing a more efficient solution to these problems. Among the tested approaches, we identified that BARON solving the Big-M formulation and MOSEK solving the HR-Conic formulation, either with IP or OA, were the best solvers to tackle these convex GDP reformulated problems. In general, we show how the conic representation of convex constraints within disjunctions can result in an exact and more efficiently solvable mixed-integer representation of a convex GDP.

The results in this paper also point to specific existing improvement opportunities. In the first place, the automatic reformulation of the convex constraints into cones is a task worth pursuing. Previous success in the quadratic case allows commercial solvers such as CPLEX or Gurobi to automatically detect conic structures and address those more efficiently. An extension of these routines to exponential cones is therefore of interest. Modeling extensions that allow for disjunctive programming are the natural place to include these automatic reformulations. Approaches have been made at the modeling language level, e.g., in GAMS [282] Pyomo [59] ⋆, and Julia ⋆. These could also be made at the solver level, with indicator constraints such as in CPLEX [52] and MOSEK [35]. These techniques have also shown potential for the global optimization of non-convex GDP or MINLP [310], motivating further research into it.

Interesting future directions are the exploration of conic formulations in more advanced reformulations of GDPs, such as intermediate Big-M / Hull formulations [306] and basic steps reformulations [293]. Moreover, conic programming tools can be used in more advanced solution methods of GDP than the recasting of the problem into MINLP. Examples of those methods are Lagrangean decomposition based on the disjunctive structure of the problem [305] or logic-based algorithms [59]. The use of conic programming has already shown the potential speedup for mixed-integer programming solutions [44], and expanding those findings to GDP is of great interest.

⋆https://pyomo.readthedocs.io/en/latest/modeling_extensions/gdp/

⋆https://github.com/rdeits/ConditionalJuMP.jl

Chapter 9

Characterization of QUBO Reformulations for the Maximum k-colorable Subgraph Problem*

9.1 Introduction

Quantum computing (QC) harnesses the properties of physical systems described by quantum mechanics (e.g., subatomic particles) to perform computations in a fundamentally different way than classical computing [312]. It is widely established that QC can, in the future, revolutionize the way we perform and think about computation and be the backbone of thrilling new technologies and products [312–314].

In particular, QC has the potential to radically transform our capability to solve difficult optimization problems for which no traditional numerical or theoretical efficient solution algorithms are known to exist [315]. This is particularly the case for *combinatorial optimization* (COPT) problems; that is, optimization problems that are formulated with the use of discrete (e.g., binary) decision variables [53]. A large number of COPT problems are known to be NP-Hard [see, e.g., 316]; that is, there is no known polynomial-time algorithm that can be used to solve them. A very representative problem in this class of COPT NP-Hard problems is the *Ising model* [see, e.g., 317–319]. Since its inception, the Ising model has been used to

*Preprint available as: Rodolfo Quintero, David E. Bernal, Tamás Terlaky, and Luis F Zuluaga. "Characterization of QUBO reformulations for the maximum k-colorable subgraph problem". *arXiv preprint arXiv:2101.09462* (2021).

address problems arising in different physical systems (e.g., magnetism, lattice gas, spin glasses), as well as in neuroscience and socio-economics.

The Ising model belongs to the class of *quadratically unconstrained binary optimization* (QUBO) problems [see,e.g., 320]. Moreover, both quantum annealing devices [see, e.g., 79, 321, 322], and algorithms (such as the quantum approximate optimization algorithm (QAOA)) for gate-based quantum computers [see, e.g., 323, 324] are able to address the solution of QUBO problems. This allows the use of quantum technology to solve problems such as the Ising model and the max-cut problem, which has a natural QUBO reformulation [see, e.g., 85, 325]. Moreover, quantum technology can be used to solve a broader class of constrained COPT problems that do not have a natural QUBO reformulation. This is due to the fact that penalization methods can be used to embed the COPT problem's constraints in its objective to obtain a QUBO reformulation of the problem.

For some COPT feasibility problems (i.e., without an objective) that can be formulated using linear equality constraints, the desired QUBO reformulation can be obtained using any positive penalty parameter (to penalize the constraints' violations). For example, consider the QUBO reformulations of the number partitioning problem [79, 326], the graph isomorphism problem [327], the exact cover problem [79], and some planning problems [328], to name a few. However, when the COPT problem formulation requires (or uses) nonlinear constraints and/or an objective function, the desired QUBO reformulation is only guaranteed to be obtained for values of the penalty parameter(s) that are larger than a known, and potentially large, lower bound. For example, consider the QUBO reformulations for the maximum clique problem [79], the traveling salesman problem [79, 326], and the minimax matching problem [79]. Worst, in some cases, the desired QUBO reformulation is only guaranteed to be obtained for an unknown large enough value of the penalty parameter(s). For example, consider the QUBO reformulations of the job shop scheduling problem [329], the

de-conflicting optimal trajectories problem [330], the traveling salesman problem with time windows [331], and some of the problems discussed in [332]. Additionally, when the COPT problem formulation requires (or uses) linear inequality constraints, a potentially large number of auxiliary (i.e., slack) binary variables need to be introduced to obtain the desired QUBO reformulation. For example, consider the maximum clique QUBO reformulation provided in [79], and the COPT problems considered in [333].

The fact that large (or unknowingly large) penalty parameters, and additional binary variables might be needed to obtain the desired QUBO reformulation can hinder the ability of quantum computers to more efficiently solve COPT problems [see, e.g., 333–335]. As the results in [336] highlight, this efficiency is key towards the goal of using *noisy intermediate-scale quantum* (NISQ) devices to solve COPT problems more efficiently than with classical computers. Not surprisingly, recent articles look beyond obtaining QUBO reformulations of COPT problems such as the graph isomorphism problem as well as tree and cycle elimination problems, to look for *improved* QUBO reformulations of these problems for NISQ devices [see, e.g., 327, 335, 337–339]. That is, QUBO reformulations that are tailored to be more efficiently used in NISQ devices.

Along these lines, we consider an important COPT problem; namely, the *maximum k-colorable subgraph* (MkCS) problem [see, e.g., 340], in which the aim is to find an induced k-colorable subgraph with maximum cardinality in a given graph. This problem arises in channel assignment in spectrum sharing networks (e.g., Wi-Fi or cellular) [341, 342], VLSI design [343], human genetic research [343, 344], telecommunications [345], and cybersecurity [346].

We derive two QUBO reformulations of the MkCS problem. The first one is obtained from the standard formulation of the MkCS problem in which all the constraints are linear, except for the binary variable constraints. This QUBO reformulation is an improved version

of the QUBO reformulation that would be obtained by using the QUBO reformulation approach of Lasserre [347] for this "linear" formulation of the M*k*CS. The reason for this is that we characterize the minimum penalization coefficients that can be used to guarantee that the desired QUBO problem, obtained by penalizing the problem's linear constraints violations, is indeed equivalent to the original problem. Furthermore, we characterize the equivalence of the QUBO reformulation not only in terms of the objective value, but also in terms of the optimal solution obtained from this QUBO reformulation. In particular, we find that when the minimal values of the penalization coefficients are used, the QUBO reformulation is equivalent to the M*k*CS in terms of the problems' objectives, but not in terms of the problems' optimal solutions. However, we show that in this case, the QUBO reformulation's optimal solution can be used, in a simple way, to obtain the M*k*CS problem's optimal solution. In what follows, we will refer to this QUBO reformulation of the M*k*CS problem as the *linear-based* QUBO reformulation.

The second QUBO reformulation of the M*k*CS problem is obtained from a formulation of the M*k*CS problem in which all the linear constraints are first formulated as nonlinear equality constraints. Analogous to the results obtained for the linear-based QUBO reformulation of the M*k*CS problem, we derive a *nonlinear-based* QUBO reformulation of the M*k*CS problem. Then, we characterize the minimum penalizations coefficients that can be used to guarantee that the desired nonlinear-based QUBO problem, obtained by penalizing the problem's linear constraints violations, is indeed equivalent to the original problem. Furthermore, we characterize the equivalence of the nonlinear-based QUBO reformulation not only in terms of the objective value, but also in terms of the optimal solution obtained from this nonlinear-based QUBO reformulation. In particular, we find that when the minimal values of the penalization coefficients are used, the nonlinear-based QUBO reformulation is equivalent to the M*k*CS in terms of the problems' objectives, but not in terms of the

problems' optimal solutions. However, we show that in this case, the nonlinear-based QUBO reformulation's optimal solution can be used, in a simple way, to obtain the MkCS problem's optimal solution. This latter result extends the work done in characterizations of QUBO reformulations of the *stable set problem* [348–350], which is equivalent to the MkCS problem when $k = 1$. The nonlinear-based QUBO reformulation of the MkCS problem is a substantial improvement over the linear-based QUBO reformulation of the MkCS problem, in significant part, because the former QUBO does not need the addition of any auxiliary (i.e., slack) binary variables beyond the ones that define the original problem's formulation.

To illustrate the benefits of obtaining and characterizing these QUBO reformulations, we benchmark different QUBO reformulations of the MkCS problem using a quantum annealing device, and in particular, we look at how embedding requirements and theoretical and numerical convergence rates change depending on the QUBO reformulation being used, as well as the parameters with which is used.

The rest of the chapter is organized as follows. In Section 9.2, we present some relevant discussion to motivate our work, as well as results about QUBO reformulations for COPT problems. In Section 9.3, we formally present the MkCS problem and two associated QUBO reformulations. The first one, in Section 9.3.1, is based on a "linear" (modulo the binary variable constraints) formulation of the MkCS problem. The second one, in Section 9.3.2, is based on a "nonlinear" (beyond the binary variable constraints) formulation of the MkCS problem. In Section 9.4, we benchmark these two QUBO reformulation by performing numerical tests on D-Wave's quantum annealing devices. We also illustrate the numerical power gained by using the latest D-Wave's quantum annealing devices. In Section 9.5, we finish with some concluding remarks.

9.2 Preliminaries

Formally, given a set of n binary decision variables $\mathbf{x} \in \{0,1\}^n$ (or $\mathbf{x} \in \{-1,1\}^n$) when appropriate), a vector $\mathbf{f} \in \mathbb{R}^n$, and a matrix $\mathbf{Q} \in \S^n$, where $\S^n$ is the set of symmetric matrices in $\mathbb{R}^{n\times n}$, a *quadratically unconstrained binary optimization* (QUBO) problem is the problem of finding [see, e.g., 320, 350]:

$$\begin{aligned} z^* = \quad & \min \quad \mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{f}^\top \mathbf{x} \\ & \text{s.t.} \quad \mathbf{x} \in \{0,1\}^n. \end{aligned} \tag{QUBO}$$

It is well-known that the Ising model belongs to the class of QUBO problems (using $\{-1,1\}$ binary variables) [see, e.g., 79, 335]. Moreover, other distinguished NP-Hard COPT problems can be naturally formulated, or easily reformulated as a QUBO problem. Foremost among this type of problems is the max-cut problem [see, e.g., 351], which arises in multiple important applications in science and engineering [see, e.g., 352, Sec. 6]. Given an undirected Graph $\mathcal{G}(V,E)$, the aim in the max-cut problem is to find a subset of nodes (or cut) $S \subseteq V$, such that the cardinality of the set of edges in E between the nodes in S and $S^c := V \setminus S$ is maximized. The max-cut problem can be naturally formulated (disregarding objective constants) as a QUBO problem (using $\{-1,1\}$ binary variables) by letting $\mathbf{Q} = \mathbf{A}$, $\mathbf{f} = 0$, where $\mathbf{A} \in \mathbb{R}^{V\times V}$ is the node-to-node adjacency matrix of $\mathcal{G}(V,E)$, or by setting $\mathbf{Q} = -\operatorname{diag}(\mathbf{A}\mathbf{e}) + 2\mathbf{A}$ and $\mathbf{f} = 0$ (using $\{0,1\}$ binary variables).

Thanks to the QUBO reformulation of the max-cut problem, the ability of quantum computers to solve the max-cut problem has been widely studied in the literature. For example, consider the use QAOA algorithms in [323, 324, 353], and of quantum annealing devices in [325, 354] to solve instances of the max-cut problem. Furthermore, QUBO reformulations can be obtained for a broader class of COPT problems that do not have a natural QUBO reformulation. This is done by using penalization methods to embed the COPT problem's constraints in its objective [see, e.g., 79, 326–330, 332, 333, 335, to

name just a few]. This approach clearly broadens the class of COPT problems that can be addressed with NISQ devices. However, the efficacy of NISQ devices to solve this broader class of COPT problems can be highly affected by the way in which the corresponding QUBO reformulation is obtained. This is because the performance of NISQ devices is highly affected by the number of qubits and the coefficients that are required to encode a QUBO [see, e.g., 327, 335, 337].

To illustrate this fact, consider the problem of obtaining a QUBO reformulation for the *maximum clique* problem. Given an undirected Graph $\mathcal{G}(V,E)$, the aim in the maximum clique problem is to find the set of nodes $S \subseteq V$ with the highest cardinality such that the graph induced by S is a clique; that is, a complete subgraph [see, e.g., 355]. The cardinality of the largest induced clique of $\mathcal{G}$ is referred to as the clique number $\chi(\mathcal{G})$. Lucas [79, Sec. 2.3] obtains a QUBO reformulation for the maximum clique problem by first noticing that $\mathcal{G}(V,E)$ contains a clique of size $K \in \{2,\dots,|V|\}$ (i.e., w.l.o.g. assume $|E| \geq 1$) if and only if there is $\mathbf{x} \in \{0,1\}^{|V|}$ such that $\sum_{i=1}^{|V|} x_i = K$, and $\sum_{(i,j)\in E} x_i x_j = \frac{1}{2}K(K-1)$. Thus, the maximum clique problem can be formulated as $\chi(\mathcal{G}) = \max\{K \in \{2,\dots,|V|\} : \sum_{i=1}^{|V|} x_i = K, \sum_{(i,j)\in E} x_i x_j = \frac{1}{2}K(K-1), \mathbf{x} \in \{0,1\}^{|V|}\}$. Furthermore, Lucas [79, Sec. 2.3] shows that this latter problem can be reformulated as the following QUBO.

$$\begin{aligned} \chi(\mathcal{G}) = \quad \min \quad & -\sum_{i=1}^{|V|} x_i + (\Delta+2)\Big(1-\sum_{k=2}^{\Delta} y_k\Big)^2 + (\Delta+2)\Big(\sum_{k=2}^{\Delta} k y_k - \sum_{i=1}^{|V|} x_i\Big)^2 + \\ & \tfrac{1}{2}\Big(\sum_{k=2}^{\Delta} k y_k\Big)\Big(-1+\sum_{k=2}^{\Delta} k y_k\Big) - \sum_{(i,j)\in E} x_i x_j \\ \text{s.t.} \quad & \mathbf{x} \in \{0,1\}^{|V|}, y_k \in \{0,1\}, k = 2,\dots,\Delta, \end{aligned} \tag{9.1}$$

where Δ is the degree of $\mathcal{G}(V,E)$, and the auxiliary variable $y_k = 1$ if $\chi(\mathcal{G}) = k$ and $y_k = 0$ otherwise for $k = 2,\dots,\Delta$. Note that the QUBO problem (9.1) uses $|V|+\Delta$ logical qubits and *coefficients* [333, Section 1.2] that belong to the range $[-2\Delta(\Delta+2), 2\Delta^3 + 3\Delta(\Delta-1)+4]$ (after disregarding constant terms and appropriately replacing $x_i \to x_i^2$, $i = 1,\dots,|V|$, $y_k \to y_k^2$, $k = 2,\dots,\Delta$ in the objective of (9.1) to make it a homogenous quadratic). The performance

of NISQ devices on solving QUBO problems is negatively affected by the use of a larger number of logical qubits and larger coefficients [see, e.g., 327, 332, 333, 335, 337]. In this context, it is natural to ask if there are *improved* [see, e.g., 327, 335, 337–339] QUBO reformulation for the maximum clique problem. For example, notice that by slightly changing the definition and number of the auxiliary variables in (9.1), the range of the coefficients used in (9.1) can be substantially reduced. Namely, let $\mathbf{y} \in \{0,1\}^{\Delta}$ be defined by $\sum_{k=1}^{\Delta} y_k = K$ if $\chi(\mathcal{G}) = K$ for $K \in \{1,\dots,\Delta\}$. Then the maximum clique problem is equivalent to:

$$\begin{aligned} \chi(\mathcal{G}) = \quad \min \quad & -\textstyle\sum_{i=1}^{|V|} x_i + (\Delta+2)\Big(\sum_{k=1}^{\Delta} y_k - \sum_{i=1}^{|V|} x_i\Big)^2 + \\ & \tfrac{1}{2}\Big(\textstyle\sum_{k=1}^{\Delta} y_k\Big)\Big(-1+\sum_{k=1}^{\Delta} y_k\Big) - \sum_{(i,j)\in E} x_i x_j \\ \text{s.t.} \quad & \mathbf{x} \in \{0,1\}^{|V|}, \mathbf{y} \in \{0,1\}^{\Delta}. \end{aligned} \tag{9.2}$$

Note that the QUBO problem (9.2) uses coefficients that belong to a much smaller range $[-2(\Delta+2), 4(\Delta+2)+1]$ than the range of coefficients used in the QUBO problem (9.1) (after disregarding constant terms and appropriately replacing $x_i \to x_i^2$, $i = 1,\dots,|V|$, $y_k \to y_k^2$, $k = 1,\dots,\Delta$ in the objective of (9.1) to make it an homogenous quadratic). However, a much better QUBO formulation for the maximum clique problem can be obtained by using the fact that $\chi(\mathcal{G}) = \alpha(\mathcal{G}^c)$ [see, e.g., 355], where for a graph $\mathcal{G}(V,E)$, $\mathcal{G}^c = \mathcal{G}(V,E^c)$ is the complement of $\mathcal{G}$, and $\alpha(\mathcal{G})$ stands for the *stable set number* of the graph $\mathcal{G}$ [see, e.g., 348]; that is, the size of the largest cardinality set $S \subseteq V$, such that there are no edges between the nodes in S. This fact can be used to show that (see, e.g., [327, Thm. 6] or [355, Thm. 2.3], among others)

$$\chi(\mathcal{G}) = \alpha(\mathcal{G}^c) = \min\left\{ -\sum_{i=1}^{|V|} x_i + 2 \sum_{(i,j)\notin E} x_i x_j : \mathbf{x} \in \{0,1\}^{|V|} \right\} \tag{9.3}$$

Note that the QUBO problem (9.3) uses $|V|$ logical qubits and coefficients that belong to the range $\{-1,2\}$. Thus, in terms of number of logical qubits and range of the coefficients used in the QUBO reformulation, (9.3) improves both (9.2) and (9.1). It is worth pointing out that the QUBO reformulation (9.3) has been stated in numerous articles [see, e.g., 349, 355–357,

to name a few]. Moreover, it is well known that the range of the coefficients in (9.3) can be further reduced to $\{-1, 1\}$. Namely, it has been proved (or stated) in numerous articles [see, e.g., 320, 326, 348–350, 358] that

$$\chi(\mathcal{G}) = \alpha(\mathcal{G}^c) = \min\left\{-\sum_{i=1}^{|V|} x_i + \sum_{(i,j)\notin E} x_i x_j : \mathbf{x} \in \{0,1\}^{|V|}\right\} \tag{9.4}$$

There is, however, a caveat in the QUBO reformulation (9.4). For any $\mathbf{x} \in \mathbb{R}^n$, let $\text{supp}(\mathbf{x}) = \{i \in \{1,\ldots,n\} : x_i \neq 0\}$. Unlike for (9.1)–(9.3), given $\mathbf{x}^* \in \arg\min\{(9.4)\}$, $\text{supp}(\mathbf{x}^*)$ might not be a clique on $\mathcal{G}$ (nor an independent set in $\mathcal{G}^c$). That is, while the QUBO problems (9.1)–(9.3) are equivalent to the maximum clique problem in terms of both objective value and (loosely speaking) optimal solution, in general, the QUBO problem (9.4) is equivalent to the maximum clique problem *only* in terms of objective value. This important topic will be revisited and discussed in detail in Section 9.3.2.

Along these lines, in what follows, we consider the problem of obtaining not only a QUBO reformulation, but improved QUBO reformulation of a keystone COPT problem; namely, the *maximum k-colorable subgraph* (MkCS) problem [see, e.g., 340].

9.3 The k-subgraph coloring problem

Let $k \geq 1$ colors and a Graph $\mathcal{G}(V, E)$ on n vertices be given. A subgraph $\mathcal{H}$ of $\mathcal{G}$ is k-colorable if we can assign to each vertex of $\mathcal{H}$ a color such that no two adjacent vertices in $\mathcal{H}$ have the same color. The maximum k-colorable subgraph problem (MkCS) aims at finding a k-colorable subgraph $\mathcal{H}$ of $\mathcal{G}$ with maximum cardinality. To model this problem, notice that any k-coloring of a subgraph of $\mathcal{G}$ can be encoded in the following way. For any $i \in [n]$

(where for any $t \in \mathbb{N}$, $[t] := \{1, \ldots, t\}$) and $r \in [k]$, let

$$x_{ir} = \begin{cases} 1, & \text{if vertex } i \in [n] \text{ is colored with color } r \in [k], \\ 0, & \text{otherwise.} \end{cases} \tag{9.5}$$

Then, $\mathbf{x} \in \{0,1\}^{n \times k}$ defines a k-coloring of a subgraph of $\mathcal{G}$ if and only

$$\begin{aligned} &x_{ir} + x_{jr} \leq 1, \text{ for all } (i,j) \in E, r \in [k], \\ &\sum_{r \in [k]} x_{ir} \leq 1, \text{ for all } i \in [n]. \end{aligned} \tag{9.6}$$

Then, the MkCS can be formulated as [see, e.g., 340]:

$$\begin{aligned} \alpha_k(\mathcal{G}) := \quad & \max_{\mathbf{x} \in \{0,1\}^{n \times k}} \quad \sum_{i \in [n], r \in [k]} x_{ir} \\ & \text{s.t.} \qquad\quad x_{ir} + x_{jr} \leq 1, \quad \text{for all } (i,j) \in E, r \in [k], \\ & \qquad\qquad \sum_{r \in [k]} x_{ir} \leq 1, \quad \text{for all } i \in [n]. \end{aligned} \tag{9.7}$$

The MkCS problem falls into the class of NP-complete problems [359]. Moreover, even approximating this problem is known to be NP-hard [360]. For $k = 1$, the MkCS is equivalent to the maximum stable set problem (i.e., $\alpha_1(\mathcal{G}) = \alpha(\mathcal{G})$) that has been widely and thoroughly studied in the literature; and in particular, in the quantum computing literature [see, e.g., 326, 356, 358]. The cases $k = 2$, which is also referred to as the maximum bipartite subgraph problem, and $k > 2$ are considered significantly less in the literature [see 340, for details]. However, as mentioned earlier, the MkCS problem arises in channel assignment in spectrum sharing networks (e.g., Wi-Fi or cellular) [341, 342], VLSI design [343], human genetic research [343, 344], telecommunications [345], and cybersecurity [346]. Thus, a range of approaches have been studied in the literature to address the solution of the MkCS problem, for example, using semidefintie optimization techniques [see, e.g., 340, 361] or integer programming techniques [see, e.g., 327, 362, 363].

Next, we obtain and characterize QUBO reformulations for the MkCS problem that allow to address its solution using quantum technology. Before presenting these results, let us mention some additional facts about the MkCS problem that will be relevant to the discussion in what follows.

Notice that a MkCS $\mathcal{H}$ of $\mathcal{G}(V,E)$ can be recovered from any $\mathbf{x}^* \in \arg\max\{\alpha_k(\mathcal{G})\}$; that is, $H := \mathcal{G}(V_H, E_H)$, where $V_H = \{i \in [n] : \mathbf{x}^*_{ir} > 0 \text{ for some } r \in [k]\}$, $E_H := \{(i,j) \in E : i,j \in V_H\}$, and the coloring of the vertices is obtained by coloring vertex $i \in V_H$ with color $r \in [k]$ if and only if $\mathbf{x}^*_{ir} = 1$. Furthermore, given $\widetilde{\mathbf{x}} \in \{0,1\}^{n\times k}$, it is very simple to obtain a feasible solution $\mathbf{x}' \in \{0,1\}^{n\times k}$ for the MkCS problem by sequentially dropping color $r' \in [k]$ from vertex $i' \in [n]$; that is, setting $\widetilde{x}_{i'r'} = 0$, if $\widetilde{x}_{i'r'} = 1$ and there exists $(i',j) \in E$ such that $\widetilde{x}_{i'r'} + \widetilde{x}_{jr'} > 1$ or $\sum_{r\neq r'} \widetilde{x}_{i'r} \geq 1$. This simple fact is formally stated in Algorithm 12, in a particular form that will be helpful in stating some of the QUBO characterization results that follow.

Algorithm 12 MkCS feasibility

Input $k \geq 1$, $\mathcal{G}(V,E)$, $|V| = n$, $\mathbf{x} \in \{0,1\}^{n\times k}$
for $i \in [n]$, $(i,j) \in E$, $r \in [k]$ **do**
  if $x_{ir} + x_{jr} > 1$ **then**
    $x_{ir} \to 0$
for $i \in [n]$, $r \in [k]$ **do**
  if $x_{ir} = 1$ and $\sum_{p\neq r\in[k]} x_{ip} \geq 1$ **then**
    $x_{ir} \to 0$
Output $\mathbf{x}' := \mathbf{x}$ a feasible solution for the MkSC problem

9.3.1 Linear-based QUBO reformulation

Based on the formulation (9.7) of the MkCS problem in which all the constraints, except for the binary variable constraints are *linear*, we can derive and characterize a *linear-based* QUBO reformulation for the MkCS problem. For that purpose, let us first introduce some notation.

Given $k \geq 1$, a Graph $\mathcal{G}(V,E)$ on n vertices, and $\mathbf{x} \in \{0,1\}^{n\times k}$, $\mathbf{s} \in \{0,1\}^{|E|\times k}$, $\mathbf{t} \in \{0,1\}^n$, let

$$H_0(\mathbf{x}) = \sum_{i\in[n],r\in[k]} x_{ir}^2, \tag{9.8}$$

and

$$H_1^l(\mathbf{x},\mathbf{s}) = \sum_{(i,j)\in E,r\in[k]} \Big(x_{ir} + x_{jr} + s_{ijr} - 1\Big)^2, \tag{9.9a}$$

$$H_2^l(\mathbf{x},\mathbf{t}) = \sum_{i\in[n]} \left(\sum_{r\in[k]} x_{ir} + t_i - 1\right)^2. \tag{9.9b}$$

Furthermore, we define the following simple mappings. Given $\mathbf{x} \in \{0,1\}^{n\times k}$ and $i' \in [n], r' \in [k]$, let the mapping $\mathcal{X}_{i'r'}(\mathbf{x}) : \{0,1\}^{n\times k} \to \{0,1\}^{n\times k}$ be defined by

$$x_{ir} \to \begin{cases} 0 & \text{if } i = i', r = r' \\ x_{ir} & \text{otherwise} \end{cases}, i \in [n], r \in [k]. \tag{9.10}$$

Note that $\mathcal{X}_{i'r'}(\mathbf{x})$ is a generalization of the mapping used on proofs regarding QUBO reformulations of the stable set number problem (i.e., M1CS) [see, e.g., 320, 348–350, 358]. Here, however, to deal with the general case $k > 1$, we need an additional mapping.

Given $p = \{0,1\}$, $\mathbf{s} \in \{0,1\}^{|E|\times k}$, $\mathbf{t} \in \{0,1\}^n$, and $(i',j') \in E$, $r' \in [k]$, let the mapping $\mathcal{M}^p_{i'j',r'}(\mathbf{s},\mathbf{t}) : \{0,1\}^{|E|\times k+n} \to \{0,1\}^{|E|\times k+n}$ be defined by

$$s_{ijr} \to \begin{cases} 1 - s_{i'jr'} & \text{if } i = i', j \neq j', r = r' \\ (1 - s_{i'jr'})p & \text{if } i = i', j = j', r = r' \\ s_{ijr} & \text{otherwise} \end{cases}, (i,j) \in E, r \in [k], \tag{9.11a}$$

$$t_i \to \begin{cases} 1 - p & \text{if } i = i', \\ t_i & \text{otherwise} \end{cases}, i \in [n]. \tag{9.11b}$$

With these definitions in hand, we can now obtain the desired linear-based QUBO reformulation of the MkCS problem. For any $c_1, c_2 > 0$ define the QUBO problem:

$$\begin{aligned} \mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G}) := \quad & \max \quad H^l_{c_1,c_2}(\mathbf{x},\mathbf{s},\mathbf{t}) := H_0(\mathbf{x}) - c_1 H_1^l(\mathbf{x},\mathbf{s}) - c_2 H_2^l(\mathbf{x},\mathbf{t}) \\ & \text{s.t.} \quad \mathbf{x} \in \{0,1\}^{n\times k}, \mathbf{s} \in \{0,1\}^{|E|\times k}, \mathbf{t} \in \{0,1\}^n. \end{aligned} \tag{9.12}$$

Theorem 9.3.1 (linear-based QUBO reformulation of MkCS problem)**.** Let $k \geq 1$ and a Graph $\mathcal{G}(V,E)$ on n vertices be given. Then, for any $c_1 > 1, c_2 > 1$, $\mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G}) = \alpha_k(\mathcal{G})$, and if $\widetilde{\mathbf{x}} \in \arg\max_x\{\mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G})\}$ then $\widetilde{\mathbf{x}} \in \arg\max\{\alpha_k(\mathcal{G})\}$.

Proof. First, notice that $\widetilde{\mathbf{x}}$ is well defined and $\mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G})$ is attained as (9.12) is defined over a compact feasible set. Also, notice that for any $c_1, c_2 > 0$ and any feasible solution $\mathbf{x}' \in \{0,1\}^{n\times k}$ for the MkCS problem (9.7) with objective value $z(\mathbf{x}') := \sum_{i\in[n],r\in[k]} x_{ir}$, one can construct a feasible solution for (9.12); that is, $\mathbf{x} = \mathbf{x}'$, $s_{ijr} = 1 - \mathbf{x}'_{ir} - \mathbf{x}'_{jr}$, for all $(i,j) \in E, r \in [k]$, and $t_i = 1 - \sum_{r\in[k]} \mathbf{x}'_{ir}$, with objective value $H^l_{c_1,c_2}(\mathbf{x},\mathbf{s},\mathbf{t}) = z(\mathbf{x}')$. Thus, if $c_1, c_2 > 0$, the QUBO problem (9.12) is a relaxation of (9.7), and consequently $\mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G}) \geq \alpha_k(\mathcal{G})$. Thus, to prove the result, it is enough to show that when $c_1, c_2 > 1$, one has that $\widetilde{\mathbf{x}}$ is a feasible solution for (9.7). By contradiction, assume this is not the case and let $c_1, c_2 > 1$, $(\widetilde{\mathbf{s}},\widetilde{\mathbf{t}}) := \arg\max_{(\mathbf{s},\mathbf{t})}\{\mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G})\}$. Then either: (1) there is at least an $(i',j') \in E$ and $r' \in [k]$ such that $\widetilde{x}_{i'r'} + \widetilde{x}_{j'r'} > 1$; or (2) there is at least an $i' \in [n]$ and $r' \in [k]$ such that $\widetilde{x}_{i'r'} = 1$ and $\sum_{r\neq r'\in[k]} \widetilde{x}_{i'r} \geq 1$.

For case (1), consider the feasible solution $(\mathbf{x},\mathbf{s}^0,\mathbf{t}^0) \in \{0,1\}^{n\times k+|E|\times k+n}$ for (9.12) obtained from $(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}})$ by letting $(\mathbf{x},\mathbf{s},\mathbf{t}) = (\mathcal{X}_{i'r'}(\widetilde{\mathbf{x}}), \mathcal{M}^0_{i'j',r'}(\widetilde{\mathbf{s}},\widetilde{\mathbf{t}}))$ (cf., (9.10), (9.11)). It then follows from (9.8), (9.10), and the fact that $\widetilde{x}_{i'r'} = 1$ that

$$H_0(\mathbf{x}) = H_0(\widetilde{\mathbf{x}}) - 1. \tag{9.13}$$

Also, from (9.9a), (9.10), (9.11a), and the fact that $\widetilde{x}_{i'r'} = \widetilde{x}_{j'r'} = 1$, it follows that $-H^l_1(\mathbf{x},\mathbf{s}^0) = -H^l_1(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}}) + \sum_{(i',j\neq j')\in E} 4\widetilde{x}_{jr'}\widetilde{s}_{i'jr'} + (1 + \widetilde{s}_{i'j'r'})^2$. Thus,

$$-H^l_1(\mathbf{x},\mathbf{s}^0) \geq -H^l_1(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}}) + 1. \tag{9.14}$$

Further, from (9.9b), (9.10), (9.11b), and the fact that $\widetilde{x}_{i'r'} = 1$, it follows that $-H^l_2(\mathbf{x},\mathbf{t}^0) = -H^l_2(\widetilde{\mathbf{x}},\widetilde{\mathbf{t}}) + 2\widetilde{t}_{i'}\sum_{r\neq r'\in[k]} \widetilde{x}_{i'r} + \widetilde{t}^2_{i'}$. Thus,

$$-H^l_2(\mathbf{x},\mathbf{t}^0) \geq -H^l_2(\widetilde{\mathbf{x}},\widetilde{\mathbf{t}}). \tag{9.15}$$

Using (9.13), (9.14), (9.15), it follows that $H^l_{c_1,c_2}(\mathbf{x},\mathbf{s}^0,\mathbf{t}^0) \geq H^l_{c_1,c_2}(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}}) - 1 + c_1 > H^l_{c_1,c_2}(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}}) = \mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G})$, which contradicts the optimality of $(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}})$ for (9.12).

We proceed analogously for case (2). Consider the feasible solution $(\mathbf{x},\mathbf{s},\mathbf{t}) \in \{0,1\}^{n\times k+|E|\times k+n}$ for (9.12) obtained from $(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}})$ by letting $(\mathbf{x},\mathbf{s}^1,\mathbf{t}^1) = (\mathcal{X}_{i'r'}(\widetilde{\mathbf{x}}), \mathcal{M}^1_{i',j'}(\widetilde{\mathbf{s}},\widetilde{\mathbf{t}}))$ (cf., (9.10), (9.11)). It then follows from (9.8), (9.10), and the fact that $\widetilde{x}_{i'r'} = 1$ that (9.13) holds. Also, from (9.9a), (9.10), (9.11a), and the fact that $\widetilde{x}_{i'r'} = 1$, it follows that $-H^l_1(\mathbf{x},\mathbf{s}^1) = -H^l_1(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}}) + \sum_{(i',j)\in E} 4\widetilde{x}_{jr'}\widetilde{s}_{i'jr'}$. Thus,

$$-H^l_1(\mathbf{x},\mathbf{s}^1) \geq -H^l_1(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}}). \tag{9.16}$$

Further, from (9.9b), (9.10), (9.11b), and the fact that $\widetilde{x}_{i'r'} = 1$, $\sum_{r\neq r'\in[k]} \widetilde{x}_{j'r} \geq 1$, it follows that $-H^l_2(\mathbf{x},\mathbf{t}^1) = -H^l_2(\widetilde{\mathbf{x}},\widetilde{\mathbf{t}}) + 2(\sum_{r\neq r'\in[k]} \widetilde{x}_{i'r})(1+\widetilde{t}_{i'}) + \widetilde{\mathbf{t}}^2_{i'} - 1$. Thus,

$$-H^l_2(\mathbf{x},\mathbf{t}^1) \geq -H^l_2(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}}) + 1. \tag{9.17}$$

Using (9.13), (9.16), (9.17), it follows that $H^l_{c_1,c_2}(\mathbf{x},\mathbf{s}^1,\mathbf{t}^1) \geq H^l_{c_1,c_2}(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}}) - 1 + c_2 > H^l_{c_1,c_2}(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}}) = \mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G})$, which contradicts the optimality of $(\widetilde{\mathbf{x}},\widetilde{\mathbf{s}},\widetilde{\mathbf{t}})$ for (9.12).

Therefore $\widetilde{\mathbf{x}}$ satisfies that there is no $(i',j') \in E$ and $r' \in [k]$ such that $\widetilde{x}_{i'r'} + \widetilde{x}_{j'r'} > 1$, or $i' \in [n]$ and $r' \in [k]$ such that $\sum_{r\in[k]} \widetilde{x}_{i'r} > 1$. Therefore $\widetilde{\mathbf{x}}$ is a feasible solution of (9.7), which finishes the proof. □

It is worth to mention that, loosely speaking, the general form of the QUBO reformulation (9.12) for the MkCS problem can be obtained by using the recent results of Lasserre [347, Thm. 2.2]. Namely, one can use this result after reformulating the MkCS problem constraints as equality constraints using the approach described in [347, Sec. 2.3]. Then, after reformulating the problem using $\{1,-1\}$ binary variables (instead of $\{0,1\}$ binary variables), [347, Thm. 2.2] can be used to obtain a QUBO reformulation of the MkCS problem. However, this reformulation would require the use a penalty parameter with a value larger

than nk (cf., with the values of c_1, c_2 in Theorem 9.3.1), and require the use of more auxiliary (i.e., slack) binary variables than the ones used in Theorem 9.3.1. Thus, Theorem 9.3.1 provides an improved QUBO reformulation of the MkCS problem than the one that would be obtained using [347, Thm. 2.2].

Later, in Section 9.3.3, we will further characterize the QUBO reformulation (9.12) for the MkCS. Next, however, we derive and characterize a QUBO reformulation for the MkCS in which no auxiliary (i.e., slack) binary variables are needed.

9.3.2 Nonlinear QUBO reformulation

Next, we obtain an improved QUBO reformulation for the MkCS problem in terms of the number of binary decision variables required in the QUBO reformulation, when compared with the one provided and characterized in Section 9.3.1. For this purpose, first notice that for any $\mathbf{x} \in \{0,1\}^{n\times k}$, the linear constraints in (9.6) are equivalent to the nonlinear constraints

$$\begin{aligned} &x_{ir}x_{jr} = 0, \text{ for all } (i,j) \in E, r \in [k], \\ &x_{ir}x_{ip} = 0, \text{ for all } i \in [n], (r, p \neq r) \in [k] \times [k]. \end{aligned} \tag{9.18}$$

Then, consistent with (9.18), given $k \geq 1$, a Graph $\mathcal{G}(V,E)$ on n vertices, and $\mathbf{x} \in \{0,1\}^{n\times k}$, let

$$H_1^n(\mathbf{x}) = \sum_{(i,j)\in E, r\in[k]} x_{ir}x_{jr}, \tag{9.19a}$$

$$H_2^n(\mathbf{x}) = \sum_{i\in[n]} \left(\sum_{r\in[k], p\neq r\in[k]} x_{ir}x_{ip} \right). \tag{9.19b}$$

With these definitions in hand we can now obtain the desired *nonlinear-based* QUBO reformulation of the MkCS problem. For any $c_1, c_2 > 0$ define the QUBO problem:

$$\begin{aligned} \mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G}) := \quad & \max \quad H^n_{c_1,c_2}(\mathbf{x}) := H_0(\mathbf{x}) - c_1 H_1^n(\mathbf{x}) - c_2 H_2^n(\mathbf{x}) \\ & \text{s.t.} \quad \mathbf{x} \in \{0,1\}^{n\times k}. \end{aligned} \tag{9.20}$$

Theorem 9.3.2 (nonlinear-based QUBO reformulation of MkCS problem)**.** Let $k \geq 1$ and a Graph $\mathcal{G}(V,E)$ on n vertices be given. Then, for any $c_1 > 1, c_2 > 1$, $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G}) = \alpha_k(\mathcal{G})$, and if $\widetilde{\mathbf{x}} \in \arg\max\{\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G})\}$ then $\widetilde{\mathbf{x}} \in \arg\max\{\alpha_k(\mathcal{G})\}$.

Proof. The proof is mostly analogous to the proof of Theorem 9.3.1. First, notice that $\widetilde{\mathbf{x}}$ is well defined and $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G})$ is attained as (9.20) is defined over a compact feasible set. Also, notice that for any $c_1, c_2 > 0$ and any feasible solution $\mathbf{x}' \in \{0,1\}^{n\times k}$ for the MkCS problem (9.7) with objective value $z(\mathbf{x}') := \sum_{i\in[n],r\in[k]} x_{ir}$, one hast that $\mathbf{x}'$ is a feasible solution of (9.20) with objective value $H^n_{c_1,c_2}(\mathbf{x}) = z(\mathbf{x}')$ (i.e., $\mathbf{x}'$ satisfies (9.18)). Thus, if $c_1, c_2 > 0$, the QUBO problem (9.20) is a relaxation of (9.7), and consequently $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G}) \geq \alpha_k(\mathcal{G})$. Thus, to prove the result, it is enough to show that when $c_1, c_2 > 1$, one has that $\widetilde{\mathbf{x}}$ is a feasible solution for (9.7). By contradiction, assume this is not the case and let $c_1, c_2 > 1$. Then either: (1) there is at least an $(i', j') \in E$ and $r' \in [k]$ such that $\widetilde{x}_{i'r'} + \widetilde{x}_{j'r'} > 1$; or (2) there is at least an $i' \in [n]$ and $r' \in [k]$ such that $\widetilde{x}_{i'r'} = 1$ and $\sum_{r\neq r'\in[k]} \widetilde{x}_{i'r} \geq 1$. Notice that in either case $\widetilde{x}_{i'r'} = 1$. Now consider the feasible solution $\mathbf{x} \in \{0,1\}^{n\times k}$ for (9.20) obtained from $\widetilde{\mathbf{x}}$ by letting $\mathbf{x} = \mathcal{X}_{i'r'}(\widetilde{\mathbf{x}})$ (cf., (9.10)). Notice that from (9.8), (9.10), and the fact that $\widetilde{x}_{i'r'} = 1$ one has that (9.13) holds. Also, from (9.19a), (9.10), and the fact that $\widetilde{x}_{i'r'} = 1$, it follows that $-H^n_1(\mathbf{x}) = -H^n_1(\widetilde{\mathbf{x}}) + \sum_{(i',j\neq j')\in E} \widetilde{x}_{jr'} + \widetilde{x}_{j'r'}$. Thus,

$$-H^n_1(\mathbf{x}) \geq -H^n_1(\widetilde{\mathbf{x}}) + \widetilde{x}_{j'r'}. \tag{9.21}$$

Further, from (9.19b), (9.10), and the fact that $\widetilde{x}_{i'r'} = 1$, it follows that

$$-H^n_2(\mathbf{x}) = -H^n_2(\widetilde{\mathbf{x}}) + \sum_{r\neq r'\in[k]} x_{i'r}. \tag{9.22}$$

Using (9.13), (9.21), (9.22), it follows that

$$H^n_{c_1,c_2}(\mathbf{x}) \geq H^n_{c_1,c_2}(\widetilde{\mathbf{x}}) - 1 + c_1\widetilde{x}_{j'r'} + c_2 \sum_{r\neq r'\in[k]} x_{i'r}. \tag{9.23}$$

In case (1), we have that $\widetilde{x}_{j'r'} = 1$. Thus, from (9.23), we have that $H^n_{c_1,c_2}(\mathbf{x}) \geq H^n_{c_1,c_2}(\widetilde{\mathbf{x}}) - 1 + c_1 > H^n_{c_1,c_2}(\widetilde{\mathbf{x}}) = \mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G})$, which contradicts the optimality of $\widetilde{\mathbf{x}}$ for (9.20). Analogously, in case (2), we have that $\sum_{r\neq r'\in[k]} \widetilde{x}_{j'r} \geq 1$. Thus, from (9.23), we have that $H^n_{c_1,c_2}(\mathbf{x}) \geq H^n_{c_1,c_2}(\widetilde{\mathbf{x}}) - 1 + c_2 > H^n_{c_1,c_2}(\widetilde{\mathbf{x}}) = \mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G})$, which contradicts the optimality of $\widetilde{\mathbf{x}}$ for (9.20).

Therefore $\widetilde{\mathbf{x}}$ satisfies that there is no $(i', j') \in E$ and $r' \in [k]$ such that $\widetilde{x}_{i'r'} + \widetilde{x}_{j'r'} > 1$, or $i' \in [n]$ and $r' \in [k]$ such that $\sum_{r\in[k]} \widetilde{x}_{i'r} > 1$. Therefore $\widetilde{\mathbf{x}}$ is a feasible solution of (9.7), which finishes the proof. □

Next, we show that the value of the penalty parameters c_1, c_2 in the definition of $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G})$ in (9.20) can be further reduced to the values $c_1 = c_2 = 1$ (indeed, more generally to $c_1 = 1$, $c_2 \geq 1$, or $c_1 \geq 1$, $c_2 = 1$), while still being able to obtain an optimal solution for the MkCS problem for $\mathcal{G}(V,E)$ by solving the QUBO problem $\mathbf{Q}^n_{1,1}(k,\mathcal{G})$. In this case, $\mathbf{Q}^n_{1,1}(k,\mathcal{G})$ and $\alpha_k(\mathcal{G})$ are equivalent in terms of their optimal objective value, but not necessarily in terms of their optimal solutions. That is, the optimal solution $\widetilde{\mathbf{x}} := \arg\max(\mathbf{Q}^n_{1,1}(k,\mathcal{G}))$ might not necessarily be a feasible solution for the MkCS problem, which hinders the possibility of constructing a MkCS set $\mathcal{H}$ for $\mathcal{G}(V,E)$. However, as we formally show in the next corollary, the $\mathbf{Q}^n_{1,1}(k,\mathcal{G})$ optimal solution $\widetilde{\mathbf{x}}$ can be simply modified to obtain an optimal solution for the MkCS problem.

Corollary 9.3.1 (unit-penalty nonlinear-based QUBO formulation of MkCS problem)**.** Let $k \geq 1$ and a Graph $\mathcal{G}(V,E)$ on n vertices and $c_1, c_2 \geq 0$ be given, and let $\widetilde{\mathbf{x}} := \arg\max\{\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G})\}$ (recall (9.20)). If $c_1 = 1$, $c_2 \geq 1$ or $c_1 \geq 1$, $c_2 = 1$, then $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G}) = \alpha_k(\mathcal{G})$. Furthermore, $\mathbf{x}' \in \arg\max\{\alpha_k(\mathcal{G})\}$, where $\mathbf{x}' \in \{0,1\}^{n\times k}$ is the output obtained when k, $\mathcal{G}(V,E)$, $|V|$, $\mathbf{x} = \widetilde{\mathbf{x}}$ is used as input in Algorithm 12.

Proof. The result follows from the proof of Theorem 9.3.2 and Algorithm 12. More specifically, in the case $c_1 = 1$, $c_2 \geq 1$, notice that Algorithm 12, step (4), is equivalent to applying

the mapping $\mathcal{X}_{ir}(\cdot)$ (recall (9.10)) to the current solution $\mathbf{x}$ in the Algorithm when $x_{ir} = x_{jr} = 1$ for some $(i,j) \in E$, $r \in [k]$. Thus, it follows from (9.23) that the value of $H^n_{c_1,c_2} = H^n_{1,c_2}(\mathbf{x})$ can only increase or stay equal after Algorithm 12, step (4). Similarly, in the case $c_1 \geq 1$, $c_2 = 1$, notice that Algorithm 12, step (7), is equivalent to applying the mapping $\mathcal{X}_{ir}(\cdot)$ (recall (9.10)) to the current solution $\mathbf{x}$ in the Algorithm when $x_{ir} = 1$, $\sum_{p \neq r \in [k]} x_{ip} \geq 1$ for some $i \in [n]$, $r \in [k]$. Thus, it follows from (9.23) that the value $H^n_{c_1,c_2}(\mathbf{x}) = H^n_{c_1,1}(\mathbf{x})$ can only increase of stay equal after Algorithm 12, step (7). Thus, in both cases, at the end of Algorithm 12 one obtains a feasible solution $\mathbf{x}'$ for the MkCS problem with objective $H^n_{c_1,c_2}(\mathbf{x}') \geq H^n_{c_1,c_2}(\overline{\mathbf{x}}) = \mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G})$. Since $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G}) \geq \alpha_k(\mathcal{G})$ (see beginning of proof of Theorem 9.3.2), it follows that $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G}) = \alpha_k(\mathcal{G})$, and $\mathbf{x}' \in \arg\max\{\alpha_k(\mathcal{G}))\}$. □

In light of Theorem 9.3.2 and Corollary 9.3.1, it is natural to consider what happens if in the QUBO problem (9.20) one considers penalty parameters $0 < c_1, c_2 < 1$.

Proposition 9.3.1. Let $k \geq 1$ and $c_1, c_2 > 0$ be given. If $c_1 < 1$ or $c_2 < 1$ and $k \geq 1$, then there exists a Graph $\mathcal{G}(V,E)$ such that $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G}) > \alpha_k(\mathcal{G})$.

Proof. First, consider the case in which $0 < c_1 < 1$, and let $\mathcal{G}(V,E)$ is a clique of $k+1$ vertices. Clearly $\alpha_k(\mathcal{G}) = k$. Now, for all $i \in [k+1]$, $r \in [k]$, let

$$x_{ir} = \begin{cases} 1 & i = r, i \leq k, \\ 1 & i = k+1, r = k, \\ 0 & \text{otherwise.} \end{cases}$$

Then, $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G}) \geq H^n_{c_1,c_2}(\mathbf{x}) = (k+1) - c_1 > k = \alpha_k(\mathcal{G})$. Now, consider the case in which $0 < c_2 < 1$, and let $\mathcal{G}(V,E)$ be the graph on $k+1$ vertices obtained by taking a clique in $k+1$ vertices and adding a vertex $k+2$ and edge $(k+1, k+2)$. That is, $V = [k+2]$, and $E = \{(i,j) : 1 \leq i < j \leq$

$k+1\}\cup\{(k+1,k+2)\}$. Clearly $\alpha_k(\mathcal{G})=k+1$. Now let

$$x_{ir}=\begin{cases}1 & i=r, i\leq k\\ 1 & i=k+2, r=k\\ 1 & i=k+2, r=k-1\\ 0 & \text{otherwise}\end{cases}, \text{ for all } i\in[k+2], r\in[k].$$

Then, $\mathbf{Q}^n_{c_1,c_2}(k,\mathcal{G})\geq H^n_{c_1,c_2}(\mathbf{x})=(k+2)-c_2>k+1=\alpha_k(\mathcal{G})$. □

Remark 9.3.1. Theorem 9.3.2 together with Corollary 9.3.1 and Proposition 9.3.1 fully characterize the QUBO problem (9.20) as a means to obtain a QUBO reformulation of the MkCS problem. In short, for any $c_1, c_2 \geq 1$, solving the nonlinear-based QUBO problem (9.20) is equivalent to solving the MkCS problem with the caveat that if either $c_1 = 1$ or $c_2 = 1$, the simple Algorithm 12 might need to be applied to the optimal solution of (9.20) in order to obtain an optimal solution for the MkCS problem. On the other hand, if $0 < c_1 < 1$ or $0 < c_2 < 1$, solving the nonlinear-based QUBO problem (9.20) is not guaranteed to provide the objective value or the solution to the MkCS problem.

As illustrated in Section 9.4, the full characterization provided in this section (see summary in Remark 9.3.1) gives the freedom to fine-tune the QUBO reformulation of the MkCS problem to make the best use of quantum tools in addressing the solution of this problem.

In finishing this section, recall that the MkCS problem is equivalent to the stable set problem when $k = 1$. Thus the QUBO reformulation results [see, e.g., 320, 326, 327, 348–350, 355, 358] for the stable set problem of the form

$$\alpha(\mathcal{G})=\max\left\{\sum_{i=1}^{n}x_i^2-c_1\sum_{(i,j)\in E}x_ix_j : \mathbf{x}\in\{0,1\}^n\right\}, \tag{9.24}$$

for a given Graph $\mathcal{G}(V,E)$ on n vertices and $c_1 \geq 1$ follow from Theorem 9.3.2 and Corollary 9.3.1. In particular, Corollary 9.3.1 implies results in which c_1 is set to one in (9.24).

However, Corollary 9.3.1 brings up a fact that, to the best of our knowledge, has been ignored in the literature; namely, that when c_1 is set to one in (9.24), the support of the optimal solution of (9.24) might not necessarily correspond to a stable set of the Graph $\mathcal{G}(V,E)$. However, an optimal solution for the stable set problem can be obtained from the optimal solution of (9.24) by applying Algorithm 12 (see Corollary 9.3.1).

9.3.3 Linear-based QUBO reformulation revisited

After the results in Section 9.3.2, which provide a full characterization of the QUBO problem (9.20) to reformulate the MkCS problem, it is natural to consider if a similar full characterization of the QUBO problem (9.12) can be obtained. Indeed, it is not difficult to see that analogous results (with analogous proofs that are not included in the interest of brevity), to Corollary 9.3.1 and Proposition 9.3.1 can be obtained for the QUBO problem (9.12).

Corollary 9.3.2 (unit-penalty linear-based QUBO formulation of MkCS problem)**.** Let $k \geq 1$ and a Graph $\mathcal{G}(V,E)$ on n vertices and $c_1, c_2 \geq 0$ be given, and let $\widetilde{\mathbf{x}} := \arg\max_{\mathbf{x}}\{\mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G})\}$ (recall (9.12)). If $c_1 = 1$, $c_2 \geq 1$ or $c_1 \geq 1$, $c_2 = 1$, then $\mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G}) = \alpha_k(\mathcal{G})$. Furthermore, $\mathbf{x}' \in \arg\max\{\alpha_k(\mathcal{G})\}$, where $\mathbf{x}' \in \{0,1\}^{n\times k}$ is the output obtained when k, $\mathcal{G}(V,E)$, $|V|$, $\mathbf{x} = \widetilde{\mathbf{x}}$ is used as input in Algorithm 12.

Proposition 9.3.2. Let $k \geq 1$ and $c_1, c_2 > 0$ be given. If $c_1 < 1$ or $c_2 < 1$ and $k \geq 1$, then there exists a Graph $\mathcal{G}(V,E)$ such that $\mathbf{Q}^l_{c_1,c_2}(k,\mathcal{G}) > \alpha_k(\mathcal{G})$.

9.4 Benchmarking

To illustrate the benefits of the fully characterized QUBO reformulations presented in Section 9.3, we next benchmark the linear-based (Section 9.3.1) and nonlinear-based (Section 9.3.2) QUBO reformulations of the MkCS problem when solving them with a quantum

annealer. For this purpose, we present results pertaining the *minimum gap* [see, e.g., 364], related to the convergence rate of (an ideal) adiabatic quantum algorithm (AQC), *embedding* [see, e.g., 333] into the available quantum annealing hardware, and *time-to-solution* (TTS) [see, e.g., 365] when performing the quantum annealing.

The embedding and TTS benchmarking results are obtained using D-Wave's quantum annealers*. Specifically, we report the different results obtained when using two different D-Wave processors: 2000Q$^{\text{TM}}$ and Advantage 1.1$^{\text{TM}}$. The main difference between these two processors is the number of available qubits and their connectivity within the processor. The 2000Q$^{\text{TM}}$ processor has 2048 possible qubits (of which 2041 were available) connected in a *Chimera* connectivity graph, designed as a grid of 16×16 cells of $K_{4,4}$ bipartite graphs connected in a nearest-neighbor fashion by means of non-planar edges, where each qubit is connected to at most 6 neighbors [366]. The Advantage 1.1$^{\text{TM}}$ processor counts with 5640 qubits (5510 available) following a *Pegasus* connectivity graph, defined as three layers of 16×16 cells of $K_{4,4}$ bipartite graphs with additional connections within and among the cells, providing an increased connectivity for each qubit to maximum 15 neighbors [84].

Each QUBO reformulation proposed here is thus used to solve the MkCS problem in a D-Wave quantum annealer. These numerical experiments are similar in nature to those carried out in [327, 337–339] to compare different QUBO formulations of various COPT problems.

To generate instances $\mathcal{G}(V,E)$ for the numerical tests, given the number of nodes $|V|=n$, we generate instances with randomly defined edge set E, using Erdős-Rényi graphs $\mathcal{G}(n,p)$ with probabilities $p=0.25,0.5,0.75$ (i.e., with different levels of sparsity). In what follows, for the purpose of brevity, we will sometimes refer to the linear-based QUBO reformulation (Section 9.3.1) as L-QUBO, and to the nonlinear-based QUBO reformulation (Section 9.3.2) as

*https://www.dwavesys.com/

N-QUBO. All the classical computations are done using an Ubuntu Machine with processor Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz, 94Gb of RAM, and 20 cores.

9.4.1 Quantum Annealing

Before presenting the benchmarking results, we provide a very brief high level discussion about the ideal version of the quantum annealing algorithm run by D-Wave's quantum annealer; that is, an AQC algorithm [following 367, 368]. For additional details, the reader is directed to [80, 369, 370, among many others]. The fact that complex COPT problems (such as the MkCS problem considered here) can be reformulated as QUBO problems, means that finding the optimal solution of the problem can be regarded as finding or sampling low-energy states from an Ising spin model Hamiltonian $\mathcal{H}_f$ that is constructed from the QUBO formulation (i.e., using [367, Eq. (14)]). For that purpose, a simple Hamiltonian $\mathcal{H}_i$ with an easily prepared ground (low-energy) state is prepared (i.e., using [367, Eq. (12)]). Then, by constructing an adequate interpolation [see, e.g., 367, Eq. (1)] between the $\mathcal{H}_i$ and $\mathcal{H}_f$ Hamiltonians, the system can be set to slowly evolve (so that the *adiabatic* theorem [cf., 80] is satisfied) from the ground state of $\mathcal{H}_i$ to a state that will yield the desired low-energy state of $\mathcal{H}_f$ with high probability. In particular, here we construct the interpolation:

$$\mathcal{H}(s) = \frac{A(s)}{2}\mathcal{H}_i + \frac{B(s)}{2}\mathcal{H}_f, \tag{9.25}$$

where $s \in [0,1]$ is the adimensional (or reduced) time $s = \frac{t}{T}$ with t denoting time and T denoting the computation time, $A(s)$ is the tunneling energy curve, and $B(s)$ is the problem's Hamiltonian energy curve, for all $s \in [0,1]$. The specifications of $A(s)$ and $B(s)$ for both the D-Wave 2000Q$^{\text{TM}}$ and Advantage 1.1$^{\text{TM}}$ processors can be found in D-Wave's documentation [368].

9.4.2 Minimum Gap

Now we consider the evolution of the AQC algorithm under (9.25). It is known that the *minimum gap*; that is, the minimum energy gap between the lowest two energy levels during the evolution of the AQC algorithm determines the time of the computation [see, e.g., 371, 372]. Such energy levels correspond with the eigenvalues $E_m(s)$ of the eigenstates $m;s\rangle$ of the Hamiltonian $\mathcal{H}(s)$, where $m \in \{0,1,\ldots,2^n-1\}$ in an n qubit system, and are given by [see, 367]:

$$\mathcal{H}(s)|m;s\rangle = E_m(s)|m;s\rangle, \tag{9.26}$$

with $E_0(s) \leq E_1(s) \leq \cdots \leq E_{2^n-1}$. Thus, the minimum gap, denoted by $\Delta_{\min}$, is given by [see, 367]:

$$\Delta_{\min} = \min_{s\in[0,1]} \{E_1(s) - E_0(s)\}. \tag{9.27}$$

The minimum gap $\Delta_{\min}$ provides a lower bound on the AQC computation time that is inversely proportional to $\Delta^2_{\min}$ [see, 367, Eq. (9)]; that is, the larger $\Delta_{\min}$, the faster the AQC algorithm is expected to converge to the ground state of the Hamiltonian $\mathcal{H}_f$ [see, e.g., 370].

Here, we use the exact diagonalization of the instantaneous time-dependent Hamiltonian $\mathcal{H}(s)$ to compute $\Delta_{\min}$. Although this methodology is well known to require a prohibitive amount of computation due to the need to diagonalize matrices of size $2^n \times 2^n$ for a system with n qubits; it is suitable for the illustrative numerical tests performed here (for less computationally expensive ways to approximately compute $\Delta_{\min}$, we refer the readers to [372]).

In particular, to obtain the minimum gap results presented next, we begin by calculating, for a number of finite values of $s \in [0,1]$ (see details below), the Hamiltonian $\mathcal{H}(s)$ in (9.25) for particular instances of the L-QUBO (resp. N-QUBO) formulation of the MkCS problem. Then, we verify which eigenvalues of $\mathcal{H}(s)$ correspond to different states in the annealing

process. Two states are considered different if at the end of the annealing, their energy difference is more than a given ε. Here, we use $\varepsilon = 1$ GHz. Finally, we approximately compute the minimum difference along the annealing scaled time s of the two smallest eigenvalues corresponding to different states (the ground state and the first excited state). The approximation comes from the fact that the minimum in (9.27) is computed over a finite set of values of $s \in [0,1]$. Specifically, given the monotonic behavior of $A(s)$ and $B(s)$, we observe that $\Delta_{\min}$ is attained at a value $s \in [0,1]$ that is close to the value of $s \in [0,1]$ in which the maximum of the minimum eigenvalue is attained. Moreover, for the instances considered here, the minimum eigenvalue is concave on $s \in [0,1]$, allowing us to more efficiently sample the domain $[0,1]$ in search for the value of $\Delta_{\min}$. Namely, we first consider a coarse discretization of the domain $[0,1]$ (taking only 10 equally spaced elements of the set) and then determine the three points whose middle point would be larger than both its neighbors. This interval contains the values of $s \in [0,1]$ in which both the maximum value of the smallest eigenvalue, and $\Delta_{\min}$ is attained. We sample this interval by computing 10 points between each of these points (including them) to refine the approximation to the value s^* in which the minimization in (9.27) is attained. This procedure only requires 44 computations of the eigenvalues of the Hamiltonian $\mathcal{H}(s)$.

9.4.2.1 Minimum Gap Results

Next, we compare the minimum gap ($\Delta_{\min}$) resulting when using the L-QUBO reformulation and the N-QUBO reformulations for the MkCS problem with varying penalty parameters (i.e., c_1, c_2) on small instances of the MkCS problem.

In particular, Figures 9.1 and 9.2 compare the $\Delta_{\min}$ obtained from the L-QUBO (9.12) and N-QUBO (9.20) for instances of the MkCS problem in which $k = 1$, where the underlying graphs are randomly selected $\mathcal{G}(5, 0.25)$ and $\mathcal{G}(5, 0.75)$ graphs. These bar plots, as well

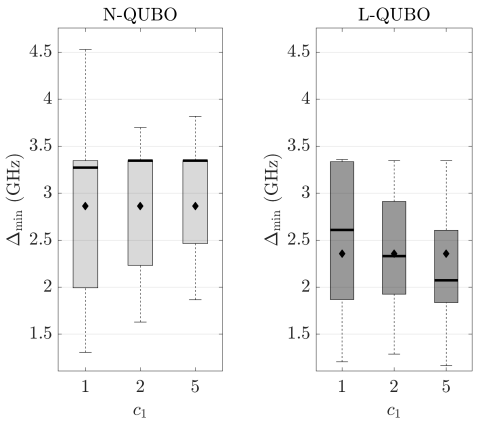


Figure 9.1: Mingap: $k = 1$, $\mathcal{G}(5, 0.25)$.

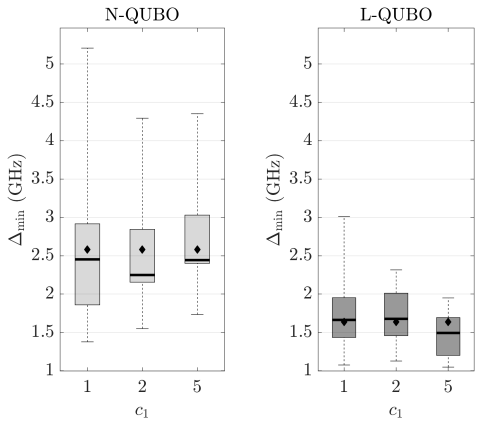


Figure 9.2: Mingap: $k = 1$, $\mathcal{G}(5, 0.75)$.

as the remaining ones in this section, provide information about the distribution of $\Delta_{\min}$. Specifically, each bar is obtained by computing $\Delta_{\min}$ on 100 randomly generated graphs $\mathcal{G}(5, p)$, $p \in \{0.25, 0.50, 0.75\}$, for different values of the penalization parameters. The tick line, represents the median of $\Delta_{\min}$, the black diamond represents $\overline{\Delta}_{\min}$, the average of $\Delta_{\min}$, the bar encompasses the values within the 25% and 75% quantiles of $\Delta_{\min}$'s distribution, and the dotted interval encompasses the values within the 0% and 100% of $\Delta_{\min}$'s distribution.

From Figures 9.1 and 9.2, it follows that the N-QUBO results in higher $\overline{\Delta}_{\min}$ than the L-QUBO; therefore, in theory, the N-QUBO Hamiltonian should converge faster to a low energy state than the L-QUBO Hamiltonian. We can state this more formally by performing a simple hypothesis test. Let $\overline{\Delta}^{a}_{\min}(k, p, c_1, c_2)$ (resp. $\mu_{\Delta^{a}_{\min}}(k, p, c_1, c_2)$) be the average (resp. mean) of the minimum gap for the a-QUBO formulation of the MkCS instance from $\mathcal{G}(5, p)$ graphs, and penalty parameters c_1, c_2. Then, consider the hypothesis test:

$$\begin{aligned} H_o: &\quad \mu_{\Delta^{\mathrm{L}}_{\min}}(1, p, 1, \cdot) \geq \mu_{\Delta^{\mathrm{N}}_{\min}}(1, p, 1, \cdot) - \delta\overline{\Delta}^{\mathrm{N}}_{\min}(1, p, 1, \cdot) \\ H_a: &\quad \mu_{\Delta^{\mathrm{L}}_{\min}}(1, p, 1, \cdot) < \mu_{\Delta^{\mathrm{N}}_{\min}}(1, p, 1, \cdot) - \delta\overline{\Delta}^{\mathrm{N}}_{\min}(1, p, 1, \cdot), \end{aligned} \tag{9.28}$$

where $\delta \in [0, 100\%]$. That is, in (9.28) we are statistically comparing the left-most bars of the N-QUBO subplot and the L-QUBO subplot of Figures 9.1 and 9.2 (for brevity, the results for

the case $p = 0.50$ have not bee plotted), under the null hypothesis that the L-QUBO provides a higher mean $\Delta_{\min}$. Then, for any $p \in \{0.25, 0.50, 0.75\}$ one gets that the null hypothesis H_o in (9.28) can be rejected with 95% confidence for values of δ up to 2%. Thus, loosely speaking, the N-QUBO results in values of $\Delta_{\min}$ that on average are 2% higher than the ones obtained by the L-QUBO, when using penalty parameter $c_1 = 1$.

One advantage of having the full characterization of the penalty constants, for which the QUBO formulations (9.12) and (9.20) become reformulations of the MkCS problem, is that we can investigate what are the trade-offs of increasing such penalty values from their minimum ones. Intuitively, one might expect that $\Delta_{\min}$ increases (faster convergence) as the values of the penalty parameters c_1, c_2 increase. This reasoning stem from the fact that higher penalty parameters increase the suboptimality of infeasible solutions of the original problem in its associated QUBO reformulation. However, from Figures 9.1 (left) and 9.2 (left) it follows that $\overline{\Delta}_{\min}$ remains fairly unchanged as the penalty parameter c_1 increases. More formally, consider a similar hypothesis test to the one considered in (9.28).

$$\begin{aligned} H_o: &\quad \mu_{\Delta^{\mathrm{N}}_{\min}}(1, p, c_1, \cdot) \geq \mu_{\Delta^{\mathrm{N}}_{\min}}(1, p, 1, \cdot) + \delta \overline{\Delta}^{\mathrm{N}}_{\min}(1, p, 1, \cdot) \\ H_a: &\quad \mu_{\Delta^{\mathrm{N}}_{\min}}(1, p, c_1, \cdot) < \mu_{\Delta^{\mathrm{N}}_{\min}}(1, p, 1, \cdot) + \delta \overline{\Delta}^{\mathrm{N}}_{\min}(1, p, 1, \cdot). \end{aligned} \tag{9.29}$$

Then, for any $p \in \{0.25, 0.50, 0.75)\}, c_1 \in \{2, 5\}$, one gets that the null hypothesis H_o in (9.28) can be rejected with 95% confidence for values of δ less than 1%. Thus, loosely speaking, the N-QUBO with penalty parameter $c_1 = 1$ results in values of $\Delta_{\min}$ that on average are not 1% lower than the ones obtained by the N-QUBO with higher penalty parameters $c_1 \in \{2, 5\}$.

It is clearly interesting to investigate how the characteristics above look when considering higher values of k. Due to the complexity of the exact diagonalization procedure used to compute $\Delta_{\min}$, we limit to the study of the case $k = 2$ for the N-QUBO reformulation, considering penalty parameters constants $c_1 \in \{1, 2, 5\}$, and $c_2 \in \{1, 2, 5\}$.

Much like in the case when $k = 1$, from Figures 9.3 and 9.4 it follows that $\overline{\Delta}_{\min}$ barely

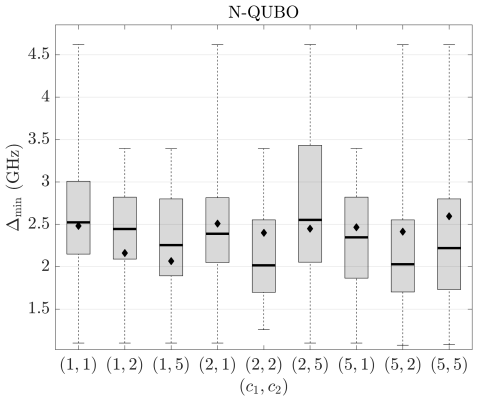


Figure 9.3: Mingap: $k = 2$, $\mathcal{G}(5, 0.25)$.

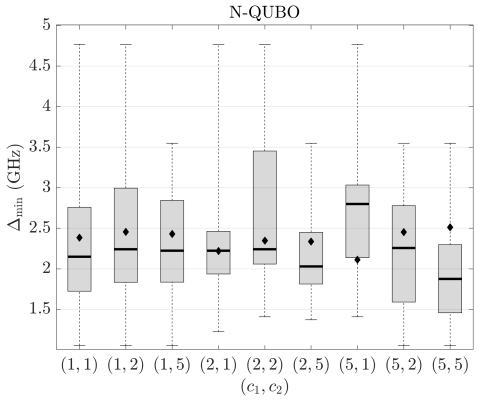


Figure 9.4: Mingap: $k = 2$, $\mathcal{G}(5, 0.75)$.

increases as the penalty parameters c_1, c_2 increase, in comparison to the value of $\overline{\Delta}_{\min}$ when the penalty parameters are set to $c_1 = c_2 = 1$. Statistically, things are a bit different. Formally, consider a similar hypothesis test to the one considered in (9.29).

$$\begin{aligned} H_o: &\quad \mu_{\Delta^{\mathrm{N}}_{\min}}(2, p, c_1, c_2) \geq \mu_{\Delta^{\mathrm{N}}_{\min}}(2, p, 1, 1) + \delta \overline{\Delta}^{\mathrm{N}}_{\min}(2, p, 1, 1) \\ H_a: &\quad \mu_{\Delta^{\mathrm{N}}_{\min}}(2, p, c_1, c_2) < \mu_{\Delta^{\mathrm{N}}_{\min}}(2, p, 1, 1) + \delta \overline{\Delta}^{\mathrm{N}}_{\min}(2, p, 1, 1). \end{aligned} \tag{9.30}$$

Table 9.1 shows the minimum value of δ in (9.30) for which the null hypothesis H_o in (9.29) can be rejected with a 95% confidence level. Loosely speaking, the value of δ indicates the percentage by which the value of $\overline{\Delta}_{\min}(2, p, 1, 1)$ must be higher in order to reject the hypothesis that larger penalty parameters (i.e., larger than $c_1 = c_2 = 1$) result in a larger mean value of $\Delta_{\min}$.

Overall, there is a recognizable pattern in Table 9.1. Namely, it is clear that the sparser the underlying graph (i.e., lower probability p) used to construct the instance of the MkCS problem, the smaller the effect of increasing penalty parameters is on increasing the mean of $\Delta_{\min}$ (i.e., accelerating convergence of an AQC algorithm). This is intuitively expected, given that sparsity in the underlying graph results in a lower number of penalty terms in the N-QUBO (9.20). In particular, notice that a significant increase in the mean of $\Delta_{\min}$, for sparse underlying graphs (i.e., $p = 0.25$), only arises when the penalty parameters are

	δ			
(c_1, c_2)	$p = 0.25$	$p = 0.50$	$p = 0.75$	H_o
(1,2)	-5%	8%	12%	Reject
(1,5)	-5%	5%	11%	Reject
(2,1)	1%	4%	1%	Reject
(2,2)	1%	12%	7%	Reject
(2,5)	1%	13%	6%	Reject
(5,1)	1%	2%	0%	Reject
(5,2)	4%	15%	11%	Reject
(5,5)	11%	22%	13%	Reject

Table 9.1: Hypothesis test (9.30) for different parameters with 95% confidence.

increased from $c_1 = c_2 = 1$ to $c_1 = c_2 = 5$. In contrast, for non-sparse underlying graphs (i.e., $p = 0.75$), increases in the penalty constants above $c_1 = c_2 = 1$ bring increases in the mean of $\Delta_{\min}$ of about 10% in most cases. In Section 9.4.4, we will analyze how these increases in the mean of $\Delta_{\min}$ affect the convergence to a solution in D-Wave's quantum annealing devices. Before doing this analysis, we first consider the differences in terms of embedding requirements between the N-QUBO and L-QUBO formulation.

9.4.3 Embedding

Current NISQ devices have a low number of qubits available with restricted connectivity. Given this, the number of qubits required to *embed* [see, e.g., 333] a QUBO reformulation of a given COPT problem in a quantum device is a very important benchmark to compare the benefits of different QUBO reformulations of a COPT problem [see, e.g., 335, 337–339]. Next, to benchmark the N-QUBO (9.20) versus the L-QUBO (9.12) reformulation of the MkCS problem, we use the number of qubits needed to embed the QUBO reformulation in both a 2048 qubits *Chimera* connectivity graph (for D-Wave's 2000Q$^{\text{TM}}$ processor), and a 5640 qubits *Pegasus* connectivity graph (for D-Wave's Advantage 1.1$^{\text{TM}}$ processor). For this purpose, we use D-Wave's embedding algorithm [see, e.g., 373, 374, for a discussion of

different embedding algorithms].

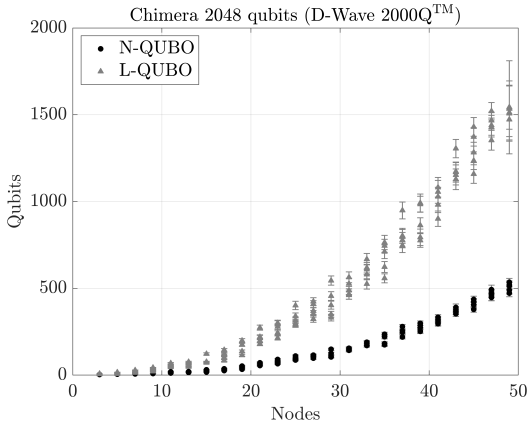


Figure 9.5: Embeding: $k = 1$, $\mathcal{G}(n, 0.25)$.

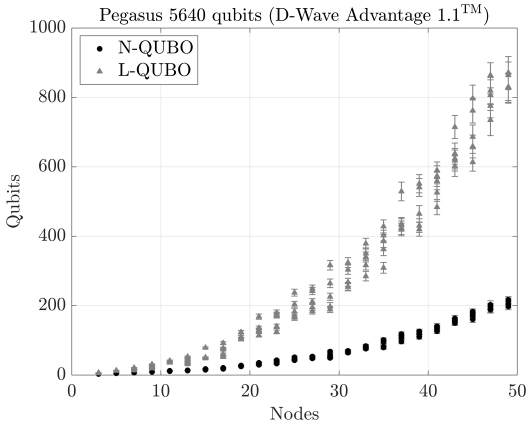


Figure 9.6: Embedding: $k = 1$, $\mathcal{G}(n, 0.25)$.

Note that for a Graph $\mathcal{G}(V, E)$ with n nodes, the number of binary variables required to formulate the L-QUBO for the associated MkCS problem is $k(n + |E| + 1)$, while kn binary variables are needed to formulate the associated N-QUBO. Not surprisingly, the N-QUBO would require less number of qubits than the L-QUBO when these QUBOs are embedded into D-Wave's quantum annealers. The following results show how the increased number of binary variables required by the L-QUBO affects the difference between the qubits required to embed both QUBO formulations.

In Figures 9.5-9.12, the number of average qubits required by both the L-QUBO and the N-QUBO formulations are plotted for values of $k \in \{1, 2, 5\}$, graphs $\mathcal{G}(n, p)$ for values of $n \in [5, 50]$, $p \in \{0.25, 0.50, 0.75\}$, and D-Wave's 2000Q™ and Advantage 1.1™ processors. The average is computed over five (5) random graphs $\mathcal{G}(n, p)$ generated for each combination of n, p values, as well as ten (10) runs of D-Wave's embedding algorithm. The bars plotted with each point in the graph represent the values within one standard deviation of the average value.

From all Figures 9.5-9.12, it is clear that in terms of embedding requirements, the N-QUBO

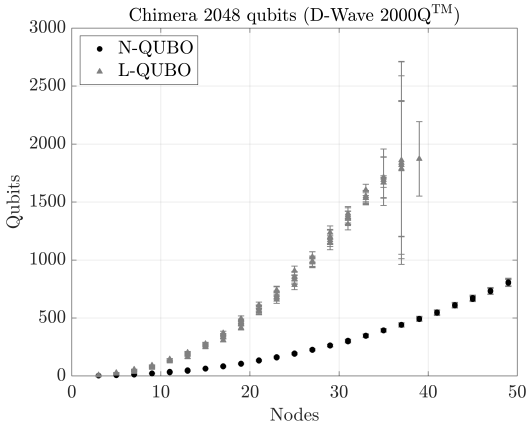


Figure 9.7: Embeding: $k = 1$, $\mathcal{G}(n, 0.75)$.

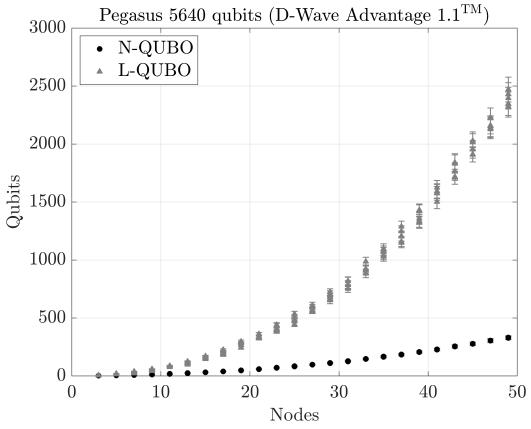


Figure 9.8: Embedding: $k = 1$, $\mathcal{G}(n, 0.75)$.

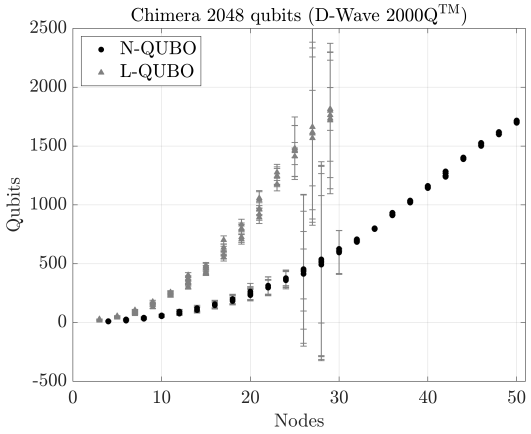


Figure 9.9: Embeding: $k = 2$, $\mathcal{G}(n, 0.50)$.

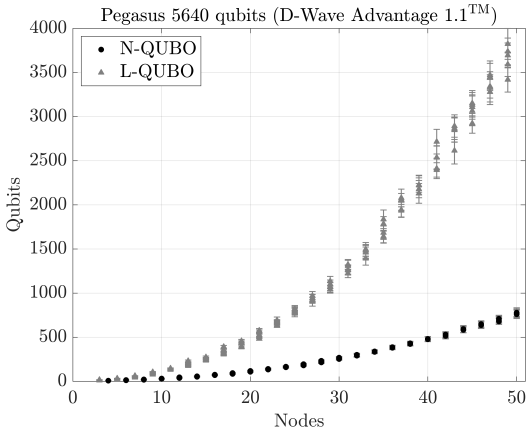


Figure 9.10: Embedding: $k = 2$, $\mathcal{G}(n, 0.50)$.

formulation is substantially better than the L-QUBO formulation. This is true not only in terms of the average qubits required to embed each QUBO, but the volatility of the number of qubits required to embed each QUBO. In Figure 9.5, in which sparse graphs (i.e., $p = 0.25$) are used for the case $k = 1$, both QUBO formulations can be embedded, for graphs with up to $n = 50$, in D-Wave's 2000Q™ processor. However, in Figure 9.7, where dense graphs (i.e., $p = 0.75$) are considered, now the L-QUBO can be embedded only for graphs with up to $n = 40$. From Figures 9.9 and 9.11 it is clear that as k and p increase, this trend of being able

to embed larger problems in terms of number of nodes n continues to be evidenced even more. Even using the more *powerful* Advantage 1.1^{TM} processor, Figure 9.12 shows that for sparse graphs (i.e., $p = 0.25$) and $k = 5$, the L-QUBO can only be embedded for graphs with up to $n = 40$, while it seems that the N-QUBO can be embedded for graphs with up to $n = 80$ (i.e., the double number of nodes).

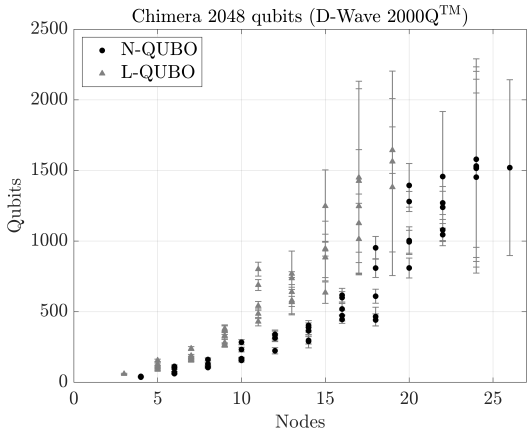


Figure 9.11: Embeding: $k = 5$, $\mathcal{G}(n, 0.25)$.

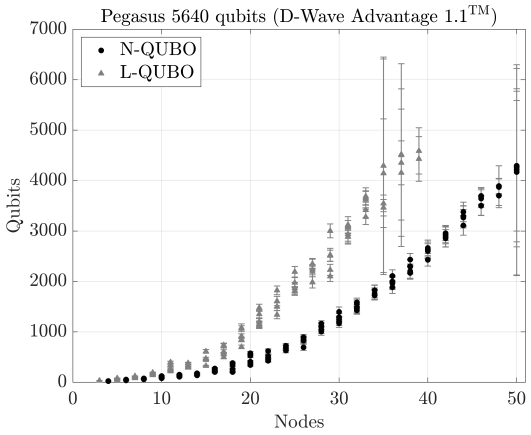


Figure 9.12: Embedding: $k = 5$, $\mathcal{G}(n, 0.25)$.

By pairwise comparing Figures 9.5, 9.7, 9.9, 9.11, versus Figures 9.6, 9.8, 9.10, 9.12, one can see the advantages of the Advantage 1.1^{TM} processor versus the 2000Q^{TM} processor. The effect of having a larger number of qubits clearly means that larger instances of the L-QUBO can be embedded in the Advantage 1.1^{TM} processor than in the 2000Q^{TM} processor. Also evident are the effects of the improved connectivity between qubits in the Advantage 1.1^{TM} processor. Namely, it is clear that the Advantage 1.1^{TM} processor is able to embed QUBO problems using a substantially lower number of qubits than in the 2000Q^{TM} processor. For example, from Figures 9.7 and 9.8, it takes about 800 qubits to embed the N-QUBO of graphs with 50 nodes in the 2000Q^{TM} processor, while it takes about half the number of qubits (about 400) to embed the N-QUBO of graphs with 50 nodes in the Advantage 1.1^{TM} processor. The pairwise comparison between Figures 9.5, 9.7, 9.9, 9.11, and Figures 9.6, 9.8,

9.10, 9.12, also shows how the added connectivity in the Advantage 1.1$^{\text{TM}}$ processor clearly lowers the volatility of the number of qubits required to embed a QUBO using D-Wave's embedding algorithm.

9.4.4 Time-To-Solution

We now finish our numerical tests by comparing (mirroring some of the tests in Sections 9.4.2.1 and 9.4.3) the *time-to-solution* (TTS) required, by both D-Wave's quantum annealer processors, when using the N-QUBO and L-QUBO reformulation of the MkCS, with penalty parameters $c_1 \in \{1,2,5\}, c_e \in \{1,2,5\}$, for values $k = 2$, on random graphs $\mathcal{G}(n,p)$ for $n \in [5,50]$, $p \in \{0.25, 0.75\}$ (similar results were obtained for $k \in \{1,5\}$, and $p = 0.50$ but are not presented for brevity). Besides benchmarking the N-QUBO versus the L-QUBO reformulation of the MkCS, these tests will be used to analyze the effect of the value of penalization constants in the TTS, and the effect in the TTS of using the more powerful Advantage 1.1$^{\text{TM}}$ processor.

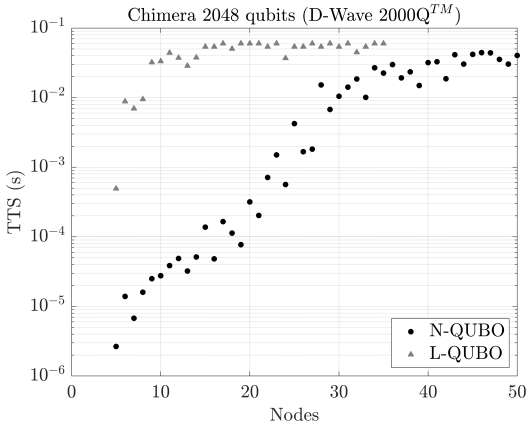


Figure 9.13: TTS: $k = 2$, $\mathcal{G}(n, 0.25)$, $c_1 = c_2 = 1$.

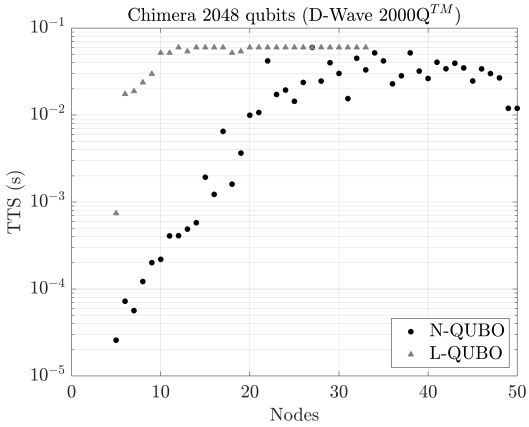


Figure 9.14: TTS: $k = 2$, $\mathcal{G}(n, 0.25)$, $c_5 = c_2 = 5$.

TTS is a common benchmark used to evaluate the performance of quantum annealers and is defined as the expected time required to find a ground state of the desired Hamiltonian

with a level of confidence α, which is set to 95% in our tests. Formally [see, e.g., 365],

$$\text{TTS} = t_{\text{run}} \frac{\ln(1-\alpha)}{\ln(1-p)}, \tag{9.31}$$

where t_{run}, fixed to 20μs in our tests, is the running time elapsed in a single run of the quantum annealer, and p is the probability of finding the ground state of the desired Hamiltonian. In our tests, p is estimated by running the quantum annealer 1000 times.

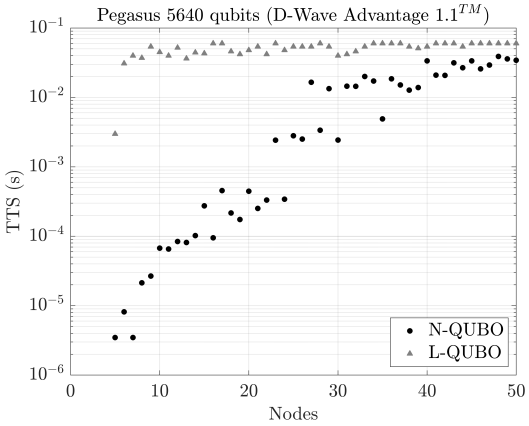


Figure 9.15: TTS: $k = 2$, $\mathcal{G}(n, 0.25)$, $c_1 = c_2 = 1$.

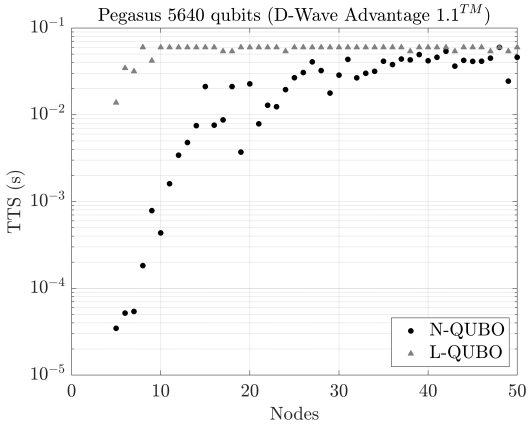


Figure 9.16: TTS: $k = 2$, $\mathcal{G}(n, 0.25)$, $c_5 = c_2 = 5$.

From Figures 9.13-9.20, it is clear that regardless of the quantum annealing processor, penalty parameters, or sparsity of the graph, the N-QUBO reformulation of the MkCS problem performs substantially better than the associated L-QUBO reformulation in terms of TTS. For example, note that from Figure 9.15 it follows that for sparse graphs (i.e., $p = 0.25$), the N-QUBO results in TTS values that are between three (3) orders of magnitude faster, for small graphs, and one (1) order of magnitude faster, for larger graphs, than the associate TTS values for the L-QUBO.

From Figure 9.17 it is clear that when considering non-sparse graphs (i.e., $p = 0.75$) in the 2000Q$^{\text{TM}}$ processor, the advantages of the N-QUBO over the L-QUBO in terms of TTS only increase. In particular, notice that while a t_{run} of 20μs is enough to find the optimal

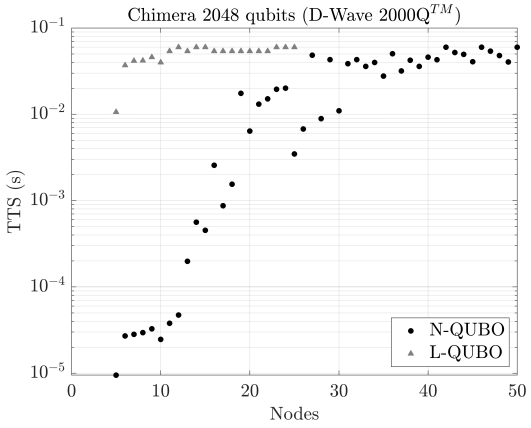


Figure 9.17: TTS: $k = 2$, $\mathcal{G}(n, 0.75)$, $c_1 = c_2 = 1$.

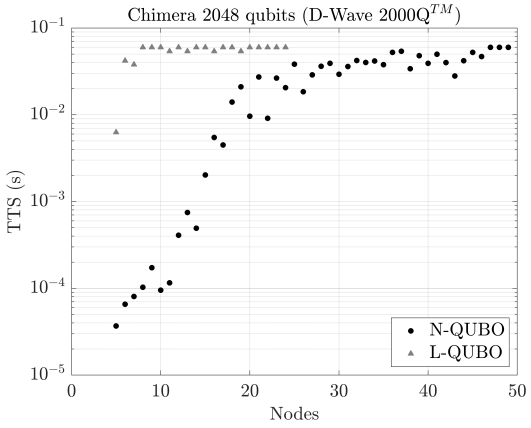


Figure 9.18: TTS: $k = 2$, $\mathcal{G}(n, 0.75)$, $c_5 = c_2 = 5$.

solution with some small probability for instances of the MkCS problem with underlying graphs of up to $n = 50$ nodes. In contrast, once the number of nodes of the underlying graph goes beyond $n = 25$, with the L-QUBO the quantum annealer is unable to find an optimal solution in all 1000 runs of 20μs.

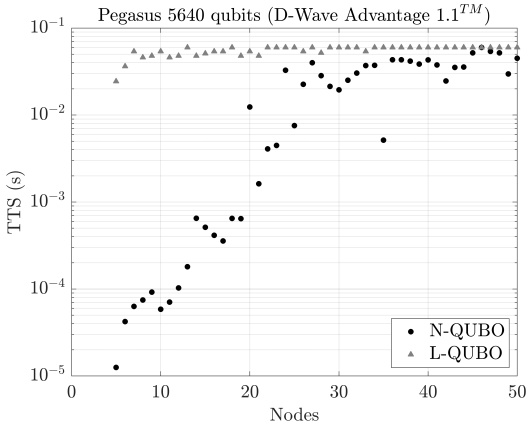


Figure 9.19: TTS: $k = 2$, $\mathcal{G}(n, 0.75)$, $c_1 = c_2 = 1$.

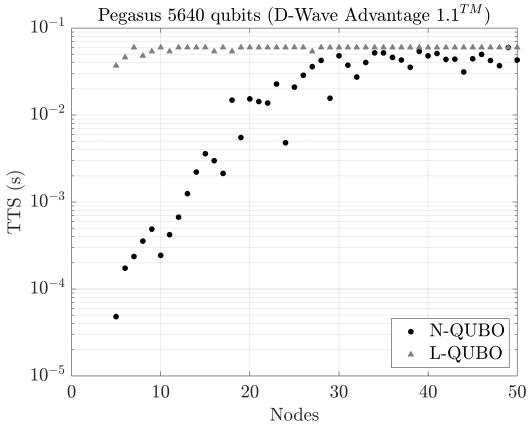


Figure 9.20: TTS: $k = 2$, $\mathcal{G}(n, 0.75)$, $c_5 = c_2 = 5$.

Not surprisingly, pairwise comparing Figures 9.13, 9.17, with Figures 9.15, 9.19, the advantages of the Advantage 1.1$^{\text{TM}}$ processor over the 2000Q$^{\text{TM}}$ processor in terms of TTS

are clear. For the L-QUBO the newer processor finds solutions for much larger instances of the MkCS problem. For the N-QUBO, the rate of increase of the TTS as the size of the MkCS problem increases is about one order of magnitude lower in the newer processor.

We can also use the results presented in this question to study whether the conclusions made in Section 9.4.2.1, particularly in Table 9.1, reflect on the actual TTS time when using a quantum annealer. Note that from Table 9.1 it was expected that for instances of the MkCS problem with $k = 2$ and underlying graphs $\mathcal{G}(n, 0.25)$, increasing the penalty constants from $c_1 = c_2 = 1$ to $c_1 = c_2 = 5$ would result in a faster convergence. However, by comparing Figures 9.13 and 9.14, as well as Figures 9.15 and 9.16, it follows that increasing the penalty constants in this way is actually counterproductive for both quantum annealing processors in terms of TTS (i.e., in Figures 9.14 and 9.16, the "slope" at which the TTS increases with the number of nodes is higher. Also, from Table 9.1 it was expected that for instances of the MkCS problem with $k = 2$ and underlying graphs $\mathcal{G}(n, 0.75)$, the increase in the penalty constants from $c_1 = c_2 = 1$ to $c_1 = c_2 = 5$ would have an even slightly higher benefit in terms of speed of convergence (compared with $\mathcal{G}(n, 0.25)$ graphs). However, by comparing Figures 9.17 and 9.18, as well as Figures 9.19 and 9.20, it follows that increasing the penalty constants in this way does not produce discernible improvements for the quantum annealing processors in terms of TTS. Most likely, this means that any theoretical advantages in terms of convergence obtained by increasing the value of penalty constants is off-set by the precision problems that using larger penalty parameters brings for the quantum annealing processors in practice. The fact that being able to use penalty parameters close to one (1) is beneficial for quantum annealers is discussed, for example, in [333]. This shows the importance of being able to fully characterize the range of penalty constants that result in a QUBO being a reformulation of a COPT problem.

9.5 Concluding remarks

In this chapter, we consider a particularly important combinatorial optimization (COPT) problem; namely, the maximum k-colorable subgraph (MkCS) problem, in which the aim is to find an induced k-colorable subgraph with maximum cardinality in a given graph. This problem arises in channel assignment in spectrum sharing networks (e.g., Wi-Fi or cellular), VLSI design, human genetic research, cybersecurity, cryptography, and scheduling. We derive two QUBO reformulations for the MkCS problem; a linear-based QUBO reformulation (Theorem 9.3.1) and a nonlinear-based QUBO reformulation (Theorem 9.3.2). Furthermore, we fully characterize the range of the penalty parameters that can be used in the QUBO reformulation. In the case of the linear-based QUBO reformulation, this analysis shows that Theorem 9.3.1 provides a better QUBO reformulation for the MkCS problem than the one that could be obtained using the QUBO reformulation techniques recently introduced by Lasserre [347]. In the case of the nonlinear-based QUBO reformulation, this analysis shows that Theorem 9.3.2 provides a better QUBO reformulation for the MkCS problem than the one that could be obtained using the well-known QUBO reformulation techniques introduced by Lucas [79]. Our proofs bring forward a fact that is overlooked in related articles. Namely, that when minimal penalty parameters are used in QUBO reformulations, the equivalence in terms of objective value between a problem and its associated QUBO reformulation does not necessarily mean that the optimal solution of the QUBO reformulation provides a feasible, optimal solution for the original problem. This is shown to be the case for the MkCS problem in Corollaries 9.3.1 and 9.3.2. Given that for $k = 1$ the MkCS problem is equivalent to the stable set problem, we show (see end of Section 9.3.2) that this issue applies to the well-known QUBO reformulation of the stable set problem (9.24). However, we show that this issue can be be simply addressed by using the greedy Algorithm 12 (for general instances of the MkCS problem).

We finish in Section 9.4 by illustrating the advantages of the nonlinear-based QUBO reformulation over the linear-based QUBO reformulation in terms of embedding requirements, convergence rate, and time-to-solution when the QUBO reformulations are used to solve the MkCS problem in a quantum annealing device. The experiments also illustrate the importance of having a full characterization of the penalty parameters that ensure the proposed QUBOs are indeed reformulations of the original problem. For example, we explore the potential theoretical and practical gains of using higher penalty parameters than the minimum ones required for the QUBO to become a reformulation of the MkCS problem. Our results show that although there are some theoretical benefits of using larger than minimal penalty parameters, they do not translate to a faster convergence to a solution of the problem on a quantum annealing computing device.

Our results contribute to recent literature that beyond obtaining QUBO reformulations of COPT problems such as the graph isomorphism problem as well as tree and cycle elimination problems, look for *improved* QUBO reformulations of these problems for NISQ devices [see, e.g., 327, 335, 337–339]. That is, QUBO reformulations that are tailored to be more efficiently used in NISQ devices.

Chapter 10

Formulating and Solving Routing Problems on Quantum Computers

10.1 Introduction

Routing problems encompass a wide range of problems in logistics and operations research. These problems are generally concerned with the optimal management of a fleet of vehicles, e.g., how should each vehicle be dispatched in order to satisfy some goal, while minimizing time or maximizing profit. There are many variants and specifications of the problem to certain settings [376]. Typical solution approaches to VRPTW on classical computing devices include mathematical formulations involving discrete variables. Consequently, classical methods tend to have worst-case solution times that scale exponentially with the number of decision variables (the 0-1 integer programming feasibility problem is NP-complete [377, §I.5, Prop. 6.6]).

For specific applications, tailored exact approaches, matheuristics, and metaheuristics need to be devised with the aim of obtaining solutions with good quality at a reasonable computational effort. For example, we are motivated by the maritime inventory routing problem (MIRP) [378], in which inventory levels of a product must be tracked. This is a

*Published as: Stuart Harwood, Claudio Gambella, Dimitar Trenev, Andrea Simonetto, David E Bernal, and Donny Greenberg. "Formulating and Solving Routing Problems on Quantum Computers". *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–17.

characteristic of commodity and bulk (e.g., oil, gas, iron ore, and grain) shipping, which in 2017 accounted for over 75of world seaborne trade measured in ton-miles [379, Fig. 1.4]. Recent work in [380] indicates the limits of some of these classical methods for the MIRP.

The ongoing evolution of quantum computing hardware and the recent advances of quantum algorithms for mathematical programming make decision-making for routing problems an avenue of research worthwhile to be explored on quantum devices. So far, quantum algorithms for mathematical optimization have often focused on quadratic unconstrained binary optimization problems (QUBOs) [326, 381], expressed in the form

$$\begin{aligned} &\min_{\mathbf{x}} \mathbf{x}^\top \mathbf{M} \mathbf{x} \\ &\text{s.t. } \mathbf{x} \in \{0,1\}^n, \end{aligned} \tag{10.1}$$

where $\mathbf{M}$ is a $n \times n$ real matrix. To achieve a representation on quantum devices, the QUBO can be transformed into an Ising model with Hamiltonian constituted as a summation of weighted tensor products of Z Pauli operators, i.e., $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, by mapping the binary variables x to spin variables $y \in \{-1, 1\}$, i.e., $x = \frac{y+1}{2}$. Equality constraints can also be added to the QUBO formulation by casting them as a quadratic penalization of the objective function [382, 383]. One of the goal of our work is then to introduce formulations of the routing problem that are amenable to emerging quantum hardware and quantum algorithms, which can provide fast heuristics for quadratic unconstrained binary optimization problems.

Gate-based quantum computers provide a particularly alluring setting to develop and propose new algorithms. Variational algorithms such as the quantum approximate optimization algorithm (QAOA) [85] and variational quantum eigensolver (VQE) [87] can be implemented on current gate-based devices. The idea behind quantum variational algorithms is to use quantum devices to efficiently sample from large parameterized distributions by executing shallow parameterized quantum circuits. In an optimization setting,

when solving QUBO problems, the quantum device is used to efficiently calculate a utility function (e.g., the expectation value or conditional value at risk [381]) of the cost Hamiltonian for a particular parameterized quantum state. A classical optimization algorithm is used to optimize the given utility function over the rotation parameters. Provided that the parameterized circuit, also called *ansatz*, represents states close to the Hamiltonian ground state, variational algorithms can obtain heuristic solutions of reasonable quality on current quantum devices. There are several reasons to motivate the adoption of quantum algorithms in solving QUBOs. First, variational algorithms like VQE and QAOA are not known to be classically simulatable [384]. Second, some encouraging results in terms of performance advantage have been documented for QAOA with respect to the classical Goemans-Williamson limit [353], for Grover Adaptive Search with respect to a classical unstructured search [385], and for quantum semidefinite programming relaxations for QUBO problems [386].

Decision making in practical optimization problems often involves modeling continuous variables and inequality constraints with mixed binary optimization (MBO) [387]. A recent contribution to solving MBO on current quantum devices in given by non-convex variants of the alternating direction method of multipliers (ADMM). Specifically, ADMM works by alternatively optimizing an augmented Lagrangian function over QUBO and continuous subproblems. The QUBO can be solved on quantum devices via quantum algorithms such as QAOA and VQE. Conditions of convergence to stationary points have been investigated, and ADMM has been shown to achieve satisfactory levels of solution quality [388].

An alternative framework to the work presented here is given by (quantum) annealing algorithms. Even if we do not discuss these approaches here, the interested reader can find studies in the works of e.g. [389–392] and [393, 394] for examples in vehicle routing. Examples of devices exploiting physical effects to solve the Ising problem include machines based

on nano-magnet arrays [395], electronic oscillators [396], optical parametric oscillators [397], and superconducting-based quantum annealers [398, 399]. Further, these physical effects have also inspired new algorithms like simulated bifurcation [400]. Heuristics like this and simulated annealing have also been accelerated via implementation on application-specific integrated circuits [401].

The contributions of this work are the following:

- We introduce mathematical optimization models for VRPTW suitable for state-of-art gate-based quantum algorithms.
- We investigate the modeling capabilities of the VRPTW formulations with respect to constraints and objectives arising in practical applications.
- We compare the VRPTW formulations from a quantum computing perspective, specifically with metrics to evaluate the difficulty in solving the underlying QUBOs.
- We test the formulations on simulated quantum devices, and draw insights on the current capabilities of quantum computing approaches for routing problems.

Given that the quantum algorithms proposed here for VRPTW are not bound to a specific quantum QUBO solver, our framework is flexible for future improvements. This is especially appealing, given the wealth of studies and results in this direction.

The remaining of the chapter is organized as follows. Section 10.2 introduces the formalization of the VRPTW that we consider, as well as how that form might be specialized when considering a MIRP via different mathematical formulations. The VRPTW formulations are inspired from those available in the operations research literature and introduced here to obtain QUBO representations suitable for quantum algorithms. Previous work on quantum-inspired algorithms applied to logistics problems have focused on the traveling salesman problem [326, 402, 403] or the related capacitated vehicle routing problem [393, 394]. Furthermore, we are not aware of any work that brings VRPTW approaches together and

compares them in the context of quantum algorithms (although see [404] for a comparison of different formulations in a classical setting).

In 10.3, we discuss how to cast and solve such formulations on quantum devices, via reformulations to QUBOs or a mixed-binary optimization (MBO) problem. For many of the formulations, this is through a reformulation as a QUBO, which enables the direct application of quantum algorithms like VQE and QAOA. However, one formulation involves continuous variables and inequality constraints; consequently a specialized decomposition method that results in QUBO subproblems is discussed in 10.3.3. The development of this method is instructive for handling inequality constraints and continuous variables that might arise in an extension of the other formulations. Section 10.4 compares the formulations along different dimensions. In particular, the strengths and weaknesses of the formulations in modeling VRPTW and MIRP are discussed in 10.4.2, while the difficulty for solving the formulations is evaluated in 10.4.3 and 10.4.4 via the number of variables, density of the QUBO matrix, and number of solutions. Section 10.4.5 includes numerical results obtained from solving a VRPTW instance with the outlined quantum algorithms. Finally, conclusions and remarks are drawn in 10.5.

Notation. Throughout this work we use the following notation. Sets are denoted with uppercase calligraphic letters (e.g., $\mathcal{A}$). Matrices are denoted with uppercase bold letters (e.g., $\mathbf{M}$), while vectors are lowercase bold letters (e.g., $\mathbf{v}$) For a vector $\mathbf{v}$, $\mathbf{Diag}(\mathbf{v})$ denotes a square matrix with $\mathbf{v}$ on its diagonal and zeros elsewhere.

10.2 Mathematical formulations for VRPTW

In this section, we describe the VRPTW at hand and specify the different mathematical formulations. An instance of VRPTW is formulated on a graph with nodes $N \cup \{d\}$ and

directed edges/arcs $\mathcal{A}$. Each node $i \in \mathcal{N}$ represents a customer, associated with which are a demand level q_i and time window $[a_i, b_i]$: q_i represents the amount of product that must be delivered ($q_i < 0$) or picked up ($q_i > 0$) after time a_i and before time b_i. The "depot" node d serves as departure and destination node for all vehicles $v \in \mathcal{V}$. In some situations, the depot node may be considered a physical node corresponding to, for instance, a warehouse, from which all vehicles start their routes fully loaded, and at which all vehicles must finish their routes. In the maritime setting, this is less common, and the depot typically serves as an artificial "source" or "sink" node, depending on the formulation. We note that the formulations discussed in this work could be seamlessly extended to the case in which starting and ending nodes for the vehicle routes are not coincident in the depot. Customers with product to pick up ($q_i > 0$) are considered suppliers; this feature is critical to handling maritime shipping problems. We allow a vehicle to arrive early (i.e., before a_i) and wait at a customer location, but not to arrive late (i.e., after b_i). Each customer is serviced exactly once (i.e., the demand cannot be split among different vehicles.). Each arc $(i, j) \in \mathcal{A}$ has an associated cost $c_{i,j}$ and travel time $t_{i,j}$. Typically, the cost for traveling from node i to j is the distance between the nodes. The vehicles are homogeneous (i.e., they have the same capacity, travel speed, cost to operate, etc.). Each vehicle has capacity (maximum load size) Q, and leaves the depot with an initial loading Q_0. When the depot corresponds to a warehouse we might assume $Q_0 = Q$, otherwise in the maritime setting (where some of the nodes are suppliers) $Q_0 = 0$. If it is ever needed, it is assumed that the depot has an infinite time window $[0, +\infty]$. The objective of the VRPTW is to minimize the total cost of transportation while servicing each customer in the given time windows.

In 10.2.1, we describe some key features of the MIRP, and how these manifest in the data of the VRPTW described above. A comprehensive overview of MIRP and a review of the literature can be found in [378]. In the following subsections, we describe four

different mathematical formulations of VRPTW: route-based, arc-based, sequence-based, and sequence-based with continuous time. Since our goal is to target QUBO subproblems, these formulations have primarily or exclusively binary variables and linear or quadratic equality constraints. The exception is the sequence-based formulation with continuous time in 10.2.5, which involves inequality constraints and continuous variables. These formulations have different levels of faithfulness in modeling VRPTW, and we discuss extensions that could make formulations more accurately capture the VRPTW setting.

10.2.1 Key features of MIRP

The MIRP is often characterized by travel times that are relatively long compared to the time windows of the nodes. These long travel times mean that fairly long time horizons must be considered in order to get the benefit of optimizing logistics. Further, there are often multiple supply points in addition to multiple demand points/customers. These supply points are often terminals producing a commodity like natural gas, and since they often have limited storage capacity, they must also be serviced, and inventory picked up, in certain time intervals. This is why demand levels q_i are signed. Another characteristic of maritime shipping, in particular in liquefied natural gas shipping, is that vessels/vehicles typically fully load at supply points and fully unload at demand points. Therefore a typical route of a vessel alternates between supply points and demand points. Combined with the assumption of a homogeneous fleet of vessels, this means that the capacity Q and the demand levels q_i are all equal in magnitude. Consequently, the constraints on vehicle capacity are less important. It is instead more important to enforce the alternating sequence of supply and demand points. This is easily achieved by restricting the arcs in $\mathcal{A}$ so that there are only arcs between a supply and demand node or vice-versa. Combined with the long travel times, narrow time windows, and long time horizons, the graph can be quite

sparse, as we can a priori remove arcs that would have the vessel arriving at the node after the time window ends.

10.2.2 Route-based formulation

In the route-based VRPTW formulation, the decisions to be made are whether routes are traveled or not. A route is a sequence of nodes $(i_1, i_2, \ldots, i_P)$ satisfying the following constraints (for some route-specific positive integer P). A route begins and ends at the depot: $i_1 = i_P = d$. Each segment is a valid arc: $(i_p, i_{p+1}) \in \mathcal{A}$, for all $1 \leq p \leq P-1$. For each feasible route, the running sum of the product delivered/picked up must be physically possible: $0 \leq Q_0 + \sum_{j=2}^{p} q_{i_j} \leq Q$, for all $p \leq P-1$ (i.e., at each stop the amount loaded on the vehicle must be non-negative and less than the vehicle's capacity). The arrival time at node i_p must be before the time window ends. If we let $T_{i_1} = 0$, then the effective arrival time at node i_{p+1} is given by $T_{i_{p+1}} = \max\left\{a_{i_{p+1}}, T_{i_p} + t_{i_p, i_{p+1}}\right\}$ for all $1 \leq p \leq P-1$. Then we require $T_{i_p} \leq b_{i_p}$ for all p.

We index the set of routes by the set $\mathcal{R}$. If route $r \in \mathcal{R}$ has node sequence $(i_1, i_2, \ldots, i_P)$, then it has cost $c_r = \sum_{p=1}^{P-1} c_{i_p, i_{p+1}}$. Finally, we define $\delta_{i,r}$ to be a constant with value 1 if route r visits customer $i \in \mathcal{N}$ (i.e., one of the nodes in the sequence defining the route is i), and zero otherwise.

We introduce variable x_r which has value 1 if a vehicle travels route r; otherwise it has value 0, and we stack the variables x_r as $\mathbf{x} = \{x_r\}_{r \in \mathcal{R}}$. As a mathematical program, the VRPTW becomes

$$\min_{\mathbf{x}} \sum_{r \in \mathcal{R}} c_r x_r \tag{10.2a}$$

$$\text{s.t.} \sum_{r \in \mathcal{R}} \delta_{i,r} x_r = 1, \forall i \in \mathcal{N}, \tag{10.2b}$$

$$x_r \in \{0, 1\}, \forall r \in \mathcal{R}. \tag{10.2c}$$

The equality constraint enforces the requirement that all customer nodes are visited by

exactly one vehicle. A constraint on the number of vehicles available can be enforced by making sure that the number of outgoing arcs from the depot d equals the number of available vehicles $|\mathcal{V}|$. If necessary, the only nodes directly connected to the depot can be thought of as "dummy" nodes, and essentially keep track of whether a vehicle is used or not.

Problem (10.2) may also be recognized as a set partitioning problem/exact covering problem, a classic problem in discrete optimization; see for instance [405], and [403, §4.1]. The number of routes $|\mathcal{R}|$ can, in general, be extremely large. If travel from any node to any other node is possible, then the number of routes is $|\mathcal{N}|!$, although the allowed arcs and constraints on a route described above will limit this. In classical computing approaches, this formulation is best handled by a column generation method (e.g., [405]). Here, we will consider $\mathcal{R}$ to be given. The instances that we will consider are either small enough that the routes can be exhaustively enumerated, or else we will use heuristics to generate a set of routes.

10.2.3 Arc-based formulation

We now consider a discrete-time arc-based formulation of the VRPTW. In this case, the range of time instants in which the vehicles can perform their routes is represented by discrete set $\mathcal{T}$. Each node's time window encloses at least one time point in $\mathcal{T}$: $\mathcal{T} \cap [a_i, b_i] \neq \emptyset$.

We introduce a variable $x_{i,s,j,t}$ which has value 1 if a vehicle travels from node i at time s to node j at time t; otherwise it has value 0.

As a mathematical program, the VRPTW becomes

$$\min_{\mathbf{x}} \sum_{i,s,j,t} c_{i,j} x_{i,s,j,t} \tag{10.3a}$$

$$\text{s.t.} \sum_{i,s,t} x_{i,s,j,t} = 1, \quad \forall j \in \mathcal{N}, \tag{10.3b}$$

$$\sum_{j,t} x_{j,t,i,s} = \sum_{j,t} x_{i,s,j,t}, \quad \forall (i,s) \in \mathcal{N} \times \mathcal{T} \tag{10.3c}$$

$$x_{i,s,j,t} = 0, \quad \forall (i,s,j,t) : t \notin [a_j, b_j], \tag{10.3d}$$

$$x_{i,s,j,t} = 0, \quad \forall (i,s,j,t) : s \notin [a_i, b_i], \tag{10.3e}$$

$$x_{i,s,j,t} = 0, \quad \forall (i,s,j,t) : (i,j) \notin \mathcal{A}, \tag{10.3f}$$

$$x_{i,s,j,t} = 0, \quad \forall (i,s,j,t) : s + t_{i,j} > t, \tag{10.3g}$$

$$x_{i,s,j,t} \in \{0,1\}, \quad \forall (i,s,j,t). \tag{10.3h}$$

Constraints (10.3b) ensure that each node (besides the depot d) is visited exactly once over all vehicles. Constraints (10.3c) ensure continuity of routes through the graph: if a vehicle enters node i at time s, then it must leave node i at that time. Note that we do not enforce this for the depot d, otherwise the vehicles could not get started. Constraints (10.3d) ensure that a vehicle arrives in a node during its time window. Constraints (10.3e) are implied by (10.3c) and (10.3b), but explicitly, a vehicle must leave a node during its time window. Constraints (10.3f) ensure that vehicles obey the allowed travel arcs. Constraints (10.3g) ensure that vehicles do not travel back in time. Note that we do not enforce the timing "exactly" (i.e., we allow $x_{i,s,j,t}$ with $s + t_{i,j}$ strictly less than t); this models the situation when a vehicle arrives early and waits.

Note that capacity constraints on the vehicles have not been enforced in this formulation. To do so, we could modify the formulation by adding an index $v \in \mathcal{V}$ to track the vehicles used. Constraint (10.3b) would be modified to make sure that each node is visited exactly once over all vehicles: $\sum_{v,i,s,t} x_{v,i,s,j,t} = 1$ for each $j \in \mathcal{N}$. Meanwhile Constraint (10.3c) would be modified to enforce continuity of routes for each vehicle: $\sum_{j,t} x_{v,j,t,i,s} = \sum_{j,t} x_{v,i,s,j,t}$, for each $(v,i,s) \in \mathcal{V} \times \mathcal{N} \times \mathcal{T}$. The other constraints are modified similarly. Then, capacity constraints

can be enforced with

$$0 \leq Q_0 + \sum_{i,s,j,u:u\leq t} q_j x_{v,i,s,j,u} \leq Q, \quad \forall (v,t) \in \mathcal{V} \times \mathcal{T}$$

(the cumulative loaded product up to time t must be nonnegative and less than the vehicle's capacity). This is similar to the constraint on routes from 10.2.2. This prevents an initially empty vehicle with capacity 2 from visiting three supply nodes with demand level 1 each, and then two nodes with demand level -1 each; while the sum of the loadings is 1, the capacity of the vehicle was exceeded at its third stop. However, as described in 10.2, the capacity constraints are less important for practically modeling MIRP. Hence, in the numerical tests, we leave these constraints out. Furthermore, we will see that this formulation will typically be the largest in terms of number of variables, and adding an index to track specific vehicles will only exacerbate that.

10.2.4 Sequence-based formulation

We describe a discrete sequence-based formulation for VRPTW. We introduce a variable $x_{v,p,i}$ which has value 1 if vehicle v visits node i at position p in its sequence; otherwise it has value 0. This makes a discretization of the time horizon unnecessary to model the VRPTW routes.

We assume that the depot is "absorbing": if a vehicle returns to the depot, it must remain there. Consequently, the arc (d,d) is in the arc set $\mathcal{A}$. Each vehicle can make a maximum number of stops P: this bound can be determined from capacity limitations, by application requirements, or simply by the number of customers. Since each vehicle starts and ends at the depot, $P-2$ is the maximum number of non-depot nodes that each vehicle can visit.

As a math program, the VRPTW becomes

$$\min_{\mathbf{x}} \sum_{v} \sum_{p<P} \sum_{(i,j)\in\mathcal{A}} c_{i,j} x_{v,p,i} x_{v,p+1,j} \tag{10.4a}$$

$$\text{s.t.} \sum_{v\in V} \sum_{p=1}^{P} x_{v,p,i} = 1, \quad \forall i \in \mathcal{N}, \tag{10.4b}$$

$$\sum_{i\in\mathcal{N}\cup\{d\}} x_{v,p,i} = 1, \quad \forall v \in \mathcal{V}, p \in \{1,\dots,P\}, \tag{10.4c}$$

$$x_{v,p,i} x_{v,p+1,j} = 0, \quad \forall v \in \mathcal{V}, p \in \{1,\dots,P-1\}, (i,j) \notin \mathcal{A}, \tag{10.4d}$$

$$x_{v,p,d} x_{v,p+1,j} = 0, \quad \forall v \in \mathcal{V}, p \in \{2,\dots,P-1\}, j \neq d, \tag{10.4e}$$

$$x_{v,1,d} = 1, \quad \forall v \in \mathcal{V}, \tag{10.4f}$$

$$x_{v,P,d} = 1, \quad \forall v \in \mathcal{V}, \tag{10.4g}$$

$$x_{v,1,i} = 0, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}, \tag{10.4h}$$

$$x_{v,P,i} = 0, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}, \tag{10.4i}$$

$$x_{v,2,j} = 0, \quad \forall v \in \mathcal{V}, (d,j) \notin \mathcal{A}, \tag{10.4j}$$

$$x_{v,P-1,j} = 0, \quad \forall v \in \mathcal{V}, (j,d) \notin \mathcal{A}, \tag{10.4k}$$

$$x_{v,p,i} \in \{0,1\}, \quad \forall (v,p,i)$$

Constraints (10.4b) ensure that each node is visited exactly once over all vehicles and sequence positions (besides the depot node d). Constraints (10.4c) ensure that each vehicle uses each position p in the sequence once. Constraints (10.4d) ensure that only allowed edges are traversed. Constraints (10.4e) ensure that once a vehicle returns to the depot, it remains there (i.e., it does not visit other nodes). Constraints (10.4f) and (10.4g) ensure that all vehicles start and end at the depot, respectively. Constraints (10.4h) and (10.4j) are merely a consequence of the assumption that all vehicles start at the depot. Note that constraints (10.4h) are implied by constraints (10.4c) and (10.4f), while constraints (10.4h) come from constraints (10.4d) and (10.4f). Finally, Constraints (10.4i) and (10.4k) are a consequence of the assumption that all vehicles end at the depot.

Capacity constraints on the vehicles are not directly enforced in this formulation either, in the general case. If all nodes $i \in \mathcal{N}$ have the same level of demand, capacity constraints can

be enforced by choosing P appropriately. Capacity constraints can also be handled more exactly by adding the constraints

$$0 \leq Q_0 + \sum_{r=2}^{p} \sum_{i} q_i x_{v,r,i} \leq Q, \quad \forall (v,p) \in \mathcal{V} \times \{1,\ldots,P-1\}.$$

However, as discussed before, capacity constraints are less important in the maritime setting, so this modeling feature is not included in the numerical testing.

Note that timing constraints have not been addressed explicitly in this formulation. In many problems, the time windows of the nodes are fairly narrow compared to the travel times between nodes and overall time horizon of the problem. Consequently, we can conservatively enforce the time windows by removing arcs from the graph that do not satisfy application-specific assumptions. Specifically, if the end of the time window of node i plus the travel time $t_{i,j}$ is greater than the end of the time window of node j, then that arc is removed. This corresponds to assuming that the vehicles always arrive at a node at the end of its time window. Thus, the set of arcs $\mathcal{A}$ used in this formulation may have to be different from the one used in the other formulations. For some problems, this may be too restrictive, which motivates the formulation of the following section.

10.2.5 Sequence-based formulation with continuous time

We here consider a sequencing-based formulation of the VRPTW with continuous variables to track the arrival time at each node. This makes it possible to seamlessly enforce time windows, unlike the formulation with binary decision only, described in 10.2.4.

As in the discrete sequence-based formulation, we define a variable $x_{v,p,i}$ which has value 1 if vehicle v visits node i at position p in its sequence; otherwise it has value 0. Further, we introduce a variable $s_i \in \mathbb{R}$ which equals the time when a vehicle arrives at node $i \neq d$. The arrival time of each vehicle v to the destination node d is the duration of route v, and is

associated with variable s_d^v. As a math program, the VRPTW becomes

$$\min_{\mathbf{x},\mathbf{s}} \sum_{v} \sum_{p<P} \sum_{(i,j)\in\mathcal{A}} c_{i,j} x_{v,p,i} x_{v,p+1,j} \tag{10.5a}$$

$$\text{s.t.} \sum_{v\in\mathcal{V}} \sum_{p=1}^{P} x_{v,p,i} = 1, \quad \forall i \in \mathcal{N}, \tag{10.5b}$$

$$\sum_{i\in\mathcal{N}\cup\{d\}} x_{v,p,i} = 1, \quad \forall v \in \mathcal{V}, p \in \{1,\dots,P\}, \tag{10.5c}$$

$$x_{v,p,i} x_{v,p+1,j} = 0, \quad \forall v \in \mathcal{V}, p \in \{1,\dots,P-1\}, (i,j) \notin \mathcal{A}, \tag{10.5d}$$

$$x_{v,p,d} x_{v,p+1,j} = 0, \quad \forall v \in \mathcal{V}, p \in \{2,\dots,P-1\}, j \neq d, \tag{10.5e}$$

$$x_{v,1,d} = 1, \quad \forall v \in \mathcal{V}, \tag{10.5f}$$

$$x_{v,P,d} = 1, \quad \forall v \in \mathcal{V}, \tag{10.5g}$$

$$x_{v,1,i} = 0, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}, \tag{10.5h}$$

$$x_{v,P,i} = 0, \quad \forall v \in \mathcal{V}, i \in \mathcal{N}, \tag{10.5i}$$

$$x_{v,2,j} = 0, \quad \forall v \in \mathcal{V}, (d,j) \notin \mathcal{A}, \tag{10.5j}$$

$$x_{v,P-1,j} = 0, \quad \forall v \in \mathcal{V}, (j,d) \notin \mathcal{A}, \tag{10.5k}$$

$$s_i \geq \sum_{v\in\mathcal{V}} \sum_{p<P} \sum_{j:(j,i)\in\mathcal{A}} (s_j + t_{j,i}) x_{v,p,j} x_{v,p+1,i}, \quad \forall i \in \mathcal{N}, \tag{10.5l}$$

$$s_d^v \geq \sum_{p<P} \sum_{j:(j,d)\in\mathcal{A}} (s_j + t_{j,d}) x_{v,p,j} x_{v,p+1,d}, \quad \forall v \in \mathcal{V}, \tag{10.5m}$$

$$a_i \leq s_i \leq b_i, \quad \forall i \in \mathcal{N}, \tag{10.5n}$$

$$x_{v,p,i} \in \{0,1\}, \quad \forall (v,p,i),$$

$$s_i \in \mathbb{R}, \quad \forall i \in \mathcal{N}.$$

All constraints pertaining only to the binary variables $x_{v,p,i}$ are shared with the sequence-based formulation discussed in 10.2.4. Constraints (10.5n) enforce the time window constraints explicitly. Constraints (10.5l) define the arrival times at nodes $i \in \mathcal{N}$ and can be justified as follows. Constraints (10.5b) and the fact that the x variables are binary, imply that for each i there is exactly one index (v^i, p^i) such that $x_{v^i,p^i,i} = 1$. Using this, constraints (10.5l) can be viewed as $s_i \geq \sum_{j:(j,i)\in\mathcal{A}} (s_j + t_{j,i}) x_{v^i,p^i-1,j} \quad \forall\, i \in \mathcal{N}$. By constraint (10.5c), there is exactly one index j' such that $x_{v^i,p^i-1,j'} = 1$. Consequently, constraints (10.5l) enforce that the arrival time at each node i must be greater than or equal to the arrival time at the previously visited

node plus the travel time. Similar reasoning can be done to derive constraints (10.5m): the only difference is that the arrival time to the depot depends on the vehicle v. Model 10.5 reads as a MBO problem, given the presence of both binary and continuous decision variables. The presence of constraints (10.5l) and (10.5m) makes the continuous relaxation of the problem non convex. As in the arc-based formulation, enforcing arrival times with an inequality rather than equality permits the possibility that a vehicle arrives early and waits.

Modeling the arrival times with continuous decision variables allows for an alternative formulation, with the aim of minimizing routes duration, without modifying the constraints set. This is achieved by switching objective (10.5a) with:

$$\min_{x,s} \sum_{v} s_d^v \tag{10.6}$$

Routes duration is a typical metric to evaluate vehicle routes in VRPTW.

10.3 Solving the routing problems on Quantum Computers

In this section, we discuss the reformulations needed to express the VRPTW formulations in a form suitable for quantum algorithms. In Sections 10.3.1 and 10.3.2, most of the formulations are cast into 10.1, which is a standard form for a QUBO, and enables the application of quantum algorithms such as VQE and QAOA. Due to the presence of continuous arrival times, the MBO formulation presented in 10.2.5 needs a different representation on quantum devices, which is discussed in 10.3.3.

10.3.1 Route and arc-based formulations as QUBO

Both problems (10.2) and (10.3) have the common form

$$\min_{\mathbf{x}} \left\{ \mathbf{c}^\top \mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \in \{0,1\}^n \right\}, \tag{10.7}$$

for some real vectors $\mathbf{b}$, $\mathbf{c}$ and real matrix $\mathbf{A}$. The challenge is to construct a matrix $\mathbf{M}$ so that the QUBO standard form in 10.1 is equivalent to the binary linear program (10.7). Specifically, we would like our matrix to encode the quadratic penalty (or energy) function

$$H : \mathbf{x} \mapsto \mathbf{c}^\top \mathbf{x} + \rho \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$$

for a real constant $\rho > 0$, to be determined. This transformation is an exact penalty reformulation of (10.7), and it is consistent with the general suggestion for binary linear programs from [403, §3], as well as the transformation specific to the set partitioning problem from [403, §4.1].

The term $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ expands to $\mathbf{x}^\top \mathbf{A}^\top \mathbf{A}\mathbf{x} - 2\mathbf{b}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{b}$. Consequently, to obtain a QUBO, we set $\mathbf{M} = \rho \mathbf{A}^\top \mathbf{A} + \rho \mathbf{Diag}(-2\mathbf{A}^\top \mathbf{b}) + \mathbf{Diag}(\mathbf{c})$. The constant term $\|\mathbf{b}\|^2$ must be saved to match the original objective value.

The main challenge is finding the right value of ρ so that the minimization of H is equivalent to (10.7). The general reasoning is as follows. Looking at the data of the constraints of problems (10.2) and (10.3), and considering that the variables are binary, the smallest value that the penalty terms can take for an infeasible solution is ρ (when $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 = 1$). Thus, ρ needs to be big enough to overwhelm any decrease in the original objective by moving to an infeasible point. Imagine flipping each variable from 0 to 1 or vice versa depending on the sign of c_i; we can upper bound that change in objective by $\sum_i |c_i|$. Thus, $\rho > \sum_i |c_i|$ suffices. For the route-based formulation (10.2), this is established in more detail in Appendix 10.B.

10.3.2 Sequence-based formulation as QUBO

The sequence-based formulation (10.4) differs from the route- and arc-based formulations since it has bilinear equality constraints. To devise the QUBO reformulation, express the

linear equality constraints (10.4b) and (10.4c) as $\mathbf{Ax} = \mathbf{b}$. Similarly, along with the bilinear equality constraints (10.4d) and (10.4e), these are turned into a penalty term

$$w : \mathbf{x} \mapsto \|\mathbf{Ax} - \mathbf{b}\|^2 + \sum_v \sum_{p<P} \sum_{(i,j)\notin\mathcal{A}} x_{v,p,i} x_{v,p+1,j} + \sum_v \sum_{p=2}^{P-1} \sum_{j:j\neq d} x_{v,p,d} x_{v,p+1,j}.$$

Consequently, an exact penalty reformulation of (10.4) is

$$\begin{aligned} \min_{\mathbf{x}} \ & \sum_v \sum_{p<P} \sum_{(i,j)\in\mathcal{A}} c_{i,j} x_{v,p,i} x_{v,p+1,j} + \rho w(\mathbf{x}) \\ \text{s.t.} \ & \text{(10.4f)}, \ \text{(10.4g)}, \\ & \text{(10.4h)}, \ \text{(10.4i)}, \\ & \text{(10.4j)}, \ \text{(10.4k)}, \\ & x_{v,p,i} \in \{0,1\}, \quad \forall (v,p,i). \end{aligned} \tag{10.8}$$

where the constraints (10.4f) to (10.4k) merely fix the values of certain variables.

For penalty parameter ρ sufficiently large, the solutions of (10.4) and (10.8) coincide. Looking at the specifics of the constraints and considering that the variables are binary, the smallest value that the penalty term $\rho w(x)$ can take for an infeasible solution is ρ (for $w(x) = 1$). Thus, ρ needs to be big enough to overwhelm any decrease in the original objective by moving to an infeasible point. Imagine flipping each bilinear term from 0 to 1 or vice versa depending on the sign of $c_{i,j}$; we can upper bound that change in objective by $\sum_v \sum_p \sum_{(i,j)\in\mathcal{A}} |c_{i,j}| = P|\mathcal{V}| \sum_{(i,j)\in\mathcal{A}} |c_{i,j}|$. Thus

$$\rho > P|\mathcal{V}| \sum_{(i,j)\in\mathcal{A}} |c_{i,j}|$$

suffices. As in 10.3.1, let $\widetilde{\mathbf{M}} = \rho \mathbf{A}^\top \mathbf{A} + \rho \mathbf{Diag}(-2\mathbf{A}^\top \mathbf{b})$. We can add the other bilinear terms from the objective and penalty to get an objective in the form $\mathbf{x}^\top \mathbf{Mx}$, as desired for a QUBO.

10.3.3 VRPTW via ADMM-based heuristic

In order to extend the range of mathematical formulations solvable on quantum near-term devices via variational-based approaches, [388] proposed a multi-block ADMM operator-splitting procedure. This iterative algorithm devises a decomposition for a specific class of MBOs into:

- a QUBO subproblem to be solved by a QUBO solver oracle (or, on near-term quantum devices by quantum algorithms such as VQE or QAOA).
- a convex constrained subproblem, which can be efficiently solved with classical optimization solvers [75].

The solutions obtained in each ADMM iteration are evaluated with a merit function, which evaluates the trade-offs between feasibility and optimality. The authors devised conditions for the MBO to converge via ADMM to stationary points of a soft-constrained reformulation of the problem. In particular, the set of MBO constraints needs to have a continuous convex relaxation, which is not the case for the sequence-based formulation with continuous time. This motivates the adoption of a strategy to reformulate the continuous subproblem with convex approximations. In particular, in our case the continuous subproblems are non-convex and they will be solved via a sequential convex approximation as follows.

Consider the cubic, non-convex constraints (10.5l)-(10.5m) of the sequence-based formulation with continuous times, namely:

$$s_i \geq \sum_{v \in \mathcal{V}} \sum_{p<P} \sum_{j:(j,i)\in\mathcal{A}} (s_j + t_{j,i}) x_{v,p,j} x_{v,p+1,i}, \quad \forall i \in \mathcal{N},$$

$$s_d^v \geq \sum_{p<P} \sum_{j:(j,d)\in\mathcal{A}} (s_j + t_{j,d}) x_{v,p,j} x_{v,p+1,d}, \quad \forall v \in \mathcal{V}$$

The idea is to deal with them in the continuous problem of the ADMM framework by using *sequential convex programming*, see [406, 407]. In particular, since we are splitting the problem onto the binary variables, we introduce continuous variable $u^{v,p,j} \in [0,1]$ for each

three-indexed binary variable $x^{v,p,j}$, and set $u^{v,p,j} = x^{v,p,j}$. Then, in the continuous problem of the ADMM, where the constraints (10.5l)-(10.5m) will become

$$s_i \geq \sum_{v \in \mathcal{V}} \sum_{p<P} \sum_{j:(j,i)\in\mathcal{A}} (s_j + t_{j,i}) u_{v,p,j} u_{v,p+1,i}, \quad \forall i \in \mathcal{N},$$

$$s_d^v \geq \sum_{p<P} \sum_{j:(j,d)\in\mathcal{A}} (s_j + t_{j,d}) u_{v,p,j} u_{v,p+1,d}, \quad \forall v \in \mathcal{V}$$

Such constraints can be compactly written as

$$\mathbf{g}(\mathbf{t}, \mathbf{u}) \leq 0. \tag{10.9}$$

Function $\mathbf{g}$ is not convex, since cubic in $(\mathbf{t}, \mathbf{u})$, but one can always use sequential convex programming approach to solve the continuous problem of the ADMM.

By using this strategy, the continuous problems of ADMM converge to a local stationary point and the overall ADMM strategy will remain an heuristic in general, but with the advantage that it limits the introduction of auxiliary binary decision variables in the QUBO subproblems and makes the solution of MBO on quantum devices a computationally tractable task.

10.4 Comparison of formulations

We are now ready to showcase the different formulations on simulated quantum hardware and compare their numerical properties and performance.

Firstly, we describe in 10.4.1 two examples that will serve as benchmarks to compare the mathematical formulations from a quantum computing perspective. We then report a summary of the qualitative modeling differences of the formulations in 10.4.2. Finally, quantitative comparisons are presented in Sections 10.4.3, 10.4.4, and 10.4.5 and are meant to: *(i)* demonstrate the inherent difficulty in finding routes with good solution quality at

a reasonable computational effort, even on classical devices; and *(ii)* report the solution metrics obtained with the quantum state-of-art solution approaches on quantum devices.

10.4.1 Example definitions

10.4.1.1 MIRP with varying time horizon

We define an example inspired by the MIRP setting with a varying time horizon. The example is a modification of instance `LR1_2_DR1_3_VC2_V6a` in Group 1 of the MIRPLIB library [378]. This example involves two supply ports and three demand ports; the objective is to minimize travel costs while visiting each port frequently enough to remove or replenish its inventory. One characteristic of this example is that we can vary the time horizon of the problem; thus we can effectively make the problem as large, in terms of number of nodes $|\mathcal{N}|$, as we want. See 10.A for its data, interpretation as a VRPTW, and the specific steps required to obtain the various formulations from 10.2.

10.4.1.2 Small VRPTW

We define a small 3-customer example, originally from [405], with data reported in Figure 10.1.

The instance has 11 valid routes, which define the feasibility set $\mathcal{R}$ for the route-based formulation (10.2): $(d,1,d)$, $(d,2,d)$, $(d,3,d)$, $(d,1,2,d)$, $(d,1,3,d)$, $(d,2,1,d)$, $(d,2,3,d)$, $(d,3,1,d)$, $(d,1,2,3,d)$, $(d,2,1,3,d)$, $(d,2,3,1,d)$. Routes $(d,1,2,3,d)$ and $(d,2,3,1,d)$ are optimal for the minimization of distance, with cost equal to five. In order to define the arc-based formulation (10.3), we need to define the discrete time points $\mathcal{T}$. We use the starting and ending points of the time windows $\{0,1,2,4,7\}$. This happens to exclude the optimal route/sequence $(d,2,3,1,d)$, which would require another time point $t \geq 8$ to allow the vehicle to return to the depot. For more complicated examples, it is often not viable to enumerate

exhaustively the set of time periods to consider. For the sequence-based formulation, we make the assumption mentioned in 10.2.4 that vehicles arrive at the end of a node's time window. This means that we have to prune the set of arcs; consequently the only valid arcs from node 1 are $(1,d)$, and similarly the only valid arcs from node 3 are $(3,d)$.

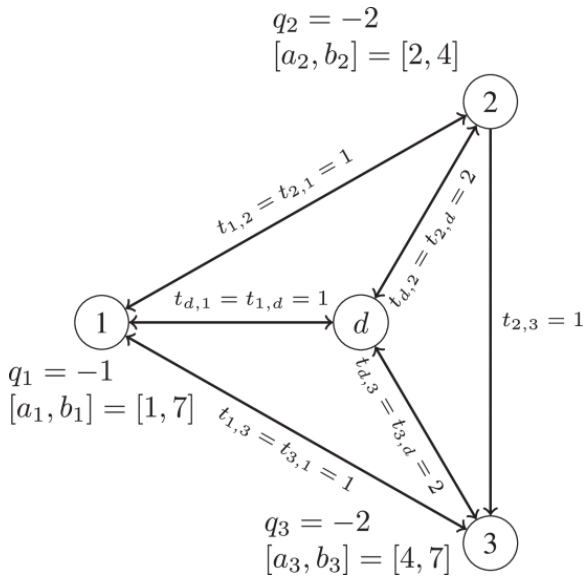


Figure 10.1: The small example VRPTW. Arc costs $c_{i,j}$ equal the travel time $t_{i,j}$. This is a nearly fully-connected graph; however, travel from node 3 to node 2 is not allowed because the time window of node 2 ends before the time window of node 3 begins. The vehicles leave depot d fully loaded at their capacity $Q = 6$. Recall the sign convention for q_i; q_i is negative for demand that must be delivered and depletes the product on a vehicle.

10.4.2 Qualitative comparisons

Each of the VRPTW formulations considered in 10.2 has different strengths and weaknesses in terms of modeling, which are summarized in Table 10.1. Most of these characteristics have been discussed previously. For instance, the sequence-based formulation models timing discretely, since it effectively assumes that vehicles arrive at the end of a time window. We remark that in MIRPs, the constraints on vehicle capacity are not necessarily a strict

requirement. This is because a demand node is often visited right after a supplier. Only the route-based formulation, as given, models the constraints on the vehicle capacities in full generality; the other formulations must be extended along the lines discussed in 10.2.3 and 10.2.4.

Another characteristic to consider is whether the formulations can handle inhomogeneous vehicles. The sequence-based and its continuous-time variant naturally can, since their variables are already indexed by vehicle v; transportation time or cost, for instance, could depend on the type of vehicle being used.

Finally, as mentioned in 10.2.5, the total duration of the routes taken is a common alternative objective used in VRPTW. The sequence-based formulation with continuous time can handle this, as can the route-based formulation by defining the cost of a route as the time that the vehicle arrives back at the depot. While the other formulations can take the arc costs $c_{i,j}$ to equal the arc travel time, this does not account for time spent waiting at a node.

Table 10.1: Qualitative characteristics of VRPTW formulations

Formulation	Route-based	Arc-based	Sequence-based	Sequence-based with continuous time
Models timing	Continuously	Discretely	Discretely	Continuously
Models capacity	Yes	No	No	No
Can handle inhomogeneous vehicles	No	No	Yes	Yes
Can handle routes duration objective	Yes	No	No	Yes

10.4.3 Solution-free metrics

As a first step in comparing the different VRPTW formulations, we examine the resulting QUBO problems, as introduced earlier in 10.3. The size and sparsity of the QUBO problem are the main resources needed to represent the problem on a quantum computer. Indeed, the number of qubits needed to represent the binary decision variables is equal to the dimension of the QUBO matrix $\mathbf{M}$, while the sparsity of the matrix serves as a proxy for the computational complexity of the problem. For example, when using a variational algorithm, such as VQE, the number of non-zero entries in the matrix corresponds to the number of different Pauli strings (or observables) used in the estimation of the Hamiltonian's expectation value. Another useful metric related to sparsity is the degree of connectivity of the QUBO problem, specifically the maximum number of non-zero entries per row (or column) of the matrix.

We now compare the formulations described in 10.2, for the MIRP example with varying size introduced in 10.4.1.1. As mentioned, we can increase the time horizon of the problem, and subsequently obtain QUBOs of increasing size and complexity for each formulation.

Table 10.2: QUBO problem size and connectivity for different time horizons of the MIRP example.

Formulation \ Time horizon		25	50	75	100	125	150	175	200
(10.2)	size	49	687	2315	3937	5839	7608	9517	11219
	connectivity	19	369	1283	2198	3142	4271	5462	6746
(10.4)	size	280	1106	2632	4676	7504	10647	14896	20048
	connectivity	56	127	196	259	328	388	460	527
(10.3)	size	582	2194	5372	9584	15473	21302	30903	37858
	connectivity	62	143	276	379	495	559	681	773

Table 10.2 shows the growth with the time horizon of the size and the degree of con-

nectivity of QUBOs for the route-based, sequence-based, and arc-based formulations. As discussed in 10.2.5, the sequence-based formulation with continuous time is not directly expressed with a QUBO, but rather it is an MBO. However, the metrics for the discrete sequence-based formulation in 10.2 are highly informative for the effort required to solve MBO on quantum devices via ADMM. In particular, the QUBO subproblems of the ADMM heuristic for MBO have the same size of the sequence-based QUBO: this is because MBO and the sequence-based formulation (10.4) share the same combinatorial structure (i.e., same binary variables and constraints involving binary variables). The connectivity of the first QUBO subproblem of ADMM in MBO differs from that of the sequence-based QUBO by a constant term.

As expected, all formulations grow steadily with the time horizon, with the route-based formulation generating the smallest, but also densest, QUBO problems. This is due to the heuristics in the route-generation pre-processing, ensuring the construction only of viable routes which do not violate any of the problem (time-window) constraints. This suggests that when one is restricted in the number of qubits one can use, the route-based formulation may prove advantageous over the sequence- and arc-based ones, as its size is controlled by the classically-computed route-generation process. On the other hand, if the number of circuit executions or qubit connectivity is an issue, using the arc- or sequence-based formulation may be preferable.

Finally, 10.2 presents the size, number of observables, and density metrics for the sequence-based and arc-based formulations. For reference we also plot quadratic (i.e., $c_2 T_H^2$) and cubic (i.e., $c_3 T_H^3$) functions of the time horizon T_H. It is evident from the plots that the size of the QUBO grows as $O(T_H^2)$, and the computational complexity (i.e., number of observables or non-zero entries in the QUBO matrix) as $O(T_H^3)$. While in this particular case, the arc-based formulation is approximately twice as big in size, this distinction is not

enough to choose one formulation over the other, and the two formulations can be thought of equivalent (with respect to the metrics in question) for all practical purposes.

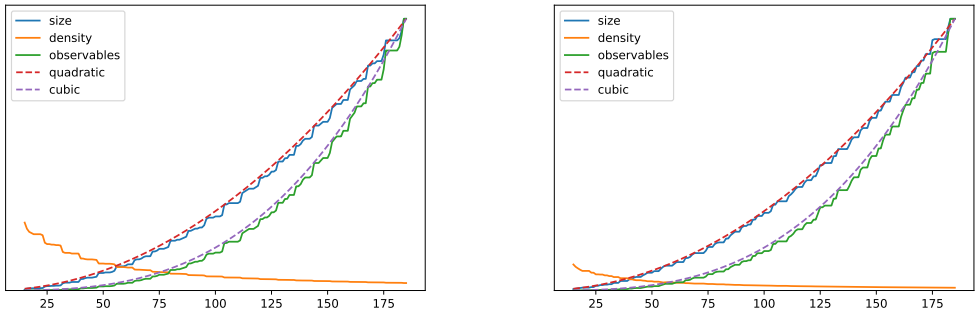


Figure 10.2: Metrics for the sequence-based (left) and arc-based (right) formulations plotted against quadratic and cubic curves (scaled by a constant factor). It is clear that the size (i.e., number of decision variables or qubits) grows as $O(T_H^2)$, while the observables (i.e., non-zero correlations or Pauli strings) grows as $O(T_H^3)$.

10.4.4 Solution-based metrics

The comparison between the different formulations can also be made in terms of the quality and quantity of feasible solutions. This is meant to describe the landscape of solutions on which the QUBO quantum solvers conduct their search for ground states.

In this section, we obtain such statistics for the route-, arc-, and sequence-based formulations of the MIRP example with varying sizes described in 10.4.1.1 using the commercial branch-and-bound solver CPLEX [52]. In particular, we have enumerated feasible solutions via CPLEX's populating routines and measured their quality via their fractional difference with the optimal solution, commonly known as optimality *gap*, computed as:

$$gap = \frac{sol - opt}{opt + \epsilon}\% \tag{10.10}$$

where *sol* denotes the objective value of the solution, *opt* the optimal solution value, and ϵ

denotes a tolerance value to avoid undefined division in case that $opt = 0$. For these results, we consider $\epsilon = 10^{-10}$.

We consider the MIRP example with time horizons 15, 20, and 25; the resulting problems are (for the most part) small enough to permit enumeration of all possible feasible solutions. We limit the characterization of the feasible solutions up to 50% optimality gap. In the unconstrained reformulation of the problem, a feasible solution beyond certain optimality gap would be indistinguishable, in terms of the objective function, from infeasible solutions of the original problem.

The results in Figure 10.3 show that the arc-based formulation provides many more feasible solutions to the same problem, followed by the sequence-based formulation. Even for these small instances, we were unable to compute all feasible solutions for the arc-formulation. The number of feasible solutions with the arc-based formulation for the smallest instance, with a time horizon of 15, exceeded 10 million and required more than 100Gb of RAM and several hours of computation. Consider that the number of binary variables for these instances makes a complete enumeration impossible in a practical amount of time.

We observe that the route- and sequence-based formulations with the shortest time horizon report all solutions being equivalent in terms of the objective function to the global optimal solution. For the larger time horizons, there is more diversity in the feasible solutions in terms of the objective values. As Table 10.2 reports, the arc- and sequence-based formulations require a larger number of variables leading to a considerably larger number of feasible solutions compared to the route-based formulation, although many of which do not have good objective function values.

Variational algorithms, such as VQE and QAOA, are not guaranteed to converge to globally optimal solutions and might get stuck in locally optimal solutions [326]. For

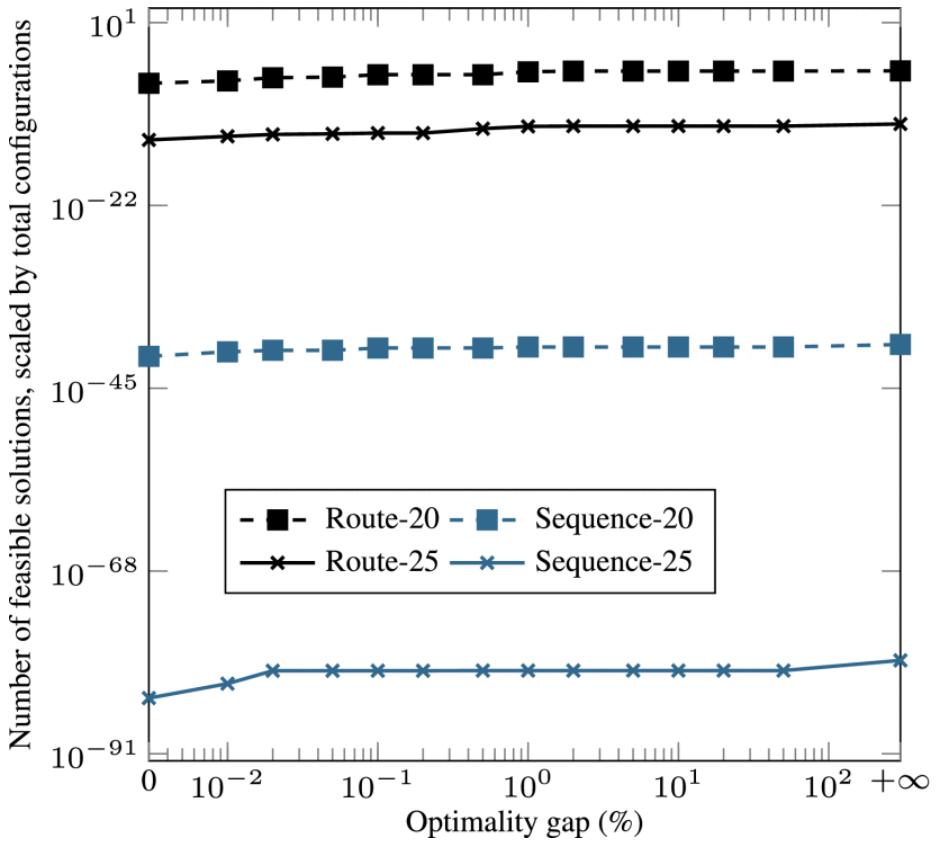


Figure 10.3: Cumulative number of feasible solutions (scaled by total number of configurations) within a certain percentage gap of the optimal solution for the route- and sequence-based formulations of the MIRP example with different time horizons. All optimal solutions are captured by an optimality gap equal to 0, while all feasible solutions are captured by an optimality gap equal to $+\infty$. Solutions with gap of $< 0.01\%$ are considered as optimal.

instance, when using algorithms prone to converge to locally optimal solutions, a preferable formulation is one where the local optimal solutions have small gaps to global optimality.

For the case with time-horizon equal to 25, only two solutions of the sequence-based formulation are optimal, while 5536 are within 0.02% of optimality, 5792 are within 0.5% of optimality, and a total of 110882 exist. A different situation happens for the route-based formulation, where the total number of feasible solutions is 1008, of which 544 are within 1% optimality gap, and 10 are found to be optimal. A similar situation happens with the time-

horizon equal to 20, where a larger fraction of the feasible solutions of the sequence-based formulation is beyond the 50% optimality gap compared to the route-based formulation. This is better observed in Figure 10.4 where the fraction of the feasible solutions found is plotted with respect to its optimality gap.

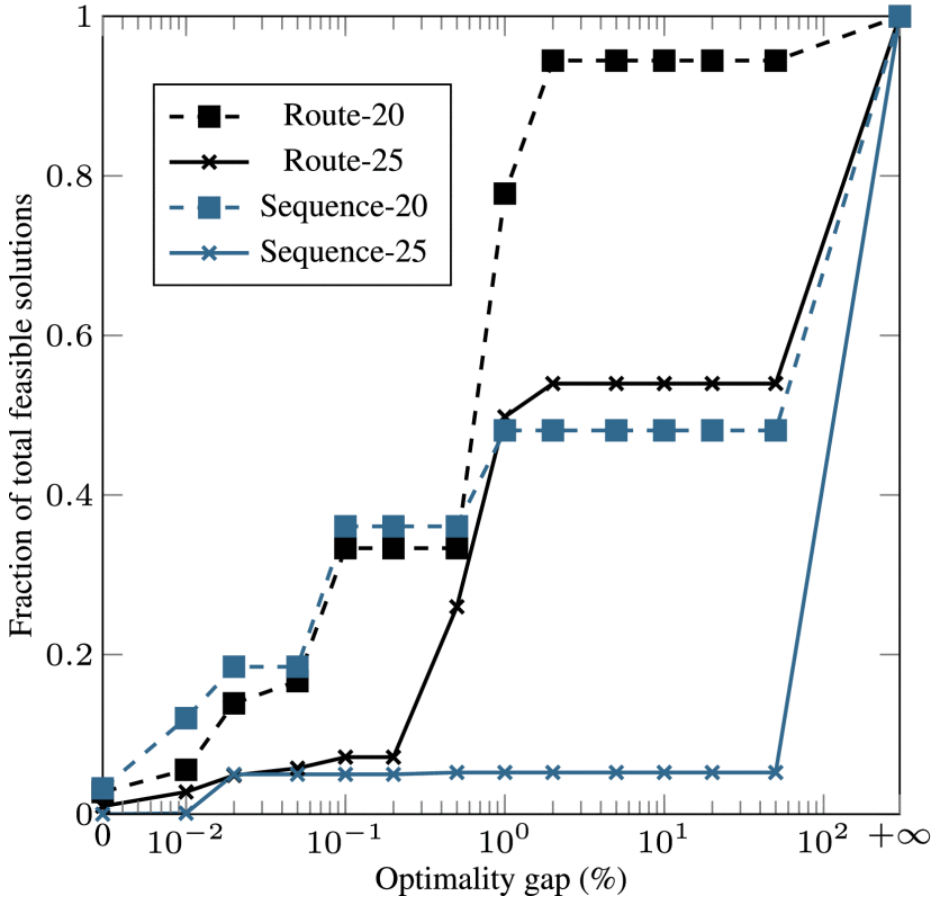


Figure 10.4: Fraction of feasible solutions within a certain percentage gap of the optimal solution for the route- and sequence-based formulations of the MIRP example with different time horizons. The plot suggests that the route-based formulation tends to have a higher density of near-optimal feasible solutions. Feasible solutions are reported as (feas) and solutions with gap of $< 0.01\%$ are reported as optimal (opt).

10.4.5 Numerical experiments

In this section, we solve the small VRPTW example from 10.4.1.2 using the IBM Q quantum simulators and devices accessed through the open-source programming framework Qiskit [408, 409]. We focus the discussion on the sequence-based formulations since their size is small enough to analyze classically, yet large enough to provide some insights of the algorithm performance. This VRPTW instance requires 16 qubits to be represented as a QUBO.

10.4.5.1 Sequence-based formulation

For the discrete sequence-based formulation (10.4), we consider both the variational quantum eigensolver (VQE) with a standard hardware-efficient ansatz (RY) based on single-qubit rotations and two-qubit entangling gates [410] (see 10.5 for a representation of the variational form), as well as the quantum approximate optimization algorithm (QAOA), where the parameterization of the circuit is constructed by the alternate application of the cost Hamiltonian and a mixing operator [411, 412]. For the classical optimization part, we consider two well-known gradient-free optimization algorithms: the simultaneous perturbation stochastic approximation (SPSA) optimizer [413], and constrained optimization by linear approximation (COBYLA) [414]. In the cases of both SPSA and COBYLA, the maximum number of function evaluations is set to 1000. We use the different ansatz/optimizer combinations (e.g., RY/SPSA, etc.), subsequently referred to as QUBO solvers, to directly solve the QUBO reformulation.

The sequence-based formulation for the small VRPTW example results in 16 decision variables. Out of the $2^{16} = 65536$ possible configurations, 2 are optimal, and another 2 are feasible (i.e., satisfy all constraints) but sub-optimal. We report our experimental results using two related metrics. The first is the single-shot probability of success, p_s, of an

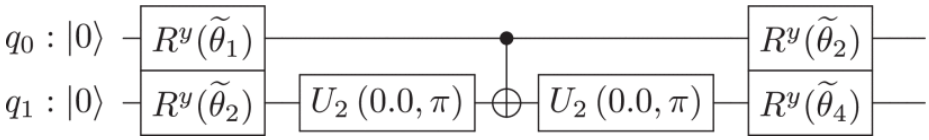


Figure 10.5: Representation of the RY ansatz with two qubits and an entanglement depth of one.

experiment, which is defined as the probability to obtain an optimal configuration when taking a single measurement of the quantum state obtained after running a given solver. Because simulating the full state-vector output of a quantum circuit is expensive (even with 16 qubits), this probability is approximated by taking 8192 independent measurement simulations and counting the number of optimal outputs. The second metric is the global probability of success, P_s, of a QUBO solver, which is used to account for the innate randomness in the underlying algorithm (e.g., the initial value for the parameters, when running the classical optimizer), and is approximated by running that given solver 250 times and counting the number of times we estimate a non-zero single-shot probability of success.

Table 10.3: Success probabilities of different solvers.

Solver \ Metric	average p_s	P_s	average p_f	P_f
RY/SPSA	2%	$\frac{60}{250} = 24\%$	4%	$\frac{128}{250} = 51.2\%$
RY/Cobyla	12%	$\frac{33}{250} = 13.2\%$	25%	$\frac{68}{250} = 27.2\%$
QAOA/SPSA	0.003%	$\frac{62}{250} = 24.8\%$	0.06%	$\frac{90}{250} = 32\%$
QAOA/Cobyla	0.02%	$\frac{108}{250} = 43.2\%$	0.04%	$\frac{143}{250} = 57.2\%$

Table 10.3 reports these two metrics, as well as the values p_f and P_f, which are defined similarly, but only require sampling of feasible, and possibly sub-optimal, configurations (since we have twice as many feasible as we have optimal configurations, $p_f \approx 2p_s$). We

should note that these results are obtained for the shallowest possible quantum circuits (depth 1 for the RY ansatz and $p = 1$ for QAOA), which means we are optimizing functions of 32 real-valued parameters in the case of VQE/RY, and 2 real-valued parameters in the case of QAOA.

Interestingly, while RY-based solvers produce the highest single-shot probability of success, their global success probability is lower than that of the QAOA-based solvers. This can be explained with the fact that while the RY ansatz can really narrow down on the optimal state (in the case of RY/COBYLA, the probability of success averaged over the runs where it was estimated to be non-zero was over 90%). Consequently, the optimizer has trouble performing the minimization in a 32-dimensional space, and often gets stuck in regions corresponding to states that have almost no overlap with an optimal configuration.

To better understand how the probability of sampling an optimal configuration depends on the number of shots, one can rely on the estimate

$$p(N) \approx P_s\left(1 - (1 - \frac{p_s}{P_s})^N\right).$$

In the above formula, N is the number of shots, and $\frac{p_s}{P_s}$ is the average single-shot probability of success, when this average is taken only over the runs where $P_s \neq 0$ (i.e., when the optimizer has succeeded to converge to a state that has a sufficiently large overlap with an optimal configuration).

The $p(N)$ computed for the different solvers are presented in 10.6. Also included is the corresponding function for uniform random sampling (in this case, $P_s = 1$, since we always have a non-zero probability of sampling an optimal configuration $p_s = \frac{2}{2^{16}} \approx 0.003\%$). Again, it is evident that the "best" choice of solver depends on the number of samples one is willing and able to take. For a single sample, the best performing solver is RY/COBYLA, which is quickly overtaken by RY/SPSA. When the number of sampling shots increases, the fact that of the QAOA circuit is not "flexible" enough to represent a state that overlaps with only

a few of the possible configurations becomes a strength, and the QAOA/COBYLA solver performs better.

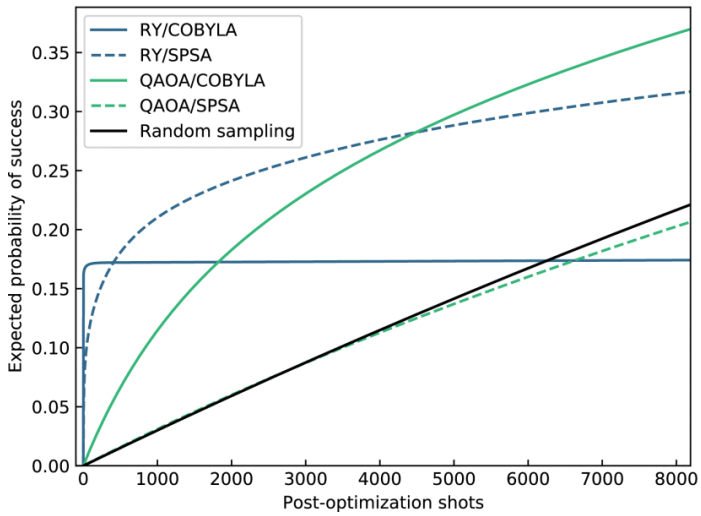


Figure 10.6: Expected probability of successfully measuring an optimal solution as a function of circuit evaluations or shots. These are the number of evaluations given an optimized circuit, that is, after the optimization phase.

10.4.5.2 Sequence-based formulation with continuous time

The sequence-based formulation with continuous time needs 16 qubits for the QUBO subproblems to be solved via the ADMM heuristics. This is because the size of the QUBOs is the same as the QUBO reformulation of the discrete sequence-based model (10.4). The numerical results reported here are referred to the 2-block implementation of ADMM with hyperparameters $\rho = 1001$ and $\beta = 1000$: this ensures ADMM to terminate in a finite number of iterations, in case the continuous subproblem is convex (see [388] for a detailed discussion on the ADMM implementation and convergence properties). The QUBO subproblems of the ADMM heuristic are solved with VQE and QAOA as quantum solvers, and COBYLA as classical optimizer. The continuous subproblems are solved via the sequential quadratic programming algorithm described in 10.3.3. Preliminary computations conducted with

COBYLA as continuous ADMM solver showed that the linear approximations performed therein resulted in a considerable increase of the computational time.

We have here tested both the minimum-distance and minimum-time VRPTW formulations described in 10.2.5. We have evaluated the MBO solutions via the following metrics:

- Probability of success, P_s, of ADMM. This is expressed as the percentage of ADMM runs that deliver an optimal solution for the problem. Note that the source of randomness of ADMM lies in the QUBO, which is solved via quantum algorithms.
- Probability of feasible solutions, P_f, found by ADMM. The two probabilities of success P_s and P_f are analogous to those introduced to evaluate QUBO solvers in 10.4.5.1.
- Number of iterations I for ADMM convergence.
- Percentage q_{opt} of QUBOs solved to optimality by the QUBO solver.

All metrics are obtained as average results over 3 runs of ADMM on the `QasmSimulator` backend available in Qiskit Aer. The two QUBO solvers tested for ADMM are VQE and QAOA with COBYLA as internal classical solver. Preliminary computations with the SPSA optimizer resulted in extremely slow ADMM convergence. For VQE, we chose the same ansatz tested for the QUBO formulations of 10.4.5.1, since the underlying combinatorial structure of the optimization problem is very similar.

The min-distance formulation turns out to be very efficiently solved by ADMM. As can be observed in 10.4, all three ADMM runs deliver a feasible and optimal route. Choosing VQE as QUBO solver results in a quicker convergence. Solving QUBOs in an exact fashion is not a guarantee for boosting ADMM convergence. Rather, a certain degree of inexactness in ADMM is beneficial for the overall solution quality [388, 415]. This point is particularly important since current computations on quantum devices are inevitably affected by errors and noise.

The min-time formulation is solved to optimality by ADMM with VQE as QUBO solver.

Table 10.4: Metrics obtained for the min-distance MBO via ADMM.

QUBO solver	P_s	P_f	I	q_{opt}
QAOA	100.00%	100.00%	54	19.38%
VQE	100.00%	100.00%	32	10.42%

For QAOA, 10.5 reports the metric obtained with 2 choices of circuit depth $p = 1, 3$. A larger depth for the variational form helps ADMM converge in fewer iterations, and the success rates P_s and P_f of ADMM with QAOA is $1/3$ in both cases. Specifically, in one simulation ADMM finds the feasible and optimal route with traveled time 6 displayed in 10.7(a).

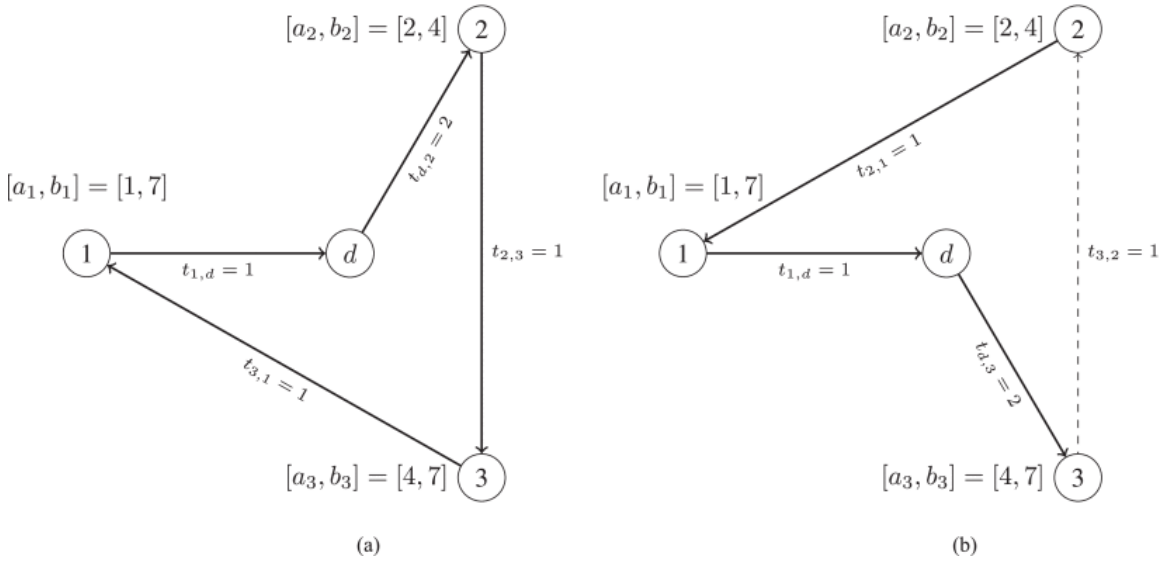


Figure 10.7: (a) and (b) Solutions found by the ADMM runs. Optimal solution for minimization of route duration is displayed on the left. The vehicle visits nodes 2 and 3 and waits one unit of time at node 3. At time 4, the vehicle leaves node 3 for visiting the remaining customer at node 1 and then reaches the depot at time 6. Infeasible solution for minimization of route duration is reported on the right. The vehicle visits node 3 and heads to node 2 at time 4. The infeasible arc for node 2 time windows is displayed with a dashed line. Finally, the vehicle visits the customer at node 1 and reaches the depot at time 7.

The other two ADMM runs with the QAOA QUBO solver deliver the route shown in 10.7(b), with infeasible arc connecting node 3 and 2. Future research work could explore the impact of choosing different mixing Hamiltonian function into the QAOA algorithm, to

aid the search for feasible routes.

QUBO solver	P_s	P_f	I	q_{opt}
QAOA, $p = 1$	33.33%	33.33%	50	18.87%
QAOA, $p = 3$	33.33%	33.33%	44	26.46%
VQE	100.00%	100.00%	31	17.57%

Table 10.5: Metrics obtained for the min-time MBO via ADMM.

For both VQE and QAOA as QUBO solvers, ADMM exhibits convergence in a finite number of iterations, thanks to the sequential convex programming solver.

10.5 Conclusions

Size, sparsity, connectivity, model faithfulness, and difficulty to find optimal, or even feasible, solutions are all characteristics of a vehicle routing problem that depend heavily on the specific instances of interest and their mathematical formulation. Here, we have provided insights into these characteristics for the VRPTW. In particular, we have investigated these characteristics for four mathematical formulations amenable to being solved on current quantum hardware. Motivated by the MIRP, we have assessed metrics that indicate hardware requirements of tens of thousands of logical qubits to solve real-world business problems. As indicated in [6], this approximate threshold is also where it can be difficult to obtain good-quality solutions with reasonable computational effort on modern classical hardware.

We are tempted to ask the question: "Which formulation is best?" Naturally, there are tradeoffs, but each offers lessons to be learned. Since the number of qubits available on hardware will be limited in the near term, preprocessing heuristics, like those used in the route-based formulation, are needed to reduce the size of the problem. The route generation heuristics help make the route-based formulation the most compact one. Good

preprocessing also seems to influence the higher density of near-optimal solutions of the route-based formulation, observed in Fig. 4. However, the route-based formulation has unfavorable sparsity and connectivity characteristics, which are also important metrics to consider for near-term hardware. This suggests using the route generation heuristics to reduce the size of the arc- or sequence-based formulations. It would be interesting to see if this helps reshape the solution landscape for these formulations as well.

The "best" QUBO solver depends on the number of samples one is willing to take from the optimized circuit. VQE and QAOA represent two extremes of the tradeoff between having a flexible ansatz that can represent the optimal solution and having a small enough number of parameters to facilitate the convergence of the classical optimizer to a high-quality solution. In our experiments, with a large enough number of samples, the QAOA/COBYLA solver had the highest probability of sampling an optimal configuration. Meanwhile, when the number of samples was limited, using VQE yielded a larger success probability.

For the sequence-based formulation with continuous time, the simulations with the sequential convex programming solver have shown the practical convergence of the ADMM heuristic to solutions with a large probability to be feasible and optimal. Furthermore, it has been shown that solving the QUBO subproblems on quantum devices at proven optimality is not a necessary condition to ensure the quality of the ADMM solutions. A certain degree of inexactness is tolerated by the ADMM algorithm, and this is a promising feature to handle the inherent noise affecting the quantum algorithms on real devices.

Future work could extend the sequence-based with continuous time formulation to handle the vehicle capacity constraints and evaluate the quality of the ADMM solutions obtained.

10.A Construction of the MIRP example

The example MIRP introduced in 10.4.1.1 is a modification of instance LR1_2_DR1_3_VC2_V6a in Group 1 of the MIRPLIB library [378]. Here, we describe its data, interpretation as a VRPTW, and any specific steps required to obtain the various formulations from 10.2.

10.A.1 Description of the example MIRP

In this example, we have two supply ports S1 and S2 and three demand ports D1, D2, D3. Each supply port produces a good but has some limited amount of storage space for it. Similarly, each demand port consumes this good, and has limited inventory of it. Different ports have different port fees as well. The port data for this example is reported in Table 10.6. The port fees have units of e.g. dollars, while initial inventory and capacity have units of volume, and production and consumption rate have units of volume per time. The distances between the ports are given in Table 10.7.

Table 10.6: Port data for MIRP example

Port	S1	S2	D1	D2	D3
Initial Inventory, I_j^0	220	270	221	215	175
Storage Capacity, I_j^C	376	420	374	403	300
Production (Consumption) rate, I_j^Δ	47	42	-34	-31	-25
Port fee	30	85	60	82	94

A fleet of vessels is available; these vessels all have the same capacity of $Q = 300$ (volume units) and speed of 665 (distance per time, e.g. km/day). The travel cost per unit distance for these vessels is 0.09 (e.g. dollars per km). We assume that the time horizon of interest begins at $t = 0$. We allow the end of the time horizon T_H to vary.

Table 10.7: Distances between ports for MIRP example (distance are symmetric)

	S1	S2	D1	D2	D3
S1	0	212.34	5305.34	5484.21	5459.31
S2		0	5496.06	5674.36	5655.55
D1			0	181.69	380.30
D2				0	386.66
D3					0

10.A.2 Interpretation as a VRPTW

The MIRP from 10.A.1 makes no mention of time windows or demand levels. In order to put this problem in the formalism of a VRPTW, we need to convert each port into a series of nodes with a fixed demand level and time window. To do this, we make the assumption that the vessels fully load at a supply port, travel to a demand port, and fully unload before returning to a supply port again. This can be a restrictive assumption for some applications; however, in this problem, each port has enough storage capacity to load or unload a full vessel, and so it is unnecessary to consider partial unloading of a vessel. And as mentioned in 10.2, this assumption is reasonable in some situations like liquefied natural gas shipping.

To begin specifying the nodes, consider a supply port j. Its initial level of inventory (at $t = 0$), is I_j^0; at time t, the inventory is $I_j(t) = I_j^0 + t \cdot I_j^\Delta$. At time t^a when $I_j(t^a) = Q$, there is enough inventory to fully load a vessel. At time t^b when $I_j(t^b) = I_j^C$, the port runs out of storage space. These times define the first time window during which the port must be visited. Now, assuming that the port has been visited p times already, the inventory level at time t is given by $I_j^0 + t \cdot I_j^\Delta - p \cdot Q$. The first time that the inventory reaches level Q again defines the beginning of the the $(p+1)^{th}$ time window. The time at which the inventory reaches the capacity I_j^C defines the end of the $(p+1)^{th}$ time window. The result is that we define nodes indexed by port and number of previous visits; node (j, p) corresponding to

supply port j that has been visited p times already has demand level $q_{(j,p)} = Q$ and time window $[a_{(j,p)}, b_{(j,p)}]$ where

$$a_{(j,p)} = (Q + pQ - I_j^0)/I_j^\Delta,$$

$$b_{(j,p)} = (I_j^C + pQ - I_j^0)/I_j^\Delta.$$

For demand ports, the definitions are similar; the inventory level at time t after the port has been visited p times already is $I_j^0 + t \cdot I_j^\Delta + p \cdot Q$. The beginning of the time windows are defined by the times at which the inventory level reaches $I_j^C - Q$, at which point there is enough space to accept a full vessel to unload. The end of the time windows are defined by the times at which the inventory level reaches zero. Then node (j, p) corresponding to demand port j that has been visited p times already has demand level $q_{(j,p)} = -Q$ and time window $[a_{(j,p)}, b_{(j,p)}]$ where

$$a_{(j,p)} = (I_j^C - Q - pQ - I_j^0)/I_j^\Delta,$$

$$b_{(j,p)} = (0 - pQ - I_j^0)/I_j^\Delta.$$

Note that we must have $I_j^C \geq Q$ or else the time window is nonsensical ($a_{(j,p)} > b_{(j,p)}$) and when $I_j^C = Q$ (as is the case for port D3) the time window is degenerate.

For a given time horizon, we construct nodes for each port until the time windows are no longer a subset of the time horizon; that is, all nodes have $b_{(j,p)} \leq T_H$.

To finish specifying the VRPTW, we need the arc data. We define arcs between any supply node and any demand node, and vice versa. This enforces the assumption that vessels do not travel directly between supply ports, or directly between demand ports. The travel time for an arc is simply the distance between the corresponding ports (see Table 10.7) divided by the vessel speed (665). The cost of the arc is the distance between the ports times the cost per unit distance (0.09), plus the port fee of the destination port (see Table 10.6). The original instance LR1_2_DR1_3_VC2_V6a includes time for loading and unloading vessels at the ports. For simplicity we ignore this feature, although we could include it by modifying

the travel times by adding in the loading/unloading time at the destination port. Note this might make the travel times asymmetric.

Not all of the arcs defined this way are physically reasonable due to the timing. We remove arcs $((j,p),(j',p'))$ where $a_{(j,p)} + t_{((j,p),(j',p'))} > b_{(j',p')}$; that is, the beginning of the time window of the origin node plus the travel time is greater than the end of the time window of the destination node.

The depot node in this problem is a dummy node, serving as an artificial source and sink for the vessels. Consequently, we assume that the initial loading Q_0 of the vessels is zero. In general, the number of vessels, their initial positions, and initial state (empty or full) are controlled through the specification of the entry arcs, which have the depot as their origin. The procedure for a general MIRP is as follows. For every vessel v we add a dummy node dum_v with time window $[0, +\infty]$. If the vessel is initially empty, the dummy node has demand level $q_{\text{dum}_v} = 0$; an arc from the depot to this dummy node is added, and then arcs from the dummy node to all supply nodes are added. If the vessel is initially fully loaded, the dummy node has demand level $q_{\text{dum}_v} = Q$ (to reflect the fact that this vessel visited a supply node sometime before the time horizon started); an arc from the depot to this dummy node is added, and then arcs from the dummy node to all demand nodes are added. The travel times from the dummy node could reflect the geographic position of the vessel, for instance, that it is in the middle of the ocean. To finish specifying the network, we add exit arcs from any node (including the dummy ones) back to the depot at zero travel time and zero cost.

However, the original instance LR1_2_DR1_3_VC2_V6a does not specify the initial states of the vessels, and so we are less constrained in defining the entry arcs. Furthermore, because of how the arc- and sequence-based formulations handle the capacity constraints, adding these dummy nodes might not be necessary. Consequently, the handling of entry arcs is

formulation-specific and discussed in the following subsection.

10.A.3 Details for each formulation

In this subsection, we go over any formulation-specific details required to handle the MIRP-as-VRPTW from the previous subsection. As mentioned, this includes the handling of entry arcs.

10.A.3.1 Route-based formulation

First, we specify the entry arcs for the formulation. Since vehicle capacity constraints are enforced through the definition of an allowed route, we must add dummy nodes as discussed in 10.A.2. We add entry arcs from the depot directly to any supply node with end of time window less than 14. Meanwhile, for every demand node with end of time window less than 14, we first add a dummy node with demand level Q. Then, we add arcs from the depot to this dummy node and then from the dummy node to the demand node (both with zero travel time and cost). This enforces the vessel to be fully loaded before it arrives at the demand node. The result is a set of seven vessels.

To finish specifying the route-based formulation, we need to define the set of routes $\mathcal{R}$. For this problem, we use a randomized greedy solution heuristic to propose routes. The core of this routine is given in Algorithm 13. This routine takes a scaling factor S and functions f_{time} and f_{node} that together assign a cost to visiting a node with a particular arrival time. The scaling factor S controls the amount of exploration/randomization that takes place, while f_{time} and f_{node} can be used to control the timing aspect of routes and which

nodes the routes tend to favor visiting. Here, we set

$$f_{time} : t \mapsto \begin{cases} 0, & t \leq 10 \\ 100t, & t > 10 \end{cases}$$

$$f_{node} : i \mapsto \begin{cases} 0, & i \in \mathcal{N} \\ (T_H/6) \cdot 1500, & i = d \end{cases}$$

This is intended to assign a high cost to routes that arrive at nodes later, thus promoting the generation of routes that greedily visit each node as early as possible, while simultaneously (through the action of f_{node}) discouraging early exit back to the depot.

The ultimate set of routes $\mathcal{R}$ is the unique set of routes generated by Algorithm 13 for increasing levels of randomization. Specifically, we run Algorithm 13 once for $S = 10^{-4}$, $\lfloor T_H \rfloor$ times for $S = 1$, and $\lfloor 10T_H \rfloor$ times for $S = +\infty$. For $S = +\infty$, the sampling at Step 15 in Algorithm 13 is from a uniform distribution over the valid proposed nodes.

10.A.3.2 Arc-based formulation

The additional data that we need to specify the arc-based formulation is the set of time periods $\mathcal{T}$. There are many options for determining this set; in general, we must balance detail and how finely we can model the timing of events, with how large the problem becomes. For this particular example, we choose the time periods to equal the set of integer time points that fall within any node's time window (besides the depot and any dummy node, which have infinite time windows), plus the initial time point $t = 0$. For this example, each node's time window contains at least one integer time point, and so this gives us at least a little flexibility in the timing of the arrival of vessels at a node. Further, this leads to some economy in the number of time points, since an integer time point might fall in the time window of multiple nodes.

Algorithm 13 Randomized greedy solution heuristic for route generation

Require: $S \geq 0$, $f_{time} : [0, T_H] \to \mathbb{R}$, $f_{node} : \mathcal{N} \cup \{d\} \to \mathbb{R}$

Initialize proposed routes: $\mathcal{R}_P = \varnothing$
Initialize unvisited nodes: $\mathcal{N}_U = \mathcal{N}$
for $v \in \mathcal{V}$ **do**
    Initialize route and current arrival time: $i_1 = d$, $t_r = 0$.
    for $p = 2, 3, 4, \ldots$ **do**
        Construct costs for proposed next nodes:
        for $i \in \mathcal{N}_U \cup \{d\}$ **do**
            if $(i_{p-1}, i) \notin \mathcal{A}$ **then**
                continue
            $t_r^i = \max\left\{a_i, t_r + t_{i_{p-1},i}\right\}$
            if $t_r^i > b_i$ **then**
                continue
            else
                $f_i = c_{i_{p-1},i} + f_{time}(t_r^i) + f_{node}(i)$
        Sample next node according to softmax of negative costs:

$$i_p \sim \mathbb{P}(i|i_{p-1}) \equiv \frac{\exp(-f_i/S)}{\sum_j \exp(-f_j/S)}.$$

        Update route and arrival time: $r \leftarrow (d, i_2, i_3, \ldots, i_p)$, $t_r \leftarrow t_r^{i_p}$
        if $i_p = d$ **then**
            End building route; break iteration over p
    Update proposed routes: $\mathcal{R}_P \leftarrow \mathcal{R}_P \cup \{r\}$
    Update set of unvisited nodes by removing those visited by route $r = (d, i_2, i_3, \ldots, i_{P-1}, d)$: $\mathcal{N}_U \leftarrow \mathcal{N}_U - \{i_2, i_3, \ldots, i_{P-1}\}$
return $\mathcal{R}_P$

Some care is required. Port D3 has degenerate time windows; however they happen to fall at integer time points. Changing the initial inventory level so that it is not an integer multiple of the consumption rate would affect this.

The arc-based formulation does not explicitly enforce capacity constraints (in this example vessel capacity is essentially enforced through the arcs, which only allow a vessel to visit an alternating sequence of supply and demand nodes). Consequently, the dummy nodes for enforcing the initial conditions of vessels are unnecessary. We add entry arcs from the depot directly to any supply node or demand node with end of time window less than 14.

As in the path-based formulation, this results in seven vessels being available.

10.A.3.3 Sequence-based formulation

To specify the sequence-based formulation, we need a maximum number of stops P for each vehicle. Given the length of the time horizon T_H, we estimate this based on the shortest travel time ($5305.34/665 \approx 8$). Thus, we set $P = \lfloor T_H/8 \rfloor + 2$.

As in the arc-based formulation, dummy nodes for enforcing the initial conditions of vessels are unnecessary. We add entry arcs from the depot directly to any supply node or demand node with end of time window less than 14.

The sequence-based formulation does not directly enforce timing constraints. As mentioned, we enforce them in a conservative way by pruning the arc set: for any arc (besides an entry arc from the depot), if the *end* of the time window of the origin node plus the travel time is greater than the end of the time window of the destination node, then that arc is removed. We essentially enforce timing constraints by assuming that vessels always arrive at the end of a node's time window. Something similar could be done by assuming that vessels always arrive at the beginning of a node's time window.

10.B Proof of sufficiently large penalty value

In this section, we establish in detail a value of the penalty parameter that is sufficient to make the route-based formulation (10.2) equivalent to minimizing the energy function

$$H : \mathbf{x} \mapsto \sum_r c_r x_r + \rho \sum_i (1 - \textstyle\sum_r \delta_{i,r} x_r)^2.$$

Proposition 10.B.1. Assume ρ satisfies

$$\rho > \sum_r |c_r|.$$

Then $\mathbf{x}^*$ is a solution of $\min\{H(\mathbf{x}) : \mathbf{x} \in \{0,1\}^n\}$ and problem (10.2) is feasible, if and only if $\mathbf{x}^*$ solves problem (10.2) (where $n = |\mathcal{R}|$).

Proof. If $\mathbf{x}^*$ solves (10.2), then it is feasible, so the penalty term is zero: $\sum_i (1 - \sum_r \delta_{i,r} x_r^*)^2 = 0$. Assume for a contradiction that there is an $\mathbf{x}^\dagger \in \{0,1\}^n$ with $H(\mathbf{x}^\dagger) < H(\mathbf{x}^*)$, or

$$\sum_r c_r x_r^\dagger + \rho \sum_i (1 - \textstyle\sum_r \delta_{i,r} x_r^\dagger)^2 < \sum_r c_r x_r^*. \tag{10.11}$$

If $\mathbf{x}^\dagger$ is feasible in (10.2), then the penalty term is zero, and so $\sum_r c_r x_r^\dagger < \sum_r c_r x_r^*$ which contradicts the optimality of $\mathbf{x}^*$; thus, we must have that $\mathbf{x}^\dagger$ is infeasible in (10.2). Since $x_r^\dagger$ and $\delta_{i,r}$ are $\{0,1\}$-valued for all r and i, the smallest value that $\rho \sum_i (1 - \sum_r \delta_{i,r} x_r^\dagger)^2$ can take is ρ (since it is infeasible, it cannot be zero). Meanwhile, by the (generalization of) the Cauchy-Schwarz inequality, $-\sum_r c_r (x_r^\dagger - x_r^*) \leq \|\mathbf{c}\|_* \left\|\mathbf{x}^\dagger - \mathbf{x}^*\right\|$ for any norm $\|\cdot\|$ and its dual norm $\|\cdot\|_*$. In particular, using the infinity-norm, we have $-\sum_r c_r (x_r^\dagger - x_r^*) \leq \|\mathbf{c}\|_1 \cdot 1$. Using $\rho \leq \rho \sum_i (1 - \sum_r \delta_{i,r} x_r^\dagger)^2$ and $-\|\mathbf{c}\|_1 \leq \sum_r c_r (x_r^\dagger - x_r^*)$ and plugging into (10.11), we have

$$-\|\mathbf{c}\|_1 + \rho < 0,$$

but upon rearranging and using the definition of the one-norm, we see this contradicts the assumption that $\rho > \sum_r |c_r|$. Thus $\mathbf{x}^* \in \arg\min_{\mathbf{x}} H(\mathbf{x})$.

Conversely, assume that $\mathbf{x}^\dagger$ solves $\min_{\mathbf{x}} H(\mathbf{x})$, and that problem (10.2) is feasible. We have $\min_{\mathbf{x}} H(\mathbf{x})$ must be less than or equal to the minimum objective value of (10.2); $H(\mathbf{x})$ equals the objective of (10.2) on the feasible set of (10.2), and the minimization of H is over a superset of the feasible set of (10.2), so the minimum must be less. Thus, we just need to establish that $\mathbf{x}^\dagger$ is feasible for (10.2). So, assume for a contradiction that $\mathbf{x}^\dagger$ is not feasible. By assumption, there exists $\mathbf{x}^*$ feasible in (10.2). Since $\mathbf{x}^\dagger$ minimizes H, we have

$$\sum_r c_r x_r^\dagger + \rho \sum_i (1 - \textstyle\sum_r \delta_{i,r} x_r^\dagger)^2 \leq \sum_r c_r x_r^*.$$

We can proceed exactly as before to obtain $-\|\mathbf{c}\|_1 + \rho \leq 0$, which still contradicts the assumption that $\rho > \sum_r |c_r|$. Therefore $\mathbf{x}^\dagger$ is feasible in (10.2), and thus optimal. □

Chapter 11

Conclusion

11.1 Summary of this Thesis

This section summarizes the significant accomplishments and findings from each chapter in the Thesis.

11.1.1 Chapter 2: a review and comparison of solvers for convex MINLP

In Chapter 2, we have completed an overview of the available software tools, or *solvers*, that have been developed to solve convex Mixed-Integer Nonlinear Programming (MINLP) problems. Moreover, we have performed a large computational study by testing these solvers with MINLP benchmark library MINLPLib [100], through the evaluation of 355 problems. These problems stemmed from a diverse set of sources, showing the versatility of convex MINLP as a modeling paradigm. Our results indicate the advances that have been made in practically solving these problems, and the difficulties that still exist when facing these challenging nonlinear discrete optimization problems. Through an initial classification of solver technologies, like those based on Branch & Bound techniques and those relying on Mixed-Integer Linear Programming based Decomposition, we were able to identify the overall advantages and disadvantages of these complementary methods. A key finding was that there is no dominant solver at this moment and not even a dominant solution technique, demonstrating the necessity of counting with an arsenal of algorithmic methods to tackle these optimization problems. Moreover, we provided a set of recommendations

for both problem modelers and solver developers to match the best solution strategies for problems with specific characteristics. Observing the solver performance concerning the problem's continuous relaxation gap, nonlinearity, and discrete density, pointing us towards the strengths and weaknesses of these solution techniques. Moreover, we acknowledge that the benchmarking of MINLP solvers needs to be performed objectively. Therefore, it is required to use comparison methods that generalize the solvers' performance beyond the chosen instances. We notice that implementing methods similar to the ones presented by Smith-Miles et al. [416] is desirable to achieve this goal.

We observed that there is still a gap between those problems that are practically solvable and what can be modeled through MINLP. Having identified this gap motivates us to keep developing solution methods and better modeling techniques for MINLP, thus motivating the remaining work presented in this Thesis.

11.1.2 Chapter 3: Feasibility Pump implementation in DICOPT

In Chapter 3, we present the implementation of the Feasibility pump algorithm for MINLP [143] in the commercial solver DICOPT [124]. The feasibility pump algorithm relies on the minimization of the constraints violation. This is achieved by iteratively minimizing the discreteness violation through a continuous problem subject to the nonlinear constraints and the nonlinear constraint violation by solving a linear discrete-valued problem. This chapter presents the difference between algorithms and solvers by showing a detailed description of the implementation details of a well-known algorithm in a commercial solver for MINLP. This difference allowed us to propose improvements to the original algorithm to adapt to the existing solver infrastructure. We proposed a new algorithm based on the integration of two existing algorithms, outer-approximation (OA) [122] and the iterative feasibility pump [143], both designed to provide guaranteed optimal solutions to convex

MINLP problems but with different features. We leverage the efficiency at finding optimal solutions from the feasibility pump to use is as a preprocessing step for the solver DICOPT, which implements the OA algorithm.

Obtaining a feasible solution early in the solution process through the feasibility pump benefits the DICOPT's performance on several fronts. It first provides a potentially feasible solution to the OA method to start its iterations. Secondly, it offers a valid linear relaxation of the nonlinear feasible set by initializing it with linear cuts found during the feasibility pump iterations. The hypothesis that the outer-approximation cuts generated while using the feasibility pump were useful for DICOPT was confirmed. It was verified that passing solution cuts to DICOPT reduced the computational time of the algorithm compared to simply using the best feasible solution as initialization. The proposed algorithm has shown better performance than DICOPT without the feasibility pump regarding solution quality, and has similar performance regarding efficiency and stability. Both advantages are necessary to improve the default performance of DICOPT, shown through experiments by solving 80 convex MINLP instances from MINLPlib. We propose a set of recommended settings that become the default in the commercial solver DICOPT when addressing convex MINLP problems through a set of computational tests.

11.1.3 Chapter 4: Center cut algorithm

Chapter 4 present another solution method for convex MINLP problems. This method uses the Chebyshev center of the region defined by the linear constraints to determine a trial solution for the discrete variables, or *integer combination*. The optimization problem of finding the Chebyshev center of a polyhedron, which corresponds to the center of the largest possible ball that fits within a linearly constrained region, can be written as a Linear Programming (LP) problem. We enforce the discrete variables in the original problem to

take integer values, resulting in an MILP problem. This integer combination is then fixed to solve a continuous Nonlinear Programming (NLP) problem, with the hypothesis that by using the center of the linear approximation, the corresponding integer combination is more likely to be a feasible solution of the convex MINLP problem. The solutions of the fixed NLP subproblem allow refining the linear Outer-approximation, which also considers the improvement of the objective function to reduce the volume of the linear relaxation region. The radius of the largest ball within the linear relaxation, whose center is the Chevishev center, is the convergence indicator for this method, which is guaranteed to tend to zero with increasing iterations. The iterative solution between the MILP that determines the Chebyshev center and the fixed NLPs problem results in an algorithm with global convergence guarantees for convex MINLP problems.

Besides deriving a convergent method that can be used as a deterministic solver for convex MINLP, we completed an implementation of this method. We experimented with it extensively using the problem library MINLPLib. Our implementation and computational results have shown this algorithm to be a very efficient procedure to find good quality feasible solutions to convex MINLP problems. Its performance is comparable to the Feasibility pump implementation in DICOPT described in Chapter 3. Furthermore, either of these methods performs better in finding good quality feasible solutions than the other depending on the instance, with both showing an advantage compared to commercial solvers. These results motivate implementing both of these methods in a flexible solver, intending to be available when addressing these convex discrete optimization problems.

11.1.4 Chapter 5: Outer-approximation with Quadratic Cuts

Chapter 5 introduces the concept of quadratic cuts for the Outer-approximation (OA) method for solving convex MINLP problems. This approach generalizes OA by allowing

quadratic inequalities, instead of linear, to be iteratively added to a driver Mixed-Integer Quadtratically Constrained Programming (MIQCP) problem. The development of this approach is largely motivated by the possibility of an efficient solution of MIQCP problems from commercial MILP solvers, and the observation that for many MINLP problems the quality of the relaxation given by linear approximations is low. The relaxation derived by this method, by definition, dominates the linear gradient-based relaxation from OA. To guarantee the validity of the relaxation, we construct a scaled second-order Taylor approximation of the nonlinear constraints. The scaling factor of the quadratic term is a coefficient multiplying the quadratic term in the Taylor expansion. We propose rigorous methods to find such a scaling factor and show that under certain technical conditions on the nonlinear functions, it can be computed by evaluating a function at its vertices.

We then apply this method by generating a single or multiple different quadratic inequalities in an approach we denote *multi-cut*, at each OA iteration. Moreover, we extend another Decomposition method for MINLP, the Partial Surrogate Cuts (PSC) [90] method, with the scaled quadratic cuts and the multi-cut strategy. We also explore the dynamic application of linear and quadratic cuts in OA and PSC, leading to a hybrid method. The motivation is that the linear approximations might be used for early iterations, where the incumbent solutions are still far from the optimum. The quadratic inequalities that yield expensive MIQCP problems can be used at later iterations when the points where these approximations are generated are near the optimal solution.

The combination of single- and multi-cut, hybrid and purely quadratic relaxations, and OA and PSC resulted in six new optimization methods for convex MINLP. Although the use of PSC might reduce the size of the mixed-integer subproblems, the weaker relaxation it yields results in a larger number of iterations, which for our results ultimately lead to higher computational times. Along these same lines, the use of quadratic inequalities

results in more expensive iterations. However, the quality of the relaxation is such that the total number of iterations is reduced. To alleviate this issue, deriving multiple cuts at each iteration, derived from the different solutions found while solving the mixed-integer problems, results in an overall reduction in the time required to solve these problems. The factors listed above lead to the most efficient method compared in this chapter being OA with multiple quadratic cuts derived at each iteration.

11.1.5 Chapter 6: Use of Regularization and Second-Order Information for Outer-approximation

Chapter 6 proposes another extension to the Outer-approximation method. Though a generalization of the level-based method proposed by Lemaréchal, Nemirovskii, and Nesterov [257], the OA method is enhanced through the solution of an auxiliary Mixed-Integer Quadratic Programming (MIQP) problem at every iteration. This problem contains the same constraints as the driver mixed-integer problem in OA with an additional constraint. This extra constraint forces the solution of the MIQP to be above a certain level of the objective function, estimated through interpolation of the optimal objective bounds and controlled through a trust hyperparameter. This problem also has a different objective function, which minimizes the squared ℓ_2-distance to the incumbent solution. The objective function is similar to one of the modified objectives in the feasibility pump as described in Chapter 3, and the solution of this problem defines the integer combination to solve the NLP subproblems as in the Center-cut algorithm presented in Chapter 4.

An interesting fact about this method is that we prove it to be equivalent to using a trust-region constraint in the mixed-integer subproblem of OA, allowing the developments on trust-region methods for continuous optimization [267] to be explored within for MINLP. Alternatively, we have proved that this method is guaranteed to converge in a finite number

of iterations. The auxiliary MIQP problem is guaranteed to provide new integer combinations as long as the algorithm has not converged. More interestingly is that the convergence guarantees of this method do not rely on the solution to optimality of the MIQP problem, and that the solution of the MILP subproblem is also feasible for the auxiliary problem.

As in the previous chapter, the second-order information is exploited to improve the performance of this method. For this particular case, we consider the second-order Taylor approximation at the optimal solution of the NLP problems of the Lagrangean function as the objective to the auxiliary MIQP problem. The second-order approximation of the Lagrangean as an objective informs the auxiliary MIQP problem about the curvature of the constraints and allows a practically efficient implementation given the availability of the Hessian of the Lagrangean from the NLP subproblems.

These two approaches were able to reduce the number of total iterations compared to the OA method. Having a convex region defined by the nonlinear constraints motivates the idea of stabilizing the OA method by exploring the neighborhood of the best-found solution. This hypothesis is supported by the results included in this chapter. The more expensive iterations requiring the suboptimal solution of the MIQP auxiliary problems result in an overall reduction in the solution time for the convex MINLP problems.

11.1.6 Chapter 7: Alternative Regularizations for Outer-approximation

Chapter 7 presents a general framework for regularization in the Outer-approximation method, generalizing the two algorithms proposed in Chapter 6. The key observation for this chapter is that the convergence guarantees of the second-order regularization methods do not rely on the chosen objective for the auxiliary MIQP problem. This observation allows the objective to be chosen such that it can be representable with linear constraints, for example, opening up the possibility of implementing other regularization options. In

terms of the distance to the incumbent solution, we propose using the ℓ_1 and ℓ_∞ norms, which are representable with linear constraints. Furthermore, we consider alternatives to the second-order Taylor series approximation of the Lagrangean, such as the first-order Taylor approximation, the Hessian of the Lagrangian, and a quasi-Newton approximation of the Lagrangian function.

We also extend the regularization methods to the LP/NLP Branch & Bound algorithm proposed by Quesada and Grossmann [90]. This extension allows for a single MILP driver problem to be solved in what is known in the literature as *single-tree* approach in contrast to the traditional OA or *multi-tree* approach that involves the solution og a sequence of MILP subproblems. At each incumbent solution of the driver MILP Branch & Bound tree, the regularization mixed-integer problem and its corresponding NLP subproblem are solved. Complementary to this, the implementation has been made widely available through the **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy [190].

This open-source and flexible implementation allowed us to perform a large computational study comparing seven different objectives alternatives for the regularization problem, both in the multi- and single-tree settings showing the potential of regularization techniques in convex MINLP.

We observed increased stability in the OA method when considering the regularization approach, as long as the objective function was indeed a regularizer. Moreover, the flexibility in the objective function of the regularizer allows for it to be chosen as a linear representable function, taking advantage of the more mature solution methods of MILP compared to MIQP. The case of the linear approximation of the Lagrangean is an example of an invalid regularizer where the method's performance was compromised, even if the regularization subproblem was an MILP.

11.1.7 Chapter 8: Easily Solvable Convex MINLP Derived from Generalized Disjunctive Programming using Cones

Chapter 8 offers a different approach toward the solution of convex MINLP problems. Instead of advancing the state-of-the-art solution algorithms for these problems, it tackles the formulation of the problems. In particular, a large subset of convex MINLP problems arises from disjunctive constraints, where a set of nonlinear constraints are enforced or relaxed based on a discrete choice. An amenable representation of such problems is Generalized Disjunctive Programming (GDP), where the discrete choices are represented through logical variables, and the sets of constraints are included inside of disjunctions. This modeling framework extends Disjunctive Programming (DP), the optimization over disjunctive sets, by allowing logical propositions between the logical variables and global constraints outside of the disjunctions to be considered. The relationship between GDP and DP allows for the reformulation of GDP into a Mixed-Integer Programming problem, where linearity depends on the GDP constraints' linearity. These reformulations introduce a binary variable for each logical variable and enforce the constraints in the disjunctions through algebraic constraints involving those binary variables. Two transformations are considered between GDP and MINLP, the Big-M and the Hull Reformulation (HR). There is a trade-off between these two transformations in terms of the number of continuous variables in the resulting MINLP and how close, or *tight*, the solution of the MINLP continuous relaxation is from the original problem optimal solution. In the case of the HR, the MINLP involves the perspective of the functions in the disjunctions. In the case that these functions are nonlinear, its perspective becomes non-differentiable at points where the binary variables are equal to zero, making its optimization challenging and motivating the derivation of approximations of the perspective function [62].

When considering a convex GDP, contrary to the traditional algebraic representation

through inequality constraints, we represent the convex sets in its disjunctions through cones. The optimization over cones, or Conic Programming (CP), uses these convex sets' features, such as conic duality, to derive efficient and stable algorithms. We propose the formulation of convex GDP problems as conic GDP problems, whose mixed-integer reformulation is also representable via the same cones. The resulting Mixed-Integer Conic Programming (MICP) problems were solved using different solvers and compared against the MINLP reformulations stemming from the algebraic representation of the convex GDP problems. The conic formulation of convex MINLP allows for a natural and systematic lifted reformulation of the problem that benefits OA methods. An interesting observation is that the perspective function could be represented exactly through conic sets, avoiding approximations required in the nonlinear HR.

To test the effects of the conic formulation, we implemented a large set of convex GDP instances. These instances included optimization problems relevant to Chemical Engineering, such as the constrained safety layout problem, the optimal process network design, and the synthesis and retrofit process sytneshsis problems. We were also able to use these formulations to tackle problems relevant to Machine Learning, such as k-mean clustering problems and logistic regression problems, and randomly generated instances that were representable using quadratic and exponential cones. This exact formulation of the tight mixed-integer HR of disjunctive sets allows the problems to be solved more efficiently through solvers that exploit the conic structure.

11.1.8 Chapter 9: Characterization of QUBO Reformulations for the Maximum k-colorable Subgraph Problem

Chapter 9 is the first chapter in this Thesis that covers the reformulation of a constrained Integer Programming problem as a Quadratic Unconstrained Binary Optimization (QUBO)

problem. The reformulation of optimization problems as QUBO is of interest given the one-to-one mapping between QUBO and the transverse field Ising model. This particular model is relevant in this Thesis since it is the framework where Quantum Computing approaches can tackle combinatorial optimization problems. In particular, the Ising model is mapped to the energy function, or *Hamiltonian*, of a Quantum system. This system is defined by a set of quantum bits, or *qubits*, whose final state represents a binary variable in the QUBO problem. The algorithms executed in Quantum Computers designed for combinatorial optimization aim to minimize this energy function, and hence the original QUBO problem. Among those algorithms, we see results in this Thesis corresponding to Quantum Annealing (QA) implemented in tailored hardware to run this algorithm, known as *Quantum Annealers*, and the Quantum Optimization Approximation Algorithm (QAOA) and Variational Quantum Eigensolver (VQE) algorithms implementable in Gate-based Quantum Computers.

The problem at hand is the maximum k-colorable subgraph (MkCS) problem, a problem from graph theory. Given a graph, we need to find the largest subgraph such that k different colors are assigned to each non-adjacent node in it. This problem is a challenging combinatorial NP-complete optimization problem with many applications in Science and Engineering. This problem can be posed as a Binary Linear Program with inequality constraints. One constraint represents the non-adjacency of same-colored nodes. The second one is the requirement of coloring each node with at most one color. The usual transformation of Binary Linearly constrained problems into QUBO entails adding slack variables to transform the inequality constraints into equalities, followed by penalizing the constraints in the objective function. The penalization of the constraints is particularly important in the reformulation of problems into QUBO. Large values tend to affect the precision at which the Quantum Computers can represent problems. Simultaneously, these penalization factors need to be large enough to effectively represent the existence of

constraints in an unconstrained problem, *i.e.*, that the objective function corresponding to a solution that violates a constraint is larger than the optimal feasible solution.

In the case of the M*k*CS, we first derive the tightest bounds for the penalization factors of each constraint that yield a valid reformulation of the problem. We then propose a nonlinear formulation of the M*k*CS, which would be more challenging to solve using classical methods given the introduction of bilinear equality constraints. However, for QUBOs, it yields a smaller problem given that slack variables are no longer required to reformulate the problem. We also derive tight values for the multipliers in this nonlinear formulation. Both these formulations are benchmarked using Quantum Annealing. In the first place, we characterize the *minimum gap* of each formulation, which represents the difference between the two smallest eigenvalues of a Hamiltonian along its time evolution. This term represents the success of finding the optimal solution in the theoretical ideal limit of Quantum Annealing, the Quantum Adiabatic Algorithm, where for larger values of the minimum gap, the required time to perform an adiabatic evolution decreases, hence making the optimization problem more likely to succeed. We continue the study by comparing the minor-embedding of the resulting QUBO formulations into two Quantum Annealing chips available from the company D-Wave. Given the physical difficulties of setting up controllable quantum systems, the chips that can encode the minimization of the energy of an Ising model can only represent a model defined over a not fully-connected graph, whose nodes are quantum bits, or *qubits*, and whose edges are possible connections, or *couplers*. An NP-Hard graph-theoretical operation called *minor-embedding* has to be performed to encode an arbitrary QUBO into these chips. The original QUBOs are reformulated in a way that they can be solved in the Quantum Annealers. Finally, we run the minor-embedded QUBOs on the Quantum Annealer and compare a metric of success of this algorithm know as *Time-to-solution* (TTS), which denotes the expected time to obtain the optimal solution of

a Hamiltonian minimization with a determined confidence percentage.

This comparison is made for a large population of randomly generated graphs, varying their size and density while considering different values for the colors k and the penalty factors. Our results show that the nonlinear reformulation of the MkCS problem is superior to the linear reformulation according to all the metrics that we compared against each other. In terms of the minimum gap, comparing a limited set of results product of the computational complexity of this calculation, we reject the null hypothesis that the linear reformulation yields a larger minimum gap than the nonlinear reformulation up to a 2% with a 95% confidence. Loosely speaking, this means that the minimum gap for the nonlinear formulation is 2% higher than the linear formulation, proving a potential advantage og the Adiabatic Quantum Algorithm framework. In terms of embedding, the smaller and sparser QUBOs that result from the nonlinear formulation allow solving problems of the same size requiring only a fraction of the qubits. More importantly, given the current limitation in the size of the chips, the nonlinear formulation could be embedded for problem sizes for which it is impossible to embed the linear reformulation. In terms of TTS, the nonlinear reformulation yields orders of magnitude improvements compared to the linear reformulation. This even in the cases where the linear reformulation was solvable. For the problem sizes where the instances could not even be embedded, the nonlinear reformulation resulted in the only way to solve these problems directly using the existing Quantum Annealers. This observation supports our main conclusion. The correct reformulations of combinatorial optimization problems are necessary to efficiently exploit unconventional computing methods.

11.1.9 Chapter 10: Formulating and Solving Routing Problems on Quantum Computers

Similarly to Chapter 9, Chapter 10 presents the different formulations of a constrained optimization problem as a QUBO and evaluates their performance in terms of metric relevant from a Quantum Computing perspective. This chapter presents different Vehicle Routing Problem with Time Windows (VRPTW) formulations, motivated by the industrially relevant Maritime Inventory Routing Problem (MIRP). This problem minimizes the cost of a set of vehicles visiting a set of customers within predefined time intervals, or *windows*, traveling through a pre-specified graph delivering or picking up product at each customer subject to inventory constraints and satisfying all the customer demands. Three different formulations are proposed to be solved directly as a QUBO, a Route-based formulation, an Arc-based formulation. Each formulation was compared against each other in terms of the resulting QUBO scaling with respect to the growth in the inputs, *e.g.,* the time horizon, its solution landscape, *i.e.,* the feasible solutions' objective function distribution, and numerical simulation of the quantum algorithms. Moreover, the sequence-based formulation was extended to include continuous variables and then tackled using an Alternating Direction Method of Multipliers (ADMM) heurstic [388], which relies on subproblems solutions via Quantum methods. This problem was tackled through QUBO solution algorithms that are implementable in Gate-based Quantum computers. In particular, the algorithms chosen are variational algorithms, *i.e.,* algorithms where a set, or *circuit*, of quantum operators, or *gates*, are parametrized by a set of continuous values, or *rotation angles*. An objective function, *e.g.,* the expectation value of the energy of the Hamiltonian, is efficiently evaluated by executing the circuit on a quantum computer. A classical optimizer can then variate the rotation angles to optimize the given objective. This parametrized circuit, or *ansatz*, is designed so that its states represent the least-energy states of the Hamiltonian with high

accuracy.

We can draw several conclusions based on the obtained results. Preprocessing techniques used to generate and trim the route-based formulation successfully reduced the problem size, a significant limitation of the existing quantum hardware. On the other hand, this formulation was the least sparse, which is also affected by another quantum hardware limitation. Taking advantage of the different formulations is a way to avoid the hurdles that each has, *e.g.*, the route-based preprocessing techniques could be applied to reduce the size of sparser problem representations, such as the arc- or sequence-based formulations.

The results from this chapter also shed light on the different quantum algorithms to address QUBO problems. We identify a trade-off between those algorithms that require a larger number of ansatz parameters, hence requiring few iterations to represent the optimal solution of the problems and those with few parameters that can be efficiently optimized classically. The two extremes of this trade-off are represented by QAOA, with the fewest parameters and therefore being desirable to take many circuit samples, and VQE with the most flexible ansatz, being the best fit given a limited sampling budget.

Finally, the results for the ADMM algorithm show the potential of developing and using hybrid algorithms where subproblems are QUBO. In this case, we can enforce constraints or solve problems over variable domains that are hard to represent using quantum computers, such as continuous variables. We can do this while taking advantage of the quantum methods to efficiently address these subproblems.

11.1.10 Chapter A: Integer Programming Techniques for Minor-Embedding in Quantum Annealers

The results in Chapter A in the Appendix are slightly out-of-scope in this Thesis. We include this chapter as an appendix since it covers a relevant problem for one of the Quantum

Computing technologies used in this Thesis. This chapter discusses the solution to the minor-embedding problem using Integer Programing Techniques. As mentioned above, the minor-embedding problem is a graph-theoretical problem relevant to Quantum Annealers. The current quantum computers do not offer full connectivity among their qubits. Therefore, when trying to implement algorithms or problems that involve arbitrary connectivity among qubits, a pre-compilation step needs to be completed. For the case of Quantum Annealing, the D-Wave devices offer a chip architecture, called *chimera* and *pegasus*, which consists of cells of 4×4 bipartite graphs interconnected among them. Although designed to be as connected as possible, these chip topologies are still relatively sparse. For instance, to implement a problem that requires higher connectivity than the chip, the minor-embedding process must be performed. This procedure requires representing a variable in two or more qubits and forcing those qubits to hold the same value during the problem's solution. The duplicated qubits need to be connected between each other in the chip and need to implement the variable interactions of the original problem. In practice, this problem is solved heuristically, yielding potentially suboptimal solutions. Determining a success metric or objective function for the minor-embedding problem is not well defined, but there exist several alternatives that are desirable, *e.g.*, embeddings that require the fewest number of qubits in order to take the most advantage of the size-limited chips.

We formulated the minor-embedding problems using an Integer Linear Programming (ILP) problem. This problem encoded all the embedding requirements as constraints and the desired properties in the objective function. The resulting ILP problem was solved using commercial solvers directly, on an *monolithic* approach, and through a tailored decomposition algorithm, which allowed solving problems that were challenging for the heuristic methods. Moreover, we could obtain embeddings to application problems in Engineering, such as the minimum Spanning tree for vehicle communication and Protein Folding

problems. The decomposition approach was more efficient at finding feasible solutions to this problem but closing the optimality gap for these problems was more efficiently achieved using the monolithic method. A considerable limitation of this approach was the scalability of the methods. Finding optimal solutions to the embedding problem via Integer Programming techniques is limited by the size of the problem, with the largest problem solved in our work having 81 qubits and requiring one hour of computation. Given that the embedding problem is a pre-compilation step using that large amount of time is not acceptable for the given application. Furthermore, the latest generation D-Wave Quantum Annealers, with the pegasus architecture, has more than 50000 qubits, leading to substantial running times and exposing a weakness of this method.

11.2 Research contributions

The major original research contributions of this Thesis are the following:

1. Provided a classification and benchmark of the solution codes, *solvers*, for convex Mixed-Integer Nonlinear Programming (MINLP) problems by gathering and solving the largest MINLP problem library publicly available.
2. Proposed a novel heuristic method for convex MINLP, modifying the feasibility pump algorithm [143], and implemented it in the commercial solver DICOPT [214] improving the solver's performance with this problem class. The implementation became the default option when solving convex MINLP problems with this solver.
3. Introduced a new deterministic solution method for convex MINLP problems based on the Chebyshev center denominated the Center-cut algorithm. This method proved to be very efficient for finding feasible solutions to highly nonlinear convex MINLP problems.

4. Extended the Outer-approximation (OA) method by proposing the scaled quadratic cuts, which derive a Mixed-Integer Quadratically Contrained (MIQCP) master problem through the guaranteed valid quadratic approximation of the nonlinear constraints. This extension generalizes the single-iteration convergence property of OA for Mixed-Integer Linear Programming (MILP) problems to MIQCP problems.
5. Introduced the concept of Regularized Outer-approximation, which by means of the solution of an auxiliary mixed-integer problem at every iteration, stabilizes the convergence of OA. This method is guaranteed to converge to the global optimal solution of convex MINLP problems and has shown its improved performance compared to OA.
6. Proposed the formulation of convex Generalized Disjunctive Programming (GDP) problems using conic sets, allowing for conic programming methods to be used for the efficient solution of convex GDP programs through the reformulation to Mixed-Integer Conic Programming (MICP) problems. This conic reformulation avoids approximating the non-differentiable perspective function and enables the solution of these problems through an extended formulation using specialized conic programming solvers.
7. Implemented 425 examples of convex GDP problems as conic GDP problems. These instances, which showed improved computational performance compared to non-conic formulations, included over two hundred applications-related instances relevant to Process Systems Engineering (PSE) and Machine learning (ML).
8. Proposed a nonlinear formulation of the Maximum k Colorable Subset (MkCS) problem, which once transformed into a Quadratic Unconstrained Binary Optimization (QUBO) problem improves over the classical linear formulation in metrics relevant to the solution of the problem via QuantumAnnealing such as minimum gap, embedding

size, and time-to-solution.

9. Evaluated and compared three different discrete formulations of the Vehicle Routing Problem with Time Windows (VRPTW) with applications in Maritime Inventory Routing Problems (MIRP) for its solution through their QUBO reformulation and their solution via variational algorithms for Gate-based Quantum Computers. A route-based formulation allows for classical preprocessing which benefits the representation of the VRPTW as a QUBO. Sequence- and arc-based formulations show favorable characteristics, such as sparse QUBO representation, which are desirable for solving larger instances where classical methods struggle. This study also provided a clear picture among the trade-offs of variational quantum algorithms in gate-based circuits, with QAOA being the least flexible in terms of problem representation but the easiest to optimize given a large budget on quantum circuit evaluation, while VQE proving to the the most succesful algorithm in this comparison with a limited number of circuit evaluations.
10. Proposed Integer Programming based formulation of the minor-embedding problem, relevant to the compilation of Quantum Annealing problems, obtaining a solution method with solution quality guarantees which are advantageous in some instances compared to the traditionally used heuristic methods.

11.3 Papers produced from this dissertation

1. Jan Kronqvist, David E Bernal, Andreas Lundell, and Ignacio E Grossmann. "A review and comparison of solvers for convex MINLP". *Optimization and Engineering* 20.2 (2019), pp. 397–455.
2. David E Bernal, Stefan Vigerske, Francisco Trespalacios, and Ignacio E Grossmann.

"Improving the performance of DICOPT in convex MINLP problems using a feasibility pump". *Optimization Methods and Software* 35.1 (2020), pp. 171–190.

3. Jan Kronqvist, David E Bernal, Andreas Lundell, and Tapio Westerlund. "A center-cut algorithm for quickly obtaining feasible solutions and solving convex MINLP problems". *Computers & Chemical Engineering* 122 (2019), pp. 105–113.
4. Lijie Su, Lixin Tang, David E Bernal, and Ignacio E Grossmann. "Improved quadratic cuts for convex mixed-integer nonlinear programs". *Computers & Chemical Engineering* 109 (2018), pp. 77–95.
5. Jan Kronqvist, David E Bernal, and Ignacio E Grossmann. "Using regularization and second order information in outer approximation for convex MINLP". *Mathematical Programming* 180.1 (2020), pp. 285–310.
6. Stuart Harwood, Claudio Gambella, Dimitar Trenev, Andrea Simonetto, David E Bernal, and Donny Greenberg. "Formulating and Solving Routing Problems on Quantum Computers". *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–17.

Preprints published from this dissertation

1. Rodolfo Quintero, David E. Bernal, Tamás Terlaky, and Luis F Zuluaga. "Characterization of QUBO reformulations for the maximum k-colorable subgraph problem". *arXiv preprint arXiv:2101.09462* (2021)
2. David E Bernal, Sridhar Tayur, and Davide Venturelli. "Quantum Integer Programming (QuIP) 47-779: Lecture Notes". *arXiv preprint arXiv:2012.11382* (2020)

11.3.1 Collaborations

This dissertation has benefited from external collaborations. The works below are products of these collaborations, for which we were privileged to serve as secondary contributors.

1. Qi Chen, Emma S Johnson, David E Bernal, Romeo Valentin, Sunjeev Kale, Johnny

Bates, John D Siirola, and Ignacio E Grossmann. "Pyomo. GDP: an ecosystem for logic based modeling and optimization development". *Optimization and Engineering* (2021), pp. 1–36.

2. Can Li, David E Bernal, Kevin C Furman, Marco A Duran, and Ignacio E Grossmann. "Sample average approximation for stochastic nonconvex mixed integer nonlinear programming via outer-approximation". *Optimization and Engineering* (2020), pp. 1–29.
3. Haokun Yang, David E Bernal, Robert E Franzoi, Faramroze G Engineer, Kysang Kwon, Sechan Lee, and Ignacio E Grossmann. "Integration of crude-oil scheduling and refinery planning by Lagrangean Decomposition". *Computers & Chemical Engineering* 138 (2020), p. 106812.
4. Cristiana L Lara, David E Bernal, Can Li, and Ignacio E Grossmann. "Global optimization algorithm for multi-period design and planning of centralized and distributed manufacturing networks". *Computers & Chemical Engineering* 127 (2019), pp. 295–310.

11.3.2 Conference proceedings papers

This dissertation has also resulted in several conference proceeding papers. We provide the listing below for the reader's convenience, as some are not otherwise reflected in this thesis document.

1. David E Bernal, Kyle EC Booth, Raouf Dridi, Hedayat Alghassi, Sridhar Tayur, and Davide Venturelli. "Integer programming techniques for minor-embedding in quantum annealers". In: *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer. 2020, pp. 112–129.
2. David E Bernal, Qi Chen, Felicity Gong, and Ignacio E Grossmann. "Mixed-integer nonlinear decomposition toolbox for Pyomo (MindtPy)". In: *Computer Aided Chemical Engineering*. Vol. 44. Elsevier, 2018, pp. 895–900.

3. Lijie Su, Lixin Tang, David E Bernal, Ignacio E Grossmann, and Bowen Wang. "Integrated scheduling of on-line blending and distribution of oil products in refinery operation". In: *Computer Aided Chemical Engineering*. Vol. 44. Elsevier, 2018, pp. 1213–1218.

11.4 Thesis limitations and Future research directions

11.4.1 Sustainable Solver Benchmarks for MINLP

A limitation of the work on Chapter 2 is given by the problems that were tested. There was a clear dominance of certain families of problems, which bias the conclusions made by studying solver performance over this set of problems. Moreover, the development of novel algorithms and their efficient implementation in solvers have made a considerable portion of the problems in the library trivially solvable. This was observed through the *Virtual best* solver, which considers the best solver for each different problem results in 95% of the total instances being solvable within 15 minutes in a standard desktop. Simultaneously, some instances in the library were challenging, to the point they haven't yet been solved. An effort similar to MIPLib2017 [419], where a data-driven approach was taken to determine the problem benchmark for MILP solvers, should be followed for the MINLP problem libraries. The first step is gathering those instances, for which efforts such as MINLP.org[301] and MINLPLib[88] are crucial.

A second limitation is that the accelerated algorithm development requires the constant update of the solver comparison. The original paper of this chapter, [26], was published in 2018. Since then, more than 50 instances have been added to the problem library. All solvers have released a new version, and new solvers have been introduced as MINLP solvers. One interesting case is that traditionally MILP solvers CPLEX[52], Gurobi [51], and

Xpress [185] have extended the problem classes that they can address into MINLP. Developing an automated, reliable, representative, up-to-date, informative, and replicable solver benchmark is of great interest to the community and should be pursued. Efforts similar to the ones carried by Prof. Hans Mittelmann at Arizona State University[420] on constant solver benchmarking should be followed to track the development of solver technology in convex MINLP. The first steps towards this goal have been made the Grossmann's group Pyomo MINLP Benchmarking Tools in GitHub repository⋆.

11.4.2 Development of Mixed-Integer Nonlinear Decomposition Toolbox

Many of the algorithmic developments presented in this Thesis entail the Decomposition of MINLP problems and the solution of existing subproblems using solvers specialized for such subproblems, such as MILP, NLP, or QUBO. Simultaneously, several of the algorithms presented in this Thesis have been implemented on experimental code, not allowing other users to use them directly. With that motivation, we started the **M**ixed-**i**nteger **n**onlinear **d**ecomposition **t**oolbox for **Py**omo - MindtPy [190]. This toolbox presents an open-source⋆ implementation of several decomposition methods for MINLP, including Outer-approximation [30], Exteded Cutting Planes[29], LP/NLP Branch & Bound [90], and Generalized Benders Decomposition[28]. It also has implementations of the regularization methods presented in Chapters 6 and 7 and the feasibility pump in a similar way as it was implemented in DICOPT, as discussed in Chapter 3. Still missing is the implementation of several other methods presented in this Thesis, such as the Center-cut algorithm[145], the Partial Surrogate Cuts method [90], the Scaled Quadratic cuts OA [235], the Extended Supported Hyperplane [121].

⋆https://github.com/grossmann-group/pyomo-MINLP-benchmarking

⋆https://pyomo.readthedocs.io/en/stable/contributed_packages/mindtpy.html

Although MindtPy already counts with a battery of implemented methods, several other proposals have been made for MINLP solving based on Pyomo and Python, such as DIOR[195], GALINI[421], CORAMIN[422], and Decogo[264]. A current opportunity is identifying if these libraries can be merged into a single centralized MINLP solution Toolbox.

11.4.3 Estimating the Scaling Factor for the Quadratic Cut in Outer-approximation

The limitation of the approach presented in Chapter 5 lies in the computation of the scaling factor for the second-order Taylor series approximation of the constraints. In the Thesis, we propose a method that finds the scaling factor that provides the tightest quadratic relaxation at the expense of solving a low-dimensional non-convex optimization problem for each constraint at each iteration. Moreover, we require information on the Hessian of the constraints. This step might become prohibitive for the algorithm. Simultaneously, given some assumptions on the nonlinear constraints, we could derive a more efficient procedure to find the scaling factor. In some instances, the value of the scaling parameter can be derived analytically, as in the case where the original problem is an MIQCP problem where the scaling parameter equals one, and the problem would be solved in a single iteration using our approach. Efficiently estimating a non-trivial lower bound on the scaling parameter would be a step in the right direction of incorporating valid quadratic cuts in OA without requiring the expensive optimization required to estimate that parameter. The optimization problem to obtain the scaling parameter is the ratio between a convex function and a convex quadratic function, hence using methods from fractional programming[423] might lead to an efficient estimation of this scaling parameter.

11.4.4 Extension of the convex MINLP methods to non-convex MINLP

The first part of this Thesis presents a set of approaches to tackle convex MINLP problems. Many of these methods might fail to return a feasible solution when the Assumption on convexity from the objective and constraints fails. There is a lot of potential in extending some of the proposed ideas for solving non-convex MINLP problems. A first approach has been taken in this direction by implementing the heuristic strategies of DICOPT [124] for non-convex MINLP: equality relaxation, augmented penalty, and stopping on worsening, in MindtPy. We have experimented with generalizing some of the heuristic techniques implemented in DICOPT for non-convex problems, described in Section 3.2.2. This first approach entails the softening of the constraints through the augmented penalty method in the feasibility pump mentioned in Chapter 3.

The Outer-approximation extensions for non-convex problems [240] have also been implemented in MindtPy, using the library MC++[424] to derive valid under- and over-estimators of non-convex functions. This implementation is still in its infancy, and it is an exciting path forward to improving the methods presented in this Thesis.

11.4.5 Generalization of convex MINLP methods to Conic Programming

Recently, the Outer-approximation (OA) method was generalized for Mixed-integer Conic Programming (MICP) problems [44]. Properties of the conic programs were used to strengthen the OA method, allowing efficient, efficient solutions of MICP problems via an OA algorithm. Given the positive experience observed in this Thesis on modeling convex discrete nonlinear optimization using cones, a natural next step is exploring the use of the OA extensions proposed herein to solve MICP problems. Particularly interesting approaches are the ones relying on convex quadratic objectives and cuts such as the ones presented in Chapters 5 and 6, which as seen in Chapter 8 can be formulated using

second-order and rotated second-order cones.

11.4.6 Automatic identification of Exponential cones

An evident limitation of using Conic Programming tools for convex optimization is the requirements of performing a translation between the algebraic inequalities into conic sets. This issue can be alleviated through Disciplined Convex Programming (DCP)[41] with would leave the user the task of writing down the constraints as an easily translatable input to cones. Unfortunately, this is not the most straightforward alternative for users who might be unfamiliar with the conic sets-based formulation paradigm of the DCP approach. Although Table 1 in Chapter8 provides a set of translations, it would be desirable if a computer performed this translation automatically. This has already been successfully implemented in the case of quadratic constraints being transformed into second-order cones; hence its extension for exponential constraints is worth exploring in the future. rencent proposals on optimization model transformations, *e.g.*, the *Bridges* from MatOptInterface in JuMP[97] and the *transformations* in Pyomo[273] seem to be the right tools to design these transformations.

11.4.7 Heuristic methods for Generalized Disjunctive Programming

Chapters 3 and 4 show how knowledge of the original problem mathematical structure can lead to the design of efficient heuristic methods. Moreover, given the increasingly challenging nonlinear discrete optimization problems that need to be solved, heuristic methods can enable the practical solution of these problems.

The solution methods for GDP problems either rely on the reformulation of the problems into MINLP, as seen in Chapter 8, or on logic-based approaches, with algorithms that explore the discrete feasible space of the logical variables followed by the solution of NLP

subproblems with only the relevant constraints included in the optimization problem [59].

The recent development of Discrete-Steepest Descent Algorithms (D-SDA) as a heuristic for MINLP [9, 10, 425] shows the potential of techniques based on exploration of the discrete lattice based on discrete convex analysis tools, see *e.g.*, [426]. Using a similar algorithm as a heuristic but considering the disjunctive nature naturally expressed on a GDP problem allows for an efficient solution for these optimization problems. Preliminary results seem to confirm this hypothesis, leading to an exciting development for GDP solution methods.

11.4.8 Decomposition Methods for Quantum Optimization

As observed from the results in Chapters 9 and 10, the study of formulations for different constrained optimization problems is an area of research interest. These formulations need to be designed such that they avoid the limitations posed by the existing quantum hardware. As observed in this Thesis, one key limiting fact of the existing Quantum devices is their size. A way to circumvent the limitations posed by the reduced hardware size is decomposition methods. Problem decomposition, as approached in the first chapters of the Thesis when applied to convex MINLP problems, aims to break down the optimization into smaller subproblems that can be solved more efficiently using specialized means. Instead of relying on efficient MILP or NLP solvers, an interesting future line of research is one where the specialized means are quantum computers in hybrid quantum-classical algorithms. These decomposition methods could exploit the optimization problems' mathematical structure and harness parallelization and specialization when solving the original problem's partitions. By solving different optimization subproblems with specialized tools, separately treating the complexity that arises from nonlinearity and discreteness, for example, more complex discrete nonlinear optimization problems can be tackled using unconventional computing methods. This would allow us to take advantage of the potential speedups

that unconventional computational paradigms can provide (*e.g.,* Quantum Computing) when solving combinatorial quadratic problems[427]. Notably, few nonlinear discrete optimization problems can be represented in a small enough QUBO to fit the current hardware. Nevertheless, by strategically partitioning general MINLP problems, subproblems with combinatorial quadratic structure can be posed and efficiently solved using these unconventional computational paradigms within a decomposition solution framework. Moreover, the use of Lagrangean and Benders based-decomposition schemes [428, 429], tailored for problems with complicating constraints and variables respectively, for solving MINLP problems using quantum computers is an interesting avenue for future research. Even though the existing technology limits current Quantum Computing hardware, the same modeling paradigm for combinatorial optimization, based on the Ising spin model, can interface with other unconventional computational methods such as Coherent Ising[430] and Simulated Bifurcation Machines[400]. Combining expertise in modeling and algorithmic design would allow future researchers to address complex discrete nonlinear problems from practical application, even without requiring the development of large-scale fault-tolerant quantum computers.

Appendix A

Integer Programming Techniques for Minor Embedding in Quantum Annealers*

A.1 Introduction

Graph minor theory (GMT), the central theme of this work, is prominent across many fields. In quantum computing, GMT is employed to extend the scope of problems that can be represented on current quantum annealing hardware [431, 432]. Mapping a dense problem (logical) graph Y to a sparse (target) graph X can be achieved by constructing connected subgraphs of the target graph X from the high degree logical vertices y. The resulting mapping is called a minor-embedding of Y inside X.

Numerous heuristics for finding minor-embeddings have been proposed [433–435]. While these approaches are generally fast, they do not provide guarantees on the quality of the produced minor-embeddings nor can they prove the nonexistence of a minor-embedding for infeasible problems. An approach that attempts to address these shortcomings was recently introduced in Dridi, Alghassi, and Tayur [91]. This approach uses tools from algebraic geometry and produces an equational formulation (as opposed to a purely combinatorial

*Published as: David E Bernal, Kyle EC Booth, Raouf Dridi, Hedayat Alghassi, Sridhar Tayur, and Davide Venturelli. "Integer programming techniques for minor-embedding in quantum annealers". In: *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer. 2020, pp. 112–129.

approach) to the minor-embedding problem.

In this chapter, starting from this equational formulation, we propose integer programming (IP) techniques for tackling the embedding problem. Our proposed approaches differ from the computationally demanding Groebner bases computation used previously and are aimed at more efficiently computing embeddings while retaining the interesting properties that arise from the equational formulation of the problem. Our first approach, detailed in Section A.3, directly translates the previous equational formulation to IP, while our second approach decomposes the problem into an assignment master problem and fiber condition checking subproblems, as described in Section A.4. The proposed methods are able to detect instance infeasibility and provide bounds on solution quality, capabilities not offered by currently employed heuristic methods. While recent work uses an approach with integer programming to address the embedding problem based on templates specific to D-Wave quantum annealers [436], the techniques we present in this chapter are hardware agnostic.

We conduct an extensive empirical analysis involving a benchmark consisting of three different families of random graphs in Section A.5. There we present our results on an illustrative and challenging case for heuristics, which motivates the use of IP over Computational Algebraic Geometry (CAG) methods in random structured and unstructured graphs, and in applications for quantum annealing. The results of the experiments indicate that, while the IP-based methods are slower than currently employed heuristics whenever the heuristics are able to find an embedding, the IP methods provide infeasibility proofs and quality guarantees which the heuristics are unable to provide. Furthermore, comparing the monolithic IP against the decomposition, our experiments suggest that the decomposition results in a more efficient way to find concise embeddings. Depending on the tested instances the decomposition approach does not perform as efficiently as the monolithic IP approach in providing optimality or infeasibility guarantees, especially seen

in the illustrative example and small structured graphs. We provide concluding remarks in Section A.6.

Notations. All graphs considered in this chapter are simple and undirected. We use $\mathbf{V}(X)$ and $\mathbf{E}(X)$ to denote the vertex and edge sets of a graph X, respectively. We also define $n = |\mathbf{V}(X)|$, and $m = |\mathbf{V}(Y)|$. Finally, given a vector v, $\mathbf{v}$ denotes the concatenation $\mathbf{v} = (v_1, \cdots, v_{|\mathbf{v}|})$.

A.2 The equational model for embedding

Let X be a fixed target graph. A *minor-embedding* of the graph Y inside X, is a map ϕ from $\mathbf{V}(Y)$ to the set of connected subtrees of X, that satisfies the following condition: for each $(y_1, y_2) \in \mathbf{E}(Y)$, there exists at least one edge in $\mathbf{E}(X)$ connecting the two subtrees $\phi(y_1)$ and $\phi(y_2)$. The condition that each *vertex model* $\phi(y)$ is a connected subtree of X can be relaxed into $\phi(y)$ is a connected subgraph. In the literature, there is another but equivalent definition of minor-embedding in terms of deleting and collapsing the edges of X. This follows from the fact that, given a minor-embedding ϕ, the graph Y can be recovered from X by collapsing each set $\phi(y)$ (into the vertex y) and ignoring (deleting) all vertices of X that are not part of any of the subtrees $\phi(y)$. For the sake of a simple and clean terminology, we shall use the term embedding instead of minor-embedding throughout the remainder of the chapter. Suppose ϕ is an embedding of the graph Y inside the graph X. The subgraph of X given by $\phi(Y) := \cup_{y \in \mathbf{V}(Y)} \phi(y)$ is called a Y minor in GMT. In the context of quantum computations, it represents what the quantum processor sees since it does not distinguish between qubits representing different nodes of the logical graph or qubits representing the same node in the logical graph, *fibers*. In practice, quantum annealers use a strong ferromagnetic coefficient to enforce these replicated values to be equal, i.e., acting as a single qubit.

The hardware configuration of the D-Wave quantum annealers is described as a graph known as Chimera graph. The Chimera graph, $C_{L,M,N}$, is a grid of $M \times N$ cells of $K_{L,L}$ biclique graphs connected in a nearest-neighbor fashion by means of non-planar edges [366]. Figure A.1 presents the working graph of the D-Wave 2000Q Quantum Annealer located at NASA Ames Research Center. Specific heuristics for finding embedding inside Chimera graphs were developed in [435, 437–439] besides of the general embedding heuristics referenced above. A new generation of quantum annealers is in development, the D-Wave *Advantage*, using the *Pegasus* topology with increased connectivity [440] (Figure A.1, right). The *Pegasus* graph, $\mathcal{P}_{L,M,N,O}$, is composed of O layers of $M \times N$ grids of $K_{L,L}$ biclique graphs with additional edges within and among cells. The *Pegasus* topology has the number of layers fixed, $O = 3$, with $K_{4,4}$ cells, i.e. $L = 4$, with 4 additional edges each.

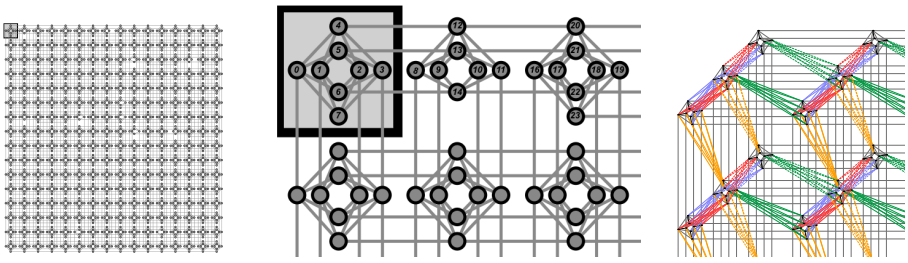


Figure A.1: Cross representation of D-Wave Systems 2000Q processor working graph corresponding to an incomplete Chimera graph $C_{4,16,16}$ (left and center) and Pegasus graph $\mathcal{P}_{4,2,2,3}$ (right).

In the equational approach, embedding the logical graph Y inside the target graph X, is represented by a surjective map $\pi : X \to Y$, which goes in the opposite direction of the map $\phi : Y \to X$, introduced earlier such that: $\pi^{-1}(y) = \phi(y)$. The map π is required to be surjective to guarantee that all logical qubits are embedded. In geometry, the subgraph $\pi^{-1}(y)$ is called the *fiber* at y of the *projection* π, and the mapping $\pi : X \to Y$ is a *fiber-bundle*. We can write:

$$\pi(x_i) = \sum_{j : y_j \in \mathbf{V}(Y)} \alpha_{ij} y_j, \quad \forall x_i \in \mathbf{V}(X) \tag{A.1}$$

where α_{ij} are binary coefficients. For this map to be well-defined we impose:

$$\sum_{j:y_j \in \mathbf{V}(Y)} \alpha_{ij} \leq 1 \quad \forall x_i \in \mathbf{V}(X), \tag{A.2}$$

that is, at most one α_{ij} is non-zero for each vertex in the graph X. The unique non zero α_{ij} (if any) represents whether the physical qubit x_i embeds y_j, i.e., $\phi(y_j) = x_i$. When all the coefficients α_{ij} are zero, we get $\pi(x_i) = 0$ indicating that the physical qubit is not used. In other words, while the domain of definition of π is $\mathbf{V}(X)$, its support is only a subset of $\mathbf{V}(X)$. The other conditions included in the definition of the embedding ϕ (e.g., the connectivity of the fibers) and the desired properties of such an embedding (e.g., the size of the fibers) can similarly be translated into equational form.

A.3 IP reformulation of polynomial equations

We tackle the problem of determining the mapping π using integer programming (IP). IP is a mathematical optimization technique used for problems modeled as a set of decision variables taking on integer values, constrained by linear constraints and looking to optimize a linear objective function. The standard solution approach to IP models is branch-and-bound tree search. Indeed, due to their many practical applications, the computational capabilities of modern IP solvers have increased tremendously in recent years [441]. These IP solvers are capable of proving instance infeasibility, and providing certificates of optimality and bounds on solution quality.

In our first approach, the previously proposed polynomial equations [91] are reformulated such that they represent the original logic and are representable in the IP formalism (i.e., linear constraints involving integer variables).

A.3.1 Constraints

Consider the mapping π given by Eqn. (A.1). In this section, we present the IP formulation of the polynomial conditions, from Dridi, Alghassi, and Tayur [91], that the coefficients $\alpha_{ij} \in \{0,1\} \quad \forall x_i \in \mathbf{V}(X), \forall y_j \in \mathbf{V}(Y)$ need to satisfy for π to be a valid embedding. This constitutes the first contribution of the chapter. Note that, with a slight abuse of notation, our IP approaches redefine α_{ij} as a binary decision variable equal to 1 if x_i belongs to the vertex model of y_j, and 0 otherwise.

1. Minimum and maximum size. These constraints ensure that the total number of qubits is bounded within the number of variables in the original problem and the total number of qubits n.

$$m \leq \sum_{i:x_i \in \mathbf{V}(X)} \sum_{j:y_j \in \mathbf{V}(Y)} \alpha_{ij} \leq n. \tag{A.3}$$

2. Well-definition of the map π. This is captured by Eq. (A.2).

3. Fiber size constraint. This constraint on the size of the vertex models $|\phi(y_j)|$, known as *fiber size*, is given by:

$$1 \leq \sum_{i:x_i \in \mathbf{V}(X)} \alpha_{ij} \leq k \quad \forall y_j \in \mathbf{V}(Y). \tag{A.4}$$

where k is the desired maximum size of each fiber $\pi^{-1}(y_j)$. The lower bound ensures that all the logical variables are embedded i.e., the map π is a surjection on the set $\mathbf{V}(X)$. We also include the following constraint:

$$1 \geq \alpha_{i_1 j} + \alpha_{i_2 j} \quad \forall x_{i_1}, x_{i_2} \in \mathbf{V}(X), \min d(x_{i_1}, x_{i_2}) > k, \forall y_j \in \mathbf{V}(Y). \tag{A.5}$$

This additional refinement excludes pairs x_{i_1} and x_{i_2} from being in the fiber $\pi^{-1}(y_j)$ whenever their distance, $d(x_{i_1}, x_{i_2})$, is larger than k, the desired maximum size of the fiber.

4. Fiber condition. We require that each fiber to be a connected subtree of X:

$$\forall x_{i_1}, x_{i_2} \in \pi^{-1}(y_j) : \alpha_{i_1 j} + \alpha_{i_2 j} + \left(\sum_{c_k(x_{i_1}, x_{i_2}) \in C_k(x_{i_1}, x_{i_2})} \left(\gamma_{c_k, j}\right) - 1 \right) \leq 2. \tag{A.6}$$

The binary variable $\gamma_{c_k,j}$ takes a value of 1 if a fiber $c_k(x_{i_1}, x_{i_2})$ is used in the vertex model of y_j, and 0 otherwise. Here $c_k(x_{i_1}, x_{i_2})$ is a fiber of size $\leq k$ connecting the two physical qubits x_{i_1} and x_{i_2}, and $\text{int}(c_k(x_{i_1}, x_{i_2})) = c_k(x_{i_1}, x_{i_2}) \backslash \{x_{i_1}, x_{i_2}\}$. We also write $C_k(x_{i_1}, x_{i_2})$ to denote the set of all fibers of size $\leq k$ connecting x_{i_1} and x_{i_2}. This condition implies the existence of a unique fiber connecting the pair and completely contained in $\pi^{-1}(y_j)$. This automatically implies that $\pi^{-1}(y_j)$ is connected. The binary $\gamma_{c_k,j}$ can be defined using the following IP representable constraints: for all $c_k(x_{i_1}, x_{i_2}) \in C_k(x_{i_1}, x_{i_2})$ and for all $y_j \in \mathbf{V}(Y)$:

$$\gamma_{c_k,j} = \prod_{\ell : x_\ell \in \text{int}(c_k(x_{i_1}, x_{i_2}))} \alpha_{\ell j} \Leftrightarrow \begin{cases} \gamma_{c_k,j} & \leq \alpha_{\ell j} \quad \forall x_\ell \in \text{int}(c_k(x_{i_1}, x_{i_2})) \\ \gamma_{c_k,j} & \geq 1 - (k-1) + \sum_{\ell : x_\ell \in \text{int}(c_k(x_{i_1}, x_{i_2}))} \alpha_{\ell j} \end{cases} \tag{A.7}$$

The constraint in Eq. (A.6) does not exclude the cases where 2 variables in the source graph $(y_{j_1}, y_{j_2}) \in \mathbf{E}(Y)$ are mapped to 4 different qubits in a fiber $\{x_{i_1}, \cdots, x_{i_4}\}$, where the vertex models are intercalated, i.e. $\phi(y_{j_1}) = \{x_{i_1}, x_{i_3}\}, \phi(y_{j_2}) = \{x_{i_2}, x_{i_4}\}$. The following constraint ensures that if two nodes in the target graph are in the vertex model of the same logical variable, and are not neighbors in the target graph, then one of the fibers joining them has to be active.

$$\begin{gathered} \alpha_{i_1 j} + \alpha_{i_2 j} - \sum_{c_k(x_{i_1}, x_{i_2}) \in C_k(x_{i_1}, x_{i_2})} \left(\gamma_{c_k, j}\right) \leq 1 \quad \forall y_j \in \mathbf{V}(Y) \\ \forall (x_{i_1}, x_{i_2}) \in \mathbf{V}(X), (x_{i_1}, x_{i_2}) \notin \mathbf{E}(X), \min d(x_{i_1}, x_{i_2}) \leq k \end{gathered} \tag{A.8}$$

5. Pullback condition. We require that for each edge (y_{j_1}, y_{i_2}) in $\mathbf{E}(Y)$, there exists at least one edge in $\mathbf{E}(X)$ connecting the fibers $\pi^{-1}(y_{j_1})$ and $\pi^{-1}(y_{i_2})$. The way we guarantee this is by requiring that the quadratic form of the logical graph y vanishes modulo the (pullback along π of the) quadratic form of the graph X. The details of this are in [91]. The resulting

constraint can be written as

$$1 \leq \sum_{i_1,i_2:(x_{i_1},x_{i_2})\in\mathbf{E}(X)} \left(\delta^{\parallel}_{i_1 i_2 j_1 j_2} + \delta^{\perp}_{i_1 i_2 j_1 j_2}\right) \quad \forall (y_{j_1}, y_{j_2}) \in \mathbf{E}(Y), \tag{A.9}$$

where we have introduced the binaries $\delta^{\parallel}_{i_1 i_2 j_1 j_2}$ and $\delta^{\perp}_{i_1 i_2 j_1 j_2}$ for all $(x_{i_1}, x_{i_2}) \in \mathbf{E}(X)$ and all $(y_{j_1}, y_{j_2}) \in \mathbf{E}(Y)$: The binary variable $\delta^{\parallel}_{i_1 i_2 j_1 j_2}$ is one if x_{i_1} and x_{i_2} are edges of the vertex-models $\phi(y_{j_1}), \phi(y_{j_2})$ respectively, and the binary variable $\delta^{\perp}_{i_1 i_2 j_1 j_2}$ is one if x_{i_2} and x_{i_1} are edges of the vertex-models $\phi(y_{j_1}), \phi(y_{j_2})$ respectively. This conditions are equivalent to $\delta^{\parallel}_{i_1 i_2 j_1 j_2} = \alpha_{i_1 j_1}\alpha_{i_2 j_2}$ and $\delta^{\perp}_{i_1 i_2 j_1 j_2} = \alpha_{i_1 j_2}\alpha_{i_2 j_1}$. We can then represent these new variables using linear inequalities as follows: $\forall (x_{i_1}, x_{i_2}) \in \mathbf{E}(X), \forall (y_{j_1}, y_{j_2}) \in \mathbf{E}(Y)$:

$$\delta^{\parallel}_{i_1 i_2 j_1 j_2} = \alpha_{i_1 j_1}\alpha_{i_2 j_2} \Leftrightarrow \begin{cases} \delta^{\parallel}_{i_1 i_2 j_1 j_2} & \leq \alpha_{i_1 j_1} \\ \delta^{\parallel}_{i_1 i_2 j_1 j_2} & \leq \alpha_{i_2 j_2} \\ \delta^{\parallel}_{i_1 i_2 j_1 j_2} & \geq \alpha_{i_1 j_1} + \alpha_{i_2 j_2} - 1 \end{cases}$$

and equivalently for $\delta^{\perp}_{i_1 i_2 j_1 j_2}$. Both variables cannot be one for a single combination of $(i_1 i_2 j_1 j_2)$ simultaneously. This leads to the following constraint.

$$\delta^{\parallel}_{i_1 i_2 j_1 j_2} + \delta^{\perp}_{i_1 i_2 j_1 j_2} \leq 1 \quad \forall (x_{i_1}, x_{i_2}) \in \mathbf{E}(X), \forall (y_{j_1}, y_{j_2}) \in \mathbf{E}(Y). \tag{A.10}$$

A.3.2 Complete IP model

The feasible region of the IP formulation is defined by:

$$F = \left\{(\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\delta}^{\parallel}, \boldsymbol{\delta}^{\perp}) | (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\delta}^{\parallel}, \boldsymbol{\delta}^{\perp}) \in ((\text{A.2}) \cap \cdots \cap (\text{A.10}))\right\}. \tag{A.11}$$

A constant objective function can be set for this problem such that any solution that lies within the feasible region defined in Eq. (A.11) optimizes it.

Embedding size. Another choice is given by the embedding size: Given the limitations on the available quantum annealing hardware in the size of available qubits, a desired property of an embedding is to have a small *qubit footprint*. The objective function in this case is encoded as

$$\min \sum_{i:x_i\in\mathbf{V}(X)} \sum_{y_j\in\mathbf{V}(Y)} \alpha_{ij} \quad \text{s.t. } (\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{\delta}^{\parallel}, \boldsymbol{\delta}^{\perp}) \in F. \tag{A.12}$$

Other objective functions such as fiber size minimization, minimal fiber size dispersion, small difference between a variable degree and fiber size, and available edges in the embedding are also IP representable and can be implemented within this framework.

A.4 Decomposition approach

Implementing all the constraints at once in the IP formulation leads to a model which is often intractable in practice. The fiber conditions require many constraints to be enforced, and only a small fraction of these are active in optimal solutions. We investigate the application of a decomposition approach which iterates between a qubit assignment master problem and fiber condition checking subproblems. The strategy adds strengthened 'no-good' constraints (i.e., cuts) to the master problem when they are found to be violated. Such an approach bears resemblance to decomposition techniques used for scheduling and routing problems, such as classical and logic-based Benders decomposition and branch-and-check [28, 429, 442].

A.4.1 Master Problem

In the master problem, we relax the fiber conditions, permitting a node in the logical graph to be mapped in multiple parts of the target graph without being connected. For our master problem, we introduce a new binary decision variable, $z_{e_x e_y} \quad \forall e_x \in \mathbf{E}(X), \forall e_y \in \mathbf{E}(Y)$, to track the embedding of problem edges in the target graph edges. The variable takes on a value of 1 if if edge $e_x = (x_{i_1}, x_{i_2}) : e_x \in \mathbf{E}(X)$ is mapped through the embedding in edge $e_y = (y_{j_1}, y_{j_2}) : e_y \in \mathbf{E}(Y)$, and 0 otherwise. For modeling purposes, we also denote $e_{x,1} = x_{i_1}, e_{x,2} = x_{i_2}, e_{y,1} = y_{j_1}$, and $e_{y,2} = y_{j_2}$. This master problem formulation includes previously expressed mapping constraints, Eq. (A.2), and size constraints in Eq. (A.4), in addition to

constraints (A.13) through (A.15) as follows:

Assignment of edges. Each edge in the source graph has to be assigned to an edge in the target graph.

$$\sum_{e_x \in \mathbf{E}(X)} z_{e_x e_y} = 1 \quad \forall e_y \in \mathbf{E}(Y). \tag{A.13}$$

Linking constraints. To link the assigned qubit values to the $z_{e_x e_y}$ variables, we use the following set of constraints $\forall e_x \in \mathbf{E}(X), \forall e_y \in \mathbf{E}(Y)$:

$$z_{e_x e_y} \leq \alpha_{e_{x,1} e_{x,2}} \quad z_{e_x e_y} \leq \alpha_{e_{y,1} e_{y,2}}. \tag{A.14}$$

Together, these constraints ensure that a problem edge can only be assigned to an edge in the target graph if the pair of nodes involved in that edge take on the required values, which are aggregated in the following constraint

$$2 \cdot z_{e_x e_y} \leq \alpha_{e_{x,1} e_{x,2}} + \alpha_{e_{y,1} e_{y,2}} \quad \forall e_x \in \mathbf{E}(X), \forall e_y \in \mathbf{E}(Y). \tag{A.15}$$

Subproblem relaxation. Although the constraints above already represent the assignment problem to be modeled in the master problem, we can include a relaxation of the subproblem to help guide to master problem towards feasible solutions. This requires the addition of another set of binary variables, w_j that track whether vertex model $\phi(y_j)$ has a size greater than one. Then, $\forall y_j \in \mathbf{V}(Y)$:

$$\sum_{i : x_i \in \mathbf{V}(X)} \alpha_{ij} - n \cdot w_j \leq 1, \tag{A.16a}$$

$$n(1 - \alpha_{ij}) + \sum_{\ell : (x_i, x_\ell) \in \mathbf{E}(X)} \alpha_{\ell j} + \sum_{\ell : (x_\ell, x_i) \in \mathbf{E}(X)} \alpha_{\ell j} \geq w_j \quad \forall x_i \in \mathbf{V}(X). \tag{A.16b}$$

This constraints ensure that the variable w_j is one if the node y_j is mapped to more than one node x_i. Eqs. (A.13)-(A.16), together with the cuts generated by the subproblems, define the master problem.

A.4.2 Subproblems

The subproblem validates if there exist vertices in the embedding belonging to the same vertex model $\phi(y_j)$ which are not connected in the target graph. If this is the case, it returns

a constraint that either: i) encourages connectivity in future iterations, or ii) removes occurrences of the disconnected vertex models from the graph. For each vertex model with more than one vertex on the embedding, it checks at each vertex on the target graph that belongs to that vertex model. If that vertex does not contain an edge that connects it to another vertex of that vertex model, then the checking procedure returns disconnected.

A.4.3 Cuts

If a particular vertex model is found to be disconnected in the solution, we add a constraint to remove the current solution and prevent future solutions from having the same disconnectivity. Let the set of disconnected vertices in the source graph be denoted as $\widehat{j} : y_{\widehat{j}} \in \widehat{Y}$. Let the set of vertices in the target graph that belong to this vertex model, $y_{\widehat{j}}$, in the current incumbent solution, be represented as the vertex model $\phi(y_{\widehat{j}}) \subseteq X$. Let the set of vertices that are adjacent to any vertex in $\phi(y_{\widehat{j}})$, but are not assigned value $y_{\widehat{j}}$, be denoted $\phi'(y_{\widehat{j}})$. The constraint generated in the current iteration for disconnected qubit $y_{\widehat{j}}$ is then given by:

$$\left(|\phi(y_{\widehat{j}})| - \sum_{i : x_i \in \phi(y_{\widehat{j}})} \alpha_{i\widehat{j}}\right) + \sum_{i : x_i \in \phi'(y_{\widehat{j}})} \alpha_{i\widehat{j}} \geq 1. \tag{A.17}$$

This removes the current infeasible solution from the search space and requires the master problem to: i) include at least one fewer vertex with this vertex model (bracketed term), or ii) include at least one more vertex with this vertex model, among the set of vertices that could improve connectivity (non-bracketed term).

Notice that we reformulated the pullback condition from the Eq. (A.9) in terms of $\boldsymbol{\delta}^{\|}$ and $\boldsymbol{\delta}^{\perp}$ into the variables $z_{e_x e_y}$ and its corresponding constraints, while the fiber condition is relaxed with the subproblem and cut generation procedure.

Following the intuition in [433], where the heuristic method tries to obtain embeddings with a small qubit footprint, the default objective function implemented in the master

problem is to minimize size. This objective leads the master problem to return compact assignments of variables. In the case that the feasibility objective is considered within this approach, the optimization procedure is stopped when the first feasible solution is found.

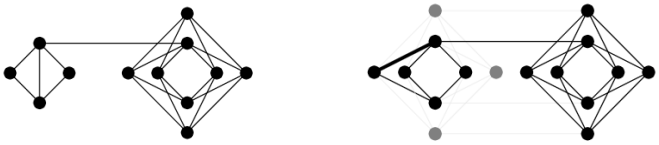
A.5 Results

The model in Section A.3 was implemented using the Python-package Pyomo [273], which interfaces with several solvers, including open-source and commercial solvers. The decomposition approach, presented in Section A.4, is implemented in C++ and uses the CPLEX 12.9 solver [52]. Our approaches are compared with the D-Wave default heurestic `minorminer`, introduced in [433]⋆. Unless otherwise stated, the monolithic IP method assumes a value of maximum fiber size $k = 3$, which is justified for the structured random graphs given their construction. This provides the monolithic method with an advantage with respect to the decomposition method given that the infeasibility proofs are contingent on the value of k. The results below were obtained using a laptop running Ubuntu 18.04 with an Intel Core i7-6820HQ CPU @ 2.7GHz with 8 threads and 16 GB of RAM.

A.5.1 Illustrative Example

This example is taken from [91], where a $K_{4,4}$ bipartite graph is connected through a single edge to a structured 4 nodes graph and is embedded in a $C_{4,1,2}$ chimera graph as seen in Figure A.2. This embedding is challenging for heuristic methods that search vertex models outside of the blocks [91]. The embedding with the minimal size is given when one of the nodes in the 4-node block is embedded in a chain of length 2, resulting in an embedding of length 13. The heuristic implemented in `minorminer` fails 50% of the times tried (1000),

⋆ github.com/dwavesystems/minorminer



(a) Source graph (b) Minimal size embedding

Figure A.2: Source graph of illustrative example [91] and its minimal size embedding in $C_{4,1,2}$. Grey nodes and edges represent unused nodes and edges in embedding, but present at the target graph. Bold edges represent edges in chains .

in the sense that it is not able to find a valid embedding in half of the experiments. We consider solving this problem using the CAG approach proposed in [91], by computing the Groebner basis of the polynomial ideal. When using the software Maple 2017 [443], which includes Faugère's algorithm [444, 445], the Groebner basis computation is unable to find a solution after 5 hours of computation before it runs out of memory. We apply our IP approach and include the open-source solvers GLPK 4.61 [446] and CBC 2.9.6 [447], and the commercial solvers Gurobi 8.1 [51] and CPLEX 12.9 [52]. Here we set a time limit of one minute per each run.

The open-source solvers fail to provide feasible solutions within the time limit when there is a constant objective function. CBC can find a solution when we consider the embedding size minimization as an objective, showing how the including an objective function can be beneficial for the IP solvers. In that case, the solver is unable to guarantee the optimality solution, although it finds the optimal solution, and it provides an optimality gap of 8.3%. The commercial solvers, on the other hand, can provide both feasible and optimal solutions in under a minute of computation. In particular, Gurobi takes 1.3 seconds to find a feasible solution and 31.2 seconds to find and prove the optimality of the solution while CPLEX takes 3.5 seconds and 9.4 seconds in the same tasks, respectively. As expected, the time required to provide a feasible solution is less than that to give optimality guarantees. Based

on the better results obtained using CPLEX compared to Gurobi, we only present the results using this solver in the remaining of the chapter. Apart from it, the decomposition approach can provide feasible solutions more efficiently, and with higher quality compared to the other approaches. In this case, it only took 0.4 seconds for the decomposition approach to provide a feasible solution which was nearly optimal, with a 7% optimality gap, and to find a provable optimal solution it required 46.5 seconds. These results suggest that the usage of commercial solvers is required for solving these challenging IP problems, and therefore the results presented in the remaining of this chapter correspond to these solvers.

A.5.2 Random Graphs

1. Random structured graphs. Here we generalize the example above. We consider the bipartite graph $K_{4,4}(p_{inter})$ parametrized by p_{inter}, which is the probability of the existence of edges between the two partitions. We randomly choose ζ edges, which we contract into nodes (each edge into a single node). This graph is then connected (attached) to a complete $K_{4,4}$ bipartite graph by 4 edges chosen with a probability p_{intra}. By construction, the resulting graph is a subgraph of $C_{4,1,2}$, and its size is $m+\zeta$. It is the smallest minor of the corresponding graph without contraction. The example of Section A.5.1 is obtained with $\zeta = 1$ (and $m = 12$).

Fixing $p_{inter} = p_{intra} = 0.5$, for each value of of contracted edges $\zeta \in \{0, \cdots, 3\}$, we generated 10 random graphs. These random graphs were embedded in a $C_{4,1,2}$ graph with a time limit of 300 seconds. Figure A.3a gives the runtimes for the monolithic IP and the decomposition methods solved using CPLEX. This figure also shows the boxplots for the 1000 runs of `minorminer`. For this case, given the way the random structured graphs are constructed, we see that the longest fiber will be at most of size 3, which we encode for the monolithic IP approach using the parameter $k = 3$. Notice that this observation biases the results

in favor of the monolithic IP approach with respect to the decomposition approach. For finding a feasible solution, the decomposition approach is more efficient than the monolithic IP approach. When $\zeta = 0$, where finding a feasible embedding is practically finding the minimally sized embedding, there is no difference in time performance between the cases of embedding size minimization and finding a feasible solution. For $\zeta > 0$, the embedding size minimization becomes more expensive, in particular for the decomposition approach. The monolithic IP and the decomposition approaches were able to find smaller or equally sized embeddings for 33 and 30 cases out of the 40 experiments, respectively. When the objective function is the size minimization, this number increased for all instances in all cases and is strictly better in 22 cases for the monolithic IP and 21 cases for the decomposition approach. Notice that the monolithic IP approach was able to find an embedding for one instance which was smaller than any of the 1000 runs of the heuristic method.

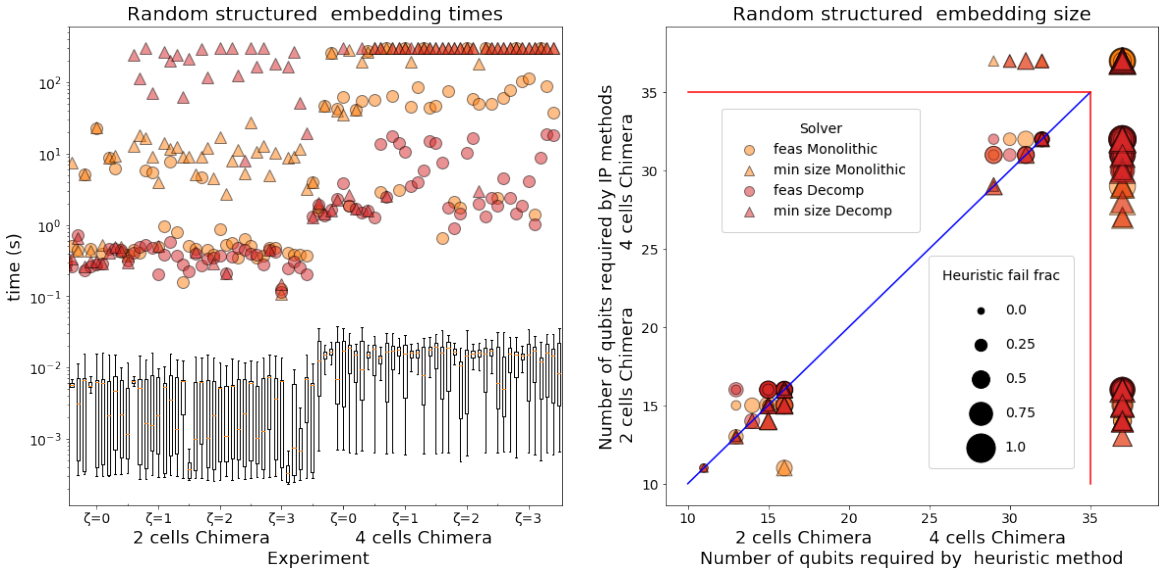
Larger instances of random structures graphs can be generated by combining two graphs like the ones described above, and include the edges appearing in a $C_{4,2,2}$ graph between the cells with probability p_{intra}. As before, we generated 10 random instances with values of $\zeta \in \{0, \cdots, 3\}$. The time performance of the different methods is shown in Figure A.3a. Out of the 40 instances the monolithic IP and the decomposition method are still able to find embeddings as succinct as the median heuristic behavior in 17 and 23 instances when trying to find a feasible solution, and in 20 and 17 instances when trying to minimize the size of the embedding, respectively. As in the previous case, the monolithic IP approach is able to find smaller embeddings than any of the 1000 runs of the heuristic method, although for this family of instances this happens for two cases.

Figure A.3b shows a comparison of the embedding median sizes obtained by the heuristic method versus the ones obtained for the IP methods. The size of the makers represents the heuristic failure rate fraction, computed from the 1000 runs of the heuristic method for

each instance. Both the monolithic and the decomposition approaches (different colors) and the feasibility and size minimization objectives (different markers) are represented in this figure. In total, out of the 80 structured instances, the heuristic failed more than 50% for 47 instances, more than 80% for 12 instances, and more than 90% for 3 instances. These instances appear on the right size of Figure A.3b. For those instances, the IP approaches were able to find a feasible solution in less than 20 seconds, and only for 19, 7, and 2 instances, respectively, the optimal solution could not be guaranteed within the time limit. For high failure rate problems (> 50% failure rate) for the heuristic, our methods find a feasible solution in under 20 seconds and prove optimality in more than half of the instances in less than 5 minutes. Notice that most of the runs corresponding to finding a feasible embedding (circles) are above the diagonal line, indicating a larger embedding size for the IP methods compared to the heuristic embeddings, while size minimization runs (triangles) lie on the diagonal or below it.

2. *Erdös-Rényi graphs.* These graphs are parametrized by the number of nodes v and the probability of an edge existing between each pair of nodes p. We consider a set of 10 random instances for each combination of $v \in \{5, 6, \cdots, 16\}$ and $p \in \{0.3, 0.5, 0.7\}$. Each of this graphs is embedded in different sizes of Chimera, $C_{4,1,1}, C_{4,2,1}, C_{4,3,1}$, and $C_{4,2,2}$, and Pegasus, $\mathcal{P}_{4,1,1,1}$ and $\mathcal{P}_{4,2,2,3}$. We set the time limit to 60 seconds. In the trivially infeasible case where $v > n$ our methods could almost immediately identify the infeasibility, contrary to the `minorminer` heuristic. The conclusion is that the runtime for the monolithic IP methods increases with the size of the target and source graphs, the density of the source graph given by p, and when the objective function is to minimize the embedding sizes.

In this experiment, we considered 2160 instances. In 1100 of them, the heuristic method could not find any feasible embedding after 1000 runs. Expectedly, the number of infeasible embeddings significantly drops when using the Pegasus graph. In 94% of these cases, at



(a) Embedding time

(b) Embedding size

Figure A.3: Embedding time and size comparison for different embedding methods for structured random graphs in $C_{4,1,2}$ and $C_{4,2,2}$ with respect to median behavior of `minorminer`. Values beyond the red lines represent embeddings where the heuristic median performance (right) or the IP methods (top) failed to return an embedding.

least one of the IP methods does not time out, meaning that the methods could prove the infeasibility of the embedding or find a feasible embedding. This proves that the methods proposed in this work are valid for providing guarantees of embeddability of graph minors in cases where the current heuristics are unable to answer this satifiability question.

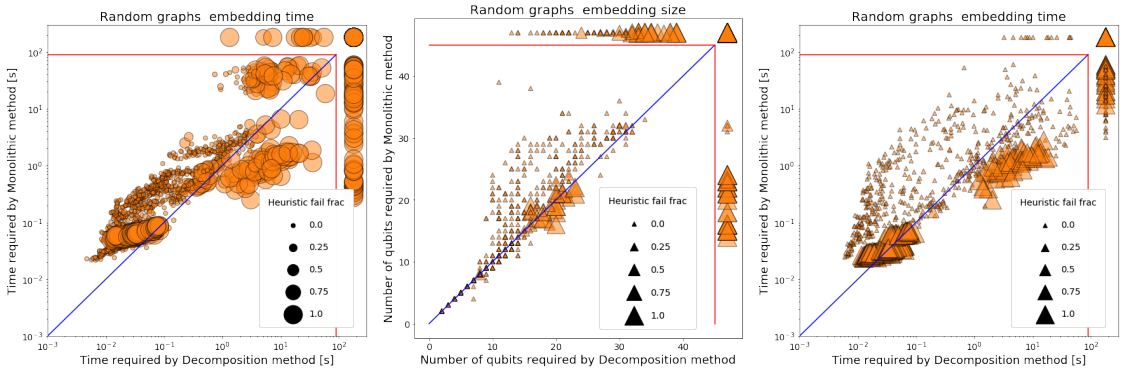
We complete our benchmark of random graphs embedding larger problems. In this case we, consider 5 random instances for each combination of $v \in \{10, 15, \cdots, 35\}$ and $p \in \{0.1, 0.3, 0.5, 0.7\}$ embedded into $C_{4,4,4}$, where the longest fiber size was increased to $k = 5$. We observe that for these instances, the only IP solver that does not run out time is CPLEX implementing either the monolithic IP or the decomposition approach with the feasible solution objective.

Figure A.4 presents the embedding size and time comparison for the small random

graph experiments. For this test-case, in 60% of the instances, the decomposition approach yielded embeddings with sizes equal or smaller than the median of the ones returned by the heuristic, when looking for a feasible solution, and in 90% of the instances when minimizing the embedding size. The monolithic IP approach was more efficient to declare infeasibility in non-trivial cases ($m < n$) than the decomposition approach, Figures A.4a and A.4b, as the values below the diagonal with large heuristic failure fractions and longer runtimes. When compared to the minimal size found after the 1000 runs of the heuristic method, the monolithic IP methods are still able to find smaller embeddings for around 5% of the cases. The comparison in Figure A.4 highlights that the sizes of the embeddings found by the decomposition approach are in most cases as small or smaller than the monolithic IP approach. In terms of the computational time, we observe that those instances that were challenging for the heuristic (large markers) are more easily solved by the decomposition approach, especially when minimizing the size of the embedding. The remaining instances appear to be solved more efficiently using the monolithic IP approach. The larger and more challenging instances lead to different results. Out of the 120 instances solved, the decomposition approach behaves better than the monolithic IP approaches, obtaining equally good or better embedding than the median heuristic behavior in 30% of the cases, compared to around 10% for the monolithic IP approaches. The solution requires larger fibers, which affected directly the formulation size of the monolithic case making it more challenging to solve. Only for one instance, a smaller embedding than any of the observed heuristic solutions is obtained, in this case by the decomposition approach.

A.5.3 Applications

1. Gadgets. It has been shown in [448], that all the cubic gadgets can be embedded in a single cell of either Chimera of Pegasus graphs, but three of the quartic gadgets required



(a) Feasibility: Time (b) Minimize size: Size (c) Minimize size: Time

Figure A.4: Embedding size and time comparison for Erdös-Rényi graphs ($v \in \{5,6,\cdots,16\}$, $p \in \{0.3,0.5,0.7\}$) given different objectives. Values beyond the red lines represent embedding where the decomposition (right) or the monolithic IP methods (top) failed to return an embedding or went over the time limit.

more than a single Chimera cell. The three gadgets were $K_6 - e$, *Double* K_4, and K_6. We find more efficient embeddings for several of the gadgets, namely K_5 (with 2 and 1 auxiliary variables), K_6, and $K_6 - e$ compared to [448]. The embeddings found can be guaranteed to be the minimal size within a few seconds of computation. For the case of the quartic gadgets, all but one (double K_4) could be embedded in a single Chimera cell, in which case our method could provide infeasibility guarantees in less than 10 seconds.

2. Spanning tree. An example of an application is the communication of vehicles/agents with a central control station that can be disrupted in a particular area and can be routed through Δ agents/vehicles. Finding the communication routing of the vehicles that minimizes the distance, is equivalent to finding the minimum spanning tree with bounded degree Δ. Rieffel et al. [449] propose three different formulations of this problem that can be embedded in a quantum annealer. Given the graph defined by the agents/vehicles $S = (V,E)$, the distances among them might change but not the graph itself. At the same time, the degree of the spanning tree Δ is fixed by the communication equipment. We generate 80

instances with graphs S with 4 vertices and between 3 and 5 edges. When reformulating the problems as QUBOs, we obtain instances where the target graph ranges in size between 19 and 29 nodes, and with 35 to 70 edges. The resulting QUBOs were embedded using the decomposition approach and compared to the heuristic in `minorminer`. We obtain smaller (or equally sized) embeddings than the median length of the heuristic in 12.5% (15%) of the instances in 5 minutes of computational time (compared to 1000 runs of the heuristic).

3. Protein folding. Perdomo-Ortiz et al. [450] encode the different configuration of the amino acids in a protein in terms of a QUBO representing the overall energy of the system. Minimizing this QUBO with respect to the different number of bonds between amino-acids would yield the protein's least-energy configuration. [450] shows not only the encoding of the problem as a QUBO, but also provides a custom algorithm to embed it in the D-Wave One chip, which has hardware described by a faulty $C_{4,4,4}$ graph. The largest instance solved in this chapter involved embedding a QUBO of 19 variables in a target graph of 127 qubits. The resulting embedding involved 81 qubits with the largest vertex model of length 5, being at the time the largest problem embedded and solved in D-Wave's quantum annealers. We highlight two qualities of our approach: 1) we are not making any assumption about the source or target graphs, allowing us to work with faulty Chimera graphs as targets; and 2) we can exploit the fact of having an existing embedding to initialize our procedures, allowing us to solve our IP problems more efficiently. Initializing with the embedding provided by Perdomo-Ortiz et al. [450] while restricting the $k = 5$ in the monolithic IP approach, we find an embedding of length 77 (4.9% qubit footprint reduction) within an hour of computation. Allowing larger fibers, we find an embedding of size 74 (8.6% qubit footprint reduction) with a fiber of size 6. These embeddings are not guaranteed to be optimal, but in both cases improve those previously found.

A.6 Conclusions

Integer programming (IP) approaches are proposed to solve the graph minor-embedding problem. Specifically, we develop a monolithic IP derived from the polynomial equations presented in Dridi, Alghassi, and Tayur [91], and a decomposition approach, both of which are capable of identifying infeasible instances and providing bounds on solution quality. These approaches are also agnostic of the source and target graphs. Both approaches were implemented and tested using a range of different source graphs with various sizes, densities, and structures. The target graphs used follow the architecture of the chips in D-Wave's current and future quantum annealers. Although slower overall than the currently-employed heuristic method [433], the proposed methods prove to be a viable solution approach for highly structured source graphs, where the heuristic fails with a higher probability.

The results presented in this chapter highlight the more general approaches to minor-embedding using IP. Another way of obtaining better performance is by reducing the search space by imposing certain limitations to the embedding, e.g. by allowing only certain topologies for the vertex models or by fixing certain embedding characteristics, like maximum fiber size. Initial attempts to include these approximations show a promising decrease in the computation time with an acceptable trade-off in quality. Our formulation and results are a baseline for future methods that can work at application-scale. A promising future direction is to use symmetries and the invariant formulation as previously suggested [91]. Finally, applications in gadget embeddings, spanning tree problems, and protein folding demonstrate the advantages of our approaches.

Crash course on Groebner bases

For completeness, we include the following introductory material adopted from [91, 451]. First, the notations: We write $\mathbb{Q}[x_0,\ldots,x_{n-1}]$ for the ring of polynomials in $x_0,\ldots,x_{n-1}$ with rational coefficients. Let $\mathcal{S}$ be a set of polynomials $f \in \mathbb{Q}[x_0,\ldots,x_{n-1}]$. Let $\mathcal{V}(S)$ denotes the algebraic variety defined by the polynomials $f \in S$, that is, the set of common zeros of the equations $f = 0$, $f \in \mathcal{S}$. The system $\mathcal{S}$ generates an *ideal* $\mathcal{I}$ by taking all linear combinations over $\mathbb{Q}[x_0,\ldots,x_{n-1}]$ of all polynomials in $\mathcal{S}$; we have $\mathcal{V}(\mathcal{S}) = \mathcal{V}(\mathcal{I})$. The ideal $\mathcal{I}$ reveals the hidden polynomials that are the consequence of the generating polynomials in $\mathcal{S}$. For instance, if one of the hidden polynomials is the constant polynomial 1 (i.e., $1 \in \mathcal{I}$), then the system $\mathcal{S}$ is inconsistent (because $1 \neq 0$). To be precise, the set of all hidden polynomials is given by the so-called *radical ideal* $\sqrt{\mathcal{I}}$, which is defined by $\sqrt{\mathcal{I}} = \{g \in \mathbb{Q}[x_1,\ldots,x_n] | \exists r \in \mathbb{N} : g^r \in \mathcal{I}\}$. We have $\mathcal{I}(\mathcal{V}(\mathcal{I})) = \sqrt{\mathcal{I}}$. Of course, the radical ideal $\sqrt{\mathcal{I}}$ is infinite. Luckily, thanks to a prominent technical result (i.e., *Dickson's lemma*), it has a finite generating set i.e., a *Groebner basis* $\mathcal{B}$, which one might take to be a triangularization of the ideal $\sqrt{\mathcal{I}}$. The computation of Groebner bases generalizes Gaussian elimination in linear systems. We also continue to have $\mathcal{V}(\mathcal{S}) = \mathcal{V}(\mathcal{I}) = \mathcal{V}(\sqrt{\mathcal{I}}) = \mathcal{V}(\mathcal{B})$. Instead of giving the technical definition of what a Groebner basis is (which can be found in [452] and in many other textbooks) let us give an example (for simplicity, we use the term "Groebner bases" to refer to *reduced* Groebner bases, which is, technically what we are working with):

Example A..1. Consider the system by

$$\mathcal{S} = \{x^2 + y^2 + z^2 - 4, x^2 + 2y^2 - 5, xz - 1\}.$$

We want to solve $\mathcal{S}$. One way to do so is to compute a Groebner basis for $\mathcal{S}$. In Figure A.5, the output of cell number 4 gives a Groebner basis of $\mathcal{S}$. We can see that the initial system has been triangulized: The last equation contains only the variable z, whilst the second has

an additional variable, and so on. The variable z is said to be eliminated with respect to the rest of the variables. When computing the Groebner basis, the underlying algorithm (Buchberger's algorithm) uses the ordering $x > y > z$ (called lexicographical ordering) for the computing of two internal calculations: cross-multiplications and Euclidean divisions. The program tries to isolate z first, then z and y, and finally x, y, and z (all variables). Different orderings yield different Groebner bases.

```
In [1]: from sympy import *

        x, y, z = symbols('x y z')

In [2]: eqs = [x**2+y**2+z**2-4, x**2+2*y**2-5, x*z-1]

In [3]: # to compute a Groebner basis we need a "monomial order" = ordering relation on the variables x, y and z

        result = groebner(eqs,  x,y, z,  order='lex')

In [4]: for eq in list(result):
            print eq
            print "\n"

x + 2*z**3 - 3*z


y**2 - z**2 - 1


2*z**4 - 3*z**2 + 1
```

Figure A.5: Jupyter notebook for computing Groebner bases using Python package sympy. More efficient algorithms exist (e.g., [444, 445]).

The mathematical power of Groebner bases doesn't stop at solving systems of algebraic equations. The applicability of Groebner bases goes well beyond this: it gives necessary and sufficient conditions for the existence of solutions. For instance, if the ideal represents a system of algebraic equations and these equations are (algebraically) dependent on certain parameters, then the intersection (A.23) gives *all* necessary and sufficient conditions for the existence of solutions. The following embedding example makes this more concrete. The

example also illustrates the original equational approach [91].

Example A..2. Consider the two graphs in Figure A.6. We would like to determine all embeddings $\pi : X \to Y$. In this case, well-definition equations (I) are given by

$$\alpha_{1,1}\alpha_{1,2}, \alpha_{1,1}\alpha_{1,3}, \alpha_{1,2}\alpha_{1,3}, \tag{A.18}$$

$$\alpha_{2,1}\alpha_{2,2}, \alpha_{2,1}\alpha_{2,3}, \alpha_{2,2}\alpha_{2,3}, \tag{A.19}$$

$$\alpha_{3,1}\alpha_{3,2}, \alpha_{3,1}\alpha_{3,3}, \alpha_{3,2}\alpha_{3,3}, \tag{A.20}$$

$$\alpha_{4,1}\alpha_{4,2}, \alpha_{4,1}\alpha_{4,3}, \alpha_{4,2}\alpha_{4,3}, \tag{A.21}$$

$$\alpha_{5,1}\alpha_{5,2}, \alpha_{5,1}\alpha_{5,3}, \alpha_{5,2}\alpha_{5,3}, \tag{A.22}$$

and

$$\alpha_{1,1}+\alpha_{1,2}+\alpha_{1,3}-\beta_1, \quad \alpha_{2,1}+\alpha_{2,2}+\alpha_{2,3}-\beta_2, \quad \alpha_{3,1}+\alpha_{3,2}+\alpha_{3,3}-\beta_3,$$

$$\alpha_{4,1}+\alpha_{4,2}+\alpha_{4,3}-\beta_4, \quad \alpha_{5,1}+\alpha_{5,2}+\alpha_{5,3}-\beta_5.$$

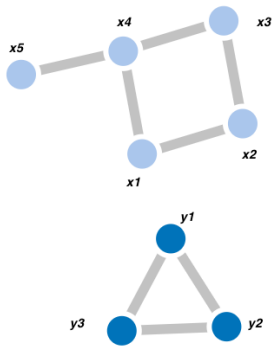


Figure A.6: The set of *all* fiber bundles $\pi : X \to Y$ defines an algebraic variety. This variety is given by the Groebner basis (A.23).

The pullback condition reads

$$-1+\alpha_{4,1}\alpha_{5,2}+\alpha_{3,1}\alpha_{4,2}+\alpha_{1,1}\alpha_{2,2}+\alpha_{3,2}\alpha_{4,1}+\alpha_{1,2}\alpha_{2,1}+\alpha_{1,2}\alpha_{4,1}+\alpha_{2,2}\alpha_{3,1}+\alpha_{1,1}\alpha_{4,2}+\alpha_{2,1}\alpha_{3,2}+\alpha_{4,2}\alpha_{5,1},$$

$$-1+\alpha_{3,3}\alpha_{4,1}+\alpha_{1,3}\alpha_{2,1}+\alpha_{2,3}\alpha_{3,1}+\alpha_{4,1}\alpha_{5,3}+\alpha_{1,3}\alpha_{4,1}+\alpha_{1,1}\alpha_{2,3}+\alpha_{4,3}\alpha_{5,1}+\alpha_{2,1}\alpha_{3,3}+\alpha_{3,1}\alpha_{4,3}+\alpha_{1,1}\alpha_{4,3},$$

$$-1+\alpha_{3,3}\alpha_{4,2}+\alpha_{1,2}\alpha_{2,3}+\alpha_{1,2}\alpha_{4,3}+\alpha_{1,3}\alpha_{2,2}+\alpha_{1,3}\alpha_{4,2}+\alpha_{2,3}\alpha_{3,2}+\alpha_{2,2}\alpha_{3,3}+\alpha_{4,2}\alpha_{5,3}+\alpha_{3,2}\alpha_{4,3}+\alpha_{4,3}\alpha_{5,2}.$$

Finally, the connected fiber condition is given by

$$-\alpha_{1,1}\alpha_{2,1}\alpha_{5,1},-\alpha_{1,1}\alpha_{3,1}\alpha_{5,1},-\alpha_{1,2}\alpha_{2,2}\alpha_{5,2},-\alpha_{1,2}\alpha_{3,2}\alpha_{5,2},-\alpha_{1,3}\alpha_{2,3}\alpha_{5,3},-\alpha_{1,3}\alpha_{3,3}\alpha_{5,3}$$

$$-\alpha_{2,1}\alpha_{3,1}\alpha_{5,1},-\alpha_{2,1}\alpha_{4,1}\alpha_{5,1},-\alpha_{2,2}\alpha_{3,2}\alpha_{5,2},-\alpha_{2,2}\alpha_{4,2}\alpha_{5,2},-\alpha_{2,3}\alpha_{3,3}\alpha_{5,3},-\alpha_{2,3}\alpha_{4,3}\alpha_{5,3},$$

$$\alpha_{2,1}\alpha_{5,1},\alpha_{2,2}\alpha_{5,2},\alpha_{2,3}\alpha_{5,3}.$$

The reduced Groebner basis of the resulted system (computed using the ordering $\alpha \succ \beta$) is given by

$$\begin{aligned}\mathcal{B} \;=\; &\big\{\beta_1-1,\beta_2-1,\beta_3-1,\beta_4-1,\beta_i^2-\beta_i,\,\alpha_{ij}^2-\alpha_{ij},\\
&\alpha_{1,2}\alpha_{1,3},\alpha_{1,2}\alpha_{3,2},\alpha_{1,3}\alpha_{3,3},\alpha_{2,2}\alpha_{2,3},\alpha_{2,2}\alpha_{4,2},\alpha_{2,2}\alpha_{5,2},\alpha_{2,3}\alpha_{4,3},\alpha_{2,3}\alpha_{5,3},\alpha_{3,2}\alpha_{3,3},\alpha_{4,2}\alpha_{4,3},\\
&\alpha_{4,2}\alpha_{5,3},\alpha_{4,3}\alpha_{5,2},\alpha_{5,2}\alpha_{5,3},\alpha_{4,2}\alpha_{5,2}-\alpha_{5,2},\alpha_{4,2}\beta_5-\alpha_{5,2},\alpha_{4,3}\alpha_{5,3}-\alpha_{5,3},\\
&\quad\vdots\\
&-\alpha_{2,2}\alpha_{5,3}-\alpha_{3,2}\alpha_{5,3}+\alpha_{1,2}\beta_5+\alpha_{2,2}\beta_5+\alpha_{3,2}\beta_5+\alpha_{3,3}\beta_5+\alpha_{5,2}+\alpha_{5,3}-\beta_5\,\big\}.\end{aligned}$$

In particular, the intersection $\mathcal{B}\cap\mathbb{Q}[\beta] = (\beta_1-1,\beta_2-1,\beta_3-1,\beta_4-1,{\beta_5}^2-\beta_5)$ gives the two Y minors (i.e., subgraphs X^β) inside X. The remainder of $\mathcal{B}$ gives the explicit expressions of the corresponding mappings.

This feature of Groebner bases can be made more precise as follows:

Theorem A..1. Let $\mathcal{I}\subset\mathbb{Q}[x_0,\ldots,x_{n-1}]$ be an ideal, and let $\mathcal{B}$ be a reduced Groebnber basis of $\mathcal{I}$ with respect to the lex order $x_0\succ\ldots\succ x_{n-1}$. Then, for every $0\le l\le n-1$, the set

$$\mathcal{B}\cap\mathbb{Q}[x_l,\ldots,x_{n-1}] \tag{A.23}$$

is a Groebner basis of the ideal $\mathcal{I} \cap Q[x_l, \ldots, x_{n-1}]$.

Let us conclude by explaining how the number of solutions of an algebraic systems $\mathcal{I} \subset \mathbb{Q}[x_0, \cdots, x_{n-1}]$ can be read from the Groebner basis. This is done using *staircase diagrams*, as follows. To each polynomial in $\mathcal{I}$ we assign a point in the Euclidean space $\mathbb{E}^n$ given by the exponents of its leading term (with respect to the given monomial order). The key idea is given by the following proposition:

Proposition A..1. The ideal $\mathcal{I} \subset \mathbb{Q}[x_0, \cdots, x_{n-1}]$ is zero dimensional if and only if the number of points under the shaded region of its staircase is finite, and this number is equal to the dimension of the quotient $Q[x_0, \cdots, x_{n-1}]/\mathcal{I}$, that is, the number of zeros of $\mathcal{I}$.

Appendix B

Other Appendices

B.A Problem Classification for Convex MINLP Review

The following table contains the problem names and their classifications with regards to the different benchmark sets in Section B.A. Correlation between the categories are shown in Figure B.1.

Instance name	Relaxation gap	cat.	Nonlinearity measure	cat.	Discreteness measure	cat.
alan	0.89 %	low	38 %	low	50 %	high
ball_mk2_10	–	high	100 %	high	100 %	high
ball_mk2_30	–	high	100 %	high	100 %	high
ball_mk3_10	–	high	100 %	high	100 %	high
ball_mk3_20	–	high	100 %	high	100 %	high
ball_mk3_30	–	high	100 %	high	100 %	high
ball_mk4_05	–	high	100 %	high	100 %	high
ball_mk4_10	–	high	100 %	high	100 %	high
ball_mk4_15	–	high	100 %	high	100 %	high
batch	9.2 %	low	48 %	low	52 %	high
batch0812	5.6 %	low	40 %	low	60 %	high
batchdes	3.9 %	low	53 %	high	47 %	low
batchs101006m	4.5 %	low	18 %	low	46 %	low
batchs121208m	3.1 %	low	15 %	low	50 %	high
batchs151208m	2.8 %	low	14 %	low	46 %	low
batchs201210m	1.7 %	low	12 %	low	45 %	low
clay0203h	100 %	high	20 %	low	20 %	low
clay0203m	100 %	high	20 %	low	60 %	high
clay0204h	100 %	high	15 %	low	20 %	low
clay0204m	100 %	high	15 %	low	62 %	high
clay0205h	100 %	high	12 %	low	19 %	low
clay0205m	100 %	high	13 %	low	63 %	high
clay0303h	100 %	high	27 %	low	21 %	low
clay0303m	100 %	high	18 %	low	64 %	high
clay0304h	100 %	high	20 %	low	20 %	low

Instance name	Relaxation gap	cat.	Nonlinearity measure	cat.	Discreteness measure	cat.
clay0304m	100 %	high	14 %	low	64 %	high
clay0305h	100 %	high	16 %	low	20 %	low
clay0305m	100 %	high	12 %	low	65 %	high
cvxnonsep_normcon20	0.34 %	low	100 %	high	50 %	high
cvxnonsep_normcon20r	0.34 %	low	50 %	high	25 %	low
cvxnonsep_normcon30	0.54 %	low	100 %	high	50 %	high
cvxnonsep_normcon30r	0.54 %	low	50 %	high	25 %	low
cvxnonsep_normcon40	0.78 %	low	100 %	high	50 %	high
cvxnonsep_normcon40r	0.78 %	low	50 %	high	25 %	low
cvxnonsep_nsig20	0.16 %	low	100 %	high	50 %	high
cvxnonsep_nsig20r	0.16 %	low	50 %	high	25 %	low
cvxnonsep_nsig30	0.11 %	low	100 %	high	50 %	high
cvxnonsep_nsig30r	0.12 %	low	50 %	high	25 %	low
cvxnonsep_nsig40	0.16 %	low	100 %	high	50 %	high
cvxnonsep_nsig40r	0.16 %	low	50 %	high	25 %	low
cvxnonsep_pcon20	0.55 %	low	100 %	high	50 %	high
cvxnonsep_pcon20r	0.55 %	low	51 %	high	26 %	low
cvxnonsep_pcon30	0.47 %	low	100 %	high	50 %	high
cvxnonsep_pcon30r	0.47 %	low	51 %	high	25 %	low
cvxnonsep_pcon40	0.39 %	low	100 %	high	50 %	high
cvxnonsep_pcon40r	0.39 %	low	51 %	high	25 %	low
cvxnonsep_psig20	0.08 %	low	100 %	high	50 %	high
cvxnonsep_psig20r	0.09 %	low	50 %	high	24 %	low
cvxnonsep_psig30	0.32 %	low	100 %	high	50 %	high
cvxnonsep_psig30r	0.32 %	low	50 %	high	24 %	low
cvxnonsep_psig40	0.34 %	low	100 %	high	50 %	high
cvxnonsep_psig40r	0.29 %	low	50 %	high	24 %	low
du-opt	1.2 %	low	100 %	high	65 %	high
du-opt5	51 %	high	100 %	high	65 %	high
enpro48pb	6.9 %	low	19 %	low	60 %	high
enpro56pb	15 %	low	19 %	low	57 %	high
ex1223	15 %	low	64 %	high	36 %	low
ex1223a	2.0 %	low	43 %	low	57 %	high
ex1223b	15 %	low	100 %	high	57 %	high
ex4	104 %	high	14 %	low	69 %	high
fac1	0.11 %	low	73 %	high	27 %	low
fac2	23 %	low	82 %	high	18 %	low
fac3	30 %	low	82 %	high	18 %	low
flay02h	25 %	low	4.3 %	low	8.7 %	low
flay02m	25 %	low	14 %	low	29 %	low
flay03h	37 %	low	2.5 %	low	10 %	low
flay03m	37 %	low	12 %	low	46 %	low
flay04h	43 %	low	1.7 %	low	10 %	low
flay04m	43 %	low	10 %	low	57 %	high
flay05h	46 %	low	1.3 %	low	10 %	low

Instance name	Relaxation gap	cat.	Nonlinearity measure	cat.	Discreteness measure	cat.
flay05m	46 %	low	8.1 %	low	65 %	high
flay06h	48 %	low	1.1 %	low	11 %	low
flay06m	48 %	low	7.0 %	low	70 %	high
fo7	100 %	high	12 %	low	37 %	low
fo7_2	100 %	high	12 %	low	37 %	low
fo7_ar2_1	100 %	high	13 %	low	38 %	low
fo7_ar25_1	100 %	high	13 %	low	38 %	low
fo7_ar3_1	100 %	high	13 %	low	38 %	low
fo7_ar4_1	100 %	high	13 %	low	38 %	low
fo7_ar5_1	100 %	high	13 %	low	38 %	low
fo8	100 %	high	11 %	low	38 %	low
fo8_ar2_1	100 %	high	11 %	low	39 %	low
fo8_ar25_1	100 %	high	11 %	low	39 %	low
fo8_ar3_1	100 %	high	11 %	low	39 %	low
fo8_ar4_1	100 %	high	11 %	low	39 %	low
fo8_ar5_1	100 %	high	11 %	low	39 %	low
fo9	100 %	high	10 %	low	40 %	low
fo9_ar2_1	100 %	high	10 %	low	40 %	low
fo9_ar25_1	100 %	high	10 %	low	40 %	low
fo9_ar3_1	100 %	high	10 %	low	40 %	low
fo9_ar4_1	100 %	high	10 %	low	40 %	low
fo9_ar5_1	100 %	high	10 %	low	40 %	low
gams01	97 %	high	22 %	low	76 %	high
gbd	0.00 %	low	25 %	low	75 %	high
hybriddynamic_fixed	10 %	low	15 %	low	14 %	low
ibs2	0.22 %	low	100 %	high	50 %	high
jit1	0.37 %	low	48 %	low	16 %	low
m3	100 %	high	23 %	low	23 %	low
m6	100 %	high	14 %	low	35 %	low
m7	100 %	high	12 %	low	37 %	low
m7_ar2_1	100 %	high	13 %	low	38 %	low
m7_ar25_1	100 %	high	13 %	low	38 %	low
m7_ar3_1	100 %	high	13 %	low	38 %	low
m7_ar4_1	100 %	high	13 %	low	38 %	low
m7_ar5_1	100 %	high	13 %	low	38 %	low
meanvarx	0.41 %	low	20 %	low	40 %	low
netmod_dol1	49 %	low	0.3 %	low	23 %	low
netmod_dol2	16 %	low	0.3 %	low	23 %	low
netmod_kar1	79 %	high	0.9 %	low	30 %	low
netmod_kar2	79 %	high	0.9 %	low	30 %	low
no7_ar2_1	100 %	high	13 %	low	38 %	low
no7_ar25_1	100 %	high	13 %	low	38 %	low
no7_ar3_1	100 %	high	13 %	low	38 %	low
no7_ar4_1	100 %	high	13 %	low	38 %	low
no7_ar5_1	100 %	high	13 %	low	38 %	low

Instance name	Relaxation gap	cat.	Nonlinearity measure	cat.	Discreteness measure	cat.
nvs03	49 %	low	100 %	high	100 %	high
nvs10	0.74 %	low	100 %	high	100 %	high
nvs11	0.41 %	low	100 %	high	100 %	high
nvs12	0.41 %	low	100 %	high	100 %	high
nvs15	89 %	high	100 %	high	100 %	high
o7	100 %	high	12 %	low	37 %	low
o7_2	100 %	high	12 %	low	37 %	low
o7_ar2_1	100 %	high	13 %	low	38 %	low
o7_ar25_1	100 %	high	13 %	low	38 %	low
o7_ar3_1	100 %	high	13 %	low	38 %	low
o7_ar4_1	100 %	high	13 %	low	38 %	low
o7_ar5_1	100 %	high	13 %	low	38 %	low
o8_ar4_1	100 %	high	11 %	low	39 %	low
o9_ar4_1	100 %	high	10 %	low	40 %	low
portfol_buyin	3.9 %	low	47 %	low	47 %	low
portfol_card	4.0 %	low	47 %	low	47 %	low
portfol_classical050_1	3.2 %	low	33 %	low	33 %	low
portfol_classical200_2	13 %	low	33 %	low	33 %	low
portfol_roundlot	0.00 %	low	47 %	low	47 %	low
procurement2mot	37 %	low	1.5 %	low	7.5 %	low
ravempb	15 %	low	25 %	low	48 %	low
risk2bpb	1.0 %	low	0.6 %	low	3.0 %	low
rsyn0805h	5.0 %	low	2.9 %	low	12 %	low
rsyn0805m	63 %	high	1.8 %	low	41 %	low
rsyn0805m02h	3.1 %	low	2.6 %	low	21 %	low
rsyn0805m02m	160 %	high	1.7 %	low	41 %	low
rsyn0805m03h	1.7 %	low	2.6 %	low	21 %	low
rsyn0805m03m	104 %	high	1.7 %	low	41 %	low
rsyn0805m04h	0.46 %	low	2.6 %	low	21 %	low
rsyn0805m04m	57 %	high	1.7 %	low	41 %	low
rsyn0810h	3.8 %	low	5.2 %	low	12 %	low
rsyn0810m	72 %	high	3.2 %	low	40 %	low
rsyn0810m02h	4.0 %	low	4.6 %	low	21 %	low
rsyn0810m02m	298 %	high	2.9 %	low	41 %	low
rsyn0810m03h	2.8 %	low	4.6 %	low	21 %	low
rsyn0810m03m	206 %	high	2.9 %	low	41 %	low
rsyn0810m04h	0.93 %	low	4.6 %	low	21 %	low
rsyn0810m04m	113 %	high	2.9 %	low	41 %	low
rsyn0815h	6.7 %	low	8.0 %	low	12 %	low
rsyn0815m	104 %	high	5.4 %	low	39 %	low
rsyn0815m02h	4.2 %	low	6.9 %	low	21 %	low
rsyn0815m02m	277 %	high	4.7 %	low	40 %	low
rsyn0815m03h	3.1 %	low	6.9 %	low	21 %	low
rsyn0815m03m	186 %	high	4.7 %	low	40 %	low
rsyn0815m04h	1.7 %	low	6.9 %	low	21 %	low

Instance name	Relaxation gap	Relaxation cat.	Nonlinearity measure	Nonlinearity cat.	Discreteness measure	Discreteness cat.
rsyn0815m04m	230 %	high	4.7 %	low	40 %	low
rsyn0820h	7.3 %	low	10 %	low	12 %	low
rsyn0820m	239 %	high	6.5 %	low	39 %	low
rsyn0820m02h	1.2 %	low	8.2 %	low	21 %	low
rsyn0820m02m	536 %	high	5.5 %	low	41 %	low
rsyn0820m03h	3.6 %	low	8.2 %	low	21 %	low
rsyn0820m03m	335 %	high	5.5 %	low	41 %	low
rsyn0820m04h	2.4 %	low	8.2 %	low	21 %	low
rsyn0820m04m	380 %	high	5.5 %	low	41 %	low
rsyn0830h	10 %	low	12 %	low	13 %	low
rsyn0830m	380 %	high	8.0 %	low	38 %	low
rsyn0830m02h	6.1 %	low	10 %	low	21 %	low
rsyn0830m02m	674 %	high	6.5 %	low	40 %	low
rsyn0830m03h	3.0 %	low	10 %	low	21 %	low
rsyn0830m03m	470 %	high	6.5 %	low	40 %	low
rsyn0830m04h	2.0 %	low	10 %	low	21 %	low
rsyn0830m04m	392 %	high	6.5 %	low	40 %	low
rsyn0840h	8.2 %	low	14 %	low	13 %	low
rsyn0840m	753 %	high	10 %	low	37 %	low
rsyn0840m02h	5.8 %	low	12 %	low	21 %	low
rsyn0840m02m	876 %	high	7.8 %	low	40 %	low
rsyn0840m03h	2.3 %	low	12 %	low	21 %	low
rsyn0840m03m	266 %	high	7.8 %	low	40 %	low
rsyn0840m04h	2.1 %	low	12 %	low	21 %	low
rsyn0840m04m	508 %	high	7.8 %	low	40 %	low
slay04h	13 %	low	5.7 %	low	17 %	low
slay04m	13 %	low	18 %	low	55 %	high
slay05h	5.9 %	low	4.3 %	low	17 %	low
slay05m	5.9 %	low	14 %	low	57 %	high
slay06h	7.0 %	low	3.5 %	low	18 %	low
slay06m	7.0 %	low	12 %	low	59 %	high
slay07h	4.6 %	low	2.9 %	low	18 %	low
slay07m	4.6 %	low	10 %	low	60 %	high
slay08h	4.9 %	low	2.5 %	low	18 %	low
slay08m	4.9 %	low	8.7 %	low	61 %	high
slay09h	4.3 %	low	2.2 %	low	18 %	low
slay09m	4.3 %	low	7.7 %	low	62 %	high
slay10h	8.1 %	low	2.0 %	low	18 %	low
slay10m	8.1 %	low	6.9 %	low	62 %	high
smallinvDAXr1b010-011	1.8 %	low	97 %	high	97 %	high
smallinvDAXr1b020-022	0.36 %	low	97 %	high	97 %	high
smallinvDAXr1b050-055	0.10 %	low	97 %	high	97 %	high
smallinvDAXr1b100-110	0.04 %	low	97 %	high	97 %	high
smallinvDAXr1b150-165	0.03 %	low	97 %	high	97 %	high
smallinvDAXr1b200-220	0.01 %	low	97 %	high	97 %	high

Instance name	Relaxation gap	Relaxation cat.	Nonlinearity measure	Nonlinearity cat.	Discreteness measure	Discreteness cat.
smallinvDAXr2b010-011	1.8 %	low	97 %	high	97 %	high
smallinvDAXr2b020-022	0.36 %	low	97 %	high	97 %	high
smallinvDAXr2b050-055	0.10 %	low	97 %	high	97 %	high
smallinvDAXr2b100-110	0.04 %	low	97 %	high	97 %	high
smallinvDAXr2b150-165	0.03 %	low	97 %	high	97 %	high
smallinvDAXr2b200-220	0.01 %	low	97 %	high	97 %	high
smallinvDAXr3b010-011	1.8 %	low	97 %	high	97 %	high
smallinvDAXr3b020-022	0.36 %	low	97 %	high	97 %	high
smallinvDAXr3b050-055	0.10 %	low	97 %	high	97 %	high
smallinvDAXr3b100-110	0.04 %	low	97 %	high	97 %	high
smallinvDAXr3b150-165	0.03 %	low	97 %	high	97 %	high
smallinvDAXr3b200-220	0.01 %	low	97 %	high	97 %	high
smallinvDAXr4b010-011	1.8 %	low	97 %	high	97 %	high
smallinvDAXr4b020-022	0.36 %	low	97 %	high	97 %	high
smallinvDAXr4b050-055	0.10 %	low	97 %	high	97 %	high
smallinvDAXr4b100-110	0.04 %	low	97 %	high	97 %	high
smallinvDAXr4b150-165	0.03 %	low	97 %	high	97 %	high
smallinvDAXr4b200-220	0.01 %	low	97 %	high	97 %	high
smallinvDAXr5b010-011	1.8 %	low	97 %	high	97 %	high
smallinvDAXr5b020-022	0.36 %	low	97 %	high	97 %	high
smallinvDAXr5b050-055	0.10 %	low	97 %	high	97 %	high
smallinvDAXr5b100-110	0.04 %	low	97 %	high	97 %	high
smallinvDAXr5b150-165	0.03 %	low	97 %	high	97 %	high
smallinvDAXr5b200-220	0.01 %	low	97 %	high	97 %	high
squfl010-025	51 %	high	96 %	high	3.8 %	low
squfl010-040	43 %	low	98 %	high	2.4 %	low
squfl010-080	49 %	low	99 %	high	1.2 %	low
squfl015-060	58 %	high	98 %	high	1.6 %	low
squfl015-080	57 %	high	99 %	high	1.2 %	low
squfl020-040	53 %	high	98 %	high	2.4 %	low
squfl020-050	57 %	high	98 %	high	2.0 %	low
squfl020-150	59 %	high	99 %	high	0.7 %	low
squfl025-025	60 %	high	96 %	high	3.8 %	low
squfl025-030	60 %	high	97 %	high	3.2 %	low
squfl025-040	61 %	high	98 %	high	2.4 %	low
squfl030-100	66 %	high	99 %	high	1.0 %	low
squfl030-150	63 %	high	99 %	high	0.7 %	low
squfl040-080	65 %	high	99 %	high	1.2 %	low
sssd08-04	62 %	high	6.7 %	low	73 %	high
sssd12-05	61 %	high	5.3 %	low	79 %	high
sssd15-04	62 %	high	4.5 %	low	82 %	high
sssd15-06	65 %	high	4.5 %	low	82 %	high
sssd15-08	63 %	high	4.5 %	low	82 %	high
sssd16-07	63 %	high	4.3 %	low	83 %	high
sssd18-06	63 %	high	4.0 %	low	84 %	high

Instance name	Relaxation gap	cat.	Nonlinearity measure	cat.	Discreteness measure	cat.
sssd18-08	67 %	high	4.0 %	low	84 %	high
sssd20-04	63 %	high	3.7 %	low	85 %	high
sssd20-08	62 %	high	3.7 %	low	85 %	high
sssd22-08	62 %	high	3.4 %	low	86 %	high
sssd25-04	64 %	high	3.1 %	low	88 %	high
sssd25-08	61 %	high	3.1 %	low	88 %	high
st_e14	15 %	low	64 %	high	36 %	low
st_miqp1	15 %	low	100 %	high	100 %	high
st_miqp2	382 %	high	50 %	high	100 %	high
st_miqp3	0.00 %	low	50 %	high	100 %	high
st_miqp4	0.05 %	low	50 %	high	50 %	high
st_miqp5	0.00 %	low	29 %	low	29 %	low
st_test1	$3 \cdot 10^6$ %	high	80 %	high	100 %	high
st_test2	9.5 %	low	83 %	high	100 %	high
st_test3	24 %	low	38 %	low	100 %	high
st_test4	11 %	low	33 %	low	100 %	high
st_test5	104 %	high	70 %	high	100 %	high
st_test6	30 %	low	100 %	high	100 %	high
st_test8	0.00 %	low	100 %	high	100 %	high
st_testgr1	0.11 %	low	100 %	high	100 %	high
st_testgr3	0.63 %	low	100 %	high	100 %	high
st_testph4	3.1 %	low	100 %	high	100 %	high
stockcycle	1.7 %	low	10 %	low	90 %	high
syn05h	0.03 %	low	21 %	low	12 %	low
syn05m	37 %	low	15 %	low	25 %	low
syn05m02h	0.02 %	low	17 %	low	19 %	low
syn05m02m	19 %	low	10 %	low	33 %	low
syn05m03h	0.02 %	low	17 %	low	19 %	low
syn05m03m	20 %	low	10 %	low	33 %	low
syn05m04h	0.01 %	low	17 %	low	19 %	low
syn05m04m	20 %	low	10 %	low	33 %	low
syn10h	0.03 %	low	23 %	low	13 %	low
syn10m	58 %	high	17 %	low	29 %	low
syn10m02h	0.08 %	low	19 %	low	21 %	low
syn10m02m	104 %	high	11 %	low	36 %	low
syn10m03h	0.05 %	low	19 %	low	21 %	low
syn10m03m	103 %	high	11 %	low	36 %	low
syn10m04h	0.04 %	low	19 %	low	21 %	low
syn10m04m	103 %	high	11 %	low	36 %	low
syn15h	0.12 %	low	26 %	low	12 %	low
syn15m	97 %	high	20 %	low	27 %	low
syn15m02h	0.10 %	low	21 %	low	20 %	low
syn15m02m	66 %	high	13 %	low	35 %	low
syn15m03h	0.07 %	low	21 %	low	20 %	low
syn15m03m	74 %	high	13 %	low	35 %	low

Instance name	Relaxation gap	cat.	Nonlinearity measure	cat.	Discreteness measure	cat.
syn15m04h	0.06 %	low	21 %	low	20 %	low
syn15m04m	91 %	high	13 %	low	35 %	low
syn20h	0.32 %	low	26 %	low	13 %	low
syn20m	221 %	high	22 %	low	31 %	low
syn20m02h	0.30 %	low	21 %	low	21 %	low
syn20m02m	179 %	high	13 %	low	38 %	low
syn20m03h	0.27 %	low	21 %	low	21 %	low
syn20m03m	178 %	high	13 %	low	38 %	low
syn20m04h	0.20 %	low	21 %	low	21 %	low
syn20m04m	179 %	high	13 %	low	38 %	low
syn30h	6.1 %	low	25 %	low	13 %	low
syn30m	932 %	high	20 %	low	30 %	low
syn30m02h	2.9 %	low	20 %	low	21 %	low
syn30m02m	677 %	high	13 %	low	38 %	low
syn30m03h	2.0 %	low	20 %	low	21 %	low
syn30m03m	593 %	high	13 %	low	38 %	low
syn30m04h	1.6 %	low	20 %	low	21 %	low
syn30m04m	613 %	high	13 %	low	38 %	low
syn40h	17 %	low	26 %	low	13 %	low
syn40m	2608 %	high	22 %	low	31 %	low
syn40m02h	2.4 %	low	21 %	low	21 %	low
syn40m02m	1072 %	high	13 %	low	38 %	low
syn40m03h	5.6 %	low	21 %	low	21 %	low
syn40m03m	1467 %	high	13 %	low	38 %	low
syn40m04h	2.0 %	low	21 %	low	21 %	low
syn40m04m	917 %	high	13 %	low	38 %	low
synthes1	87 %	high	33 %	low	50 %	high
synthes2	101 %	high	36 %	low	45 %	low
synthes3	78 %	high	35 %	low	47 %	low
tls12	98 %	high	19 %	low	82 %	high
tls2	86 %	high	16 %	low	89 %	high
tls4	79 %	high	19 %	low	85 %	high
tls5	89 %	high	19 %	low	84 %	high
tls6	91 %	high	20 %	low	83 %	high
tls7	96 %	high	16 %	low	86 %	high
unitcommit1	1.6 %	low	25 %	low	75 %	high
watercontamination0202	52 %	high	3.8 %	low	0.0 %	low
watercontamination0202r	100 %	high	48 %	low	3.6 %	low
watercontamination0303	64 %	high	4.2 %	low	0.0 %	low
watercontamination0303r	100 %	high	48 %	low	3.6 %	low

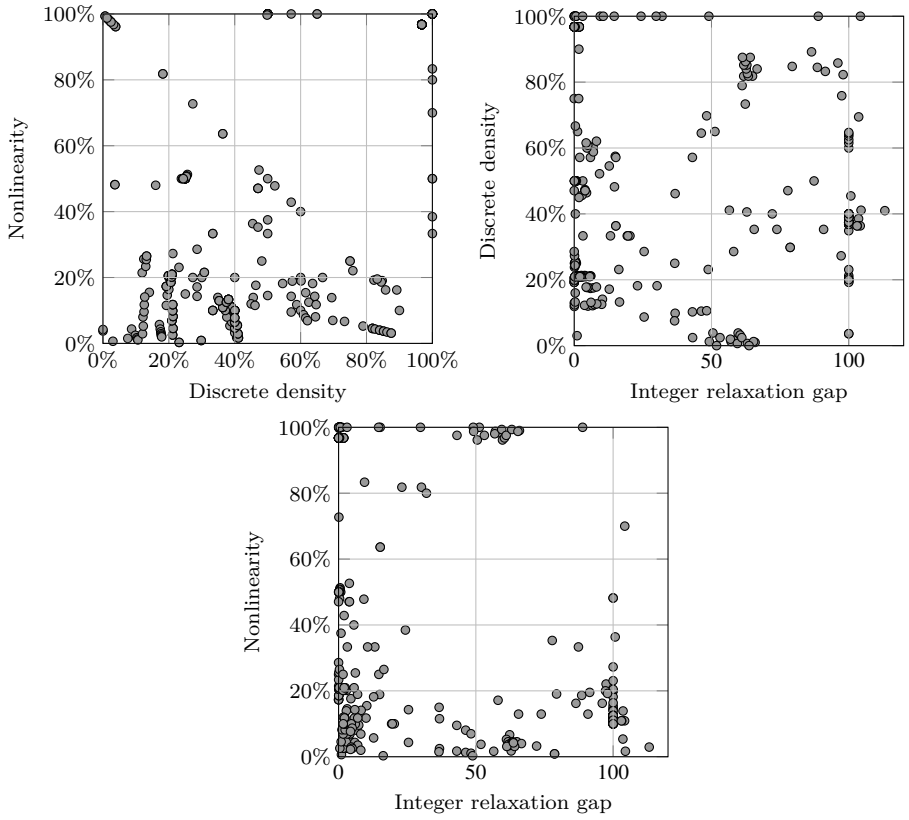


Figure B.1: As can be seen from these scatter plots, there is little to no correlation between the three categories integer relaxation gap, nonlinearity, and discrete density. Note that some outliers are missing.

B.B Solver Options

The options provided to the solvers (and subsolvers) in the benchmark are listed below. All other settings are the default values as provided by the individual solvers. Note that we have not tried to fine-tune any of the solvers; however, if there is a convex strategy or recommended convex parameters, we have used those. We have also modified limits and other parameters when it is apparent that implementation issues cause, *e.g.,* premature termination of a solver on some problem instances or numerical instability. For example, without adding the CONOPT option specified below, DICOPT and SBB fail to solve all the `smallinv`-instances in MINLPLib. Therefore, we believe that it is motivated to use these since this problem occurs in the subsolver.

Name	Value	Description
General GAMS		
MIP	CPLEX	uses CPLEX as MIP solver
threads	8	max amount of threads
optCR	0.001	relative termination tolerance
optCA	0	absolute termination tolerance
nodLim	10^8	to avoid premature termination
domLim	10^8	to avoid premature termination
iterLim	10^8	to avoid premature termination
resLim	900	time limit
AlphaECP		
ECPmaster	1	activates convex strategy
TOLepsg	10^{-6}	constraint tolerance
AOA		
IsConvex	1	activates convex strategy
IterationMax	10^7	maximal number of iterations
RelativeOptimalityTolerance	0.1	relative termination tolerance (in %)
TimeLimit	900	time limit
BONMIN		
bonmin.algorithm	B-OA B-HYB	selects the main algorithm

	B-BB	
milp_solver	CPLEX	uses CPLEX as MILP solver
bonmin.time_limit	900	sets the time limit
Couenne		
lp_solver	Clp	uses Clp as LP solver (as recommended in the manual)
DICOPT		
convex	1	activates convex strategy
stop	1	convex stopping criterion
maxcycles	10^8	iteration limit
infeasder	1	add cutting planes from infeasible NLP (convex recommendation)
nlpoptfile	1	to use the CONOPT options below
Juniper		
fp_cpx		use CPLEX for the feasibility pump
processors	8	max number of threads
mip_gap	0.001	relative termination tolerance
time-limit	900	time limit
LINDO		
USEGOP	0	deactivates global strategy
SPLEX_ITRLMT	-1	simplex iteration limit
MIP_ITRLIM	-1	MILP iteration limit the iteration limits are set as infinite to avoid premature termination
Minotaur		
lp_engine	OsiCpx	use CPLEX as MIP solver
obj_gap_percent	0.1	relative termination tolerance (in %)
bnb_time_limit	900	time limit
threads	8	max amount of threads (does not seem to have any effect as it only uses one thread)
Muriqui		
MRQ_LP_NLP_BB_OA_BASED_ALG		use LP/NLP based algorithm
in_assume_convexity	1	use convex strategy
in_absolute_convergence_tol	0	absolute termination tolerance
in_relative_convergence_tol	0.001	relative termination tolerance

in_absolute_feasibility_tol	10^{-6}	constraint tolerance
in_integer_tol	10^{-5}	integer tolerance
in_max_time	900	time limit
in_milp_solver	MRQ_CPLEX	use CPLEX as MIP solver
in_nlp_solver	MRQ_IPOPT	use IPOPT as NLP solver
in_number_of_threads	8	max amount of threads
Pavito		
mip_solver	CPLEXSolver	use CPLEX as MILP solver; with one thread since multiple threads are not supported with callbacks
cont_solver	IpoptSolver	use IPOPT as NLP solver
mip_solver_drives	true	let MILP solver manage tree
rel_gap	0.001	relative termination tolerance
timeout	900	time limit
SBB		
memnodes	$5 \cdot 10^7$	to avoid premature termination, but not too large, since memory is preallocated
rootsolver	CONOPT.1	to use the CONOPT options below
SCIP		
constraints/nonlinear/assumeconvex	true	activates convex strategy
SHOT		
Dual.MIP.NumberOfThreads	8	max number of threads
Dual.MIP.Solver	0	use CPLEX as MIP solver
Primal.FixedInteger.Solver	2	to use GAMS NLP solvers
Subsolver.GAMS.NLP.Solver	conopt	use CONOPT as GAMS NLP solver
Termination.ObjectiveGap.Absolute	0	absolute termination tolerance
Termination.ObjectiveGap.Relative	0.001	relative termination tolerance
Termination.TimeLimit	900	time limit
CONOPT (GAMS)		
RTMAXV	10^{30}	to avoid problems with unbounded variables in DICOPT and SBB

Bibliography

[1] EN Pistikopoulos, Ana Barbosa-Povoa, Jay H Lee, Ruth Misener, Alexander Mitsos, GV Reklaitis, V Venkatasubramanian, Fengqi You, and Rafiqul Gani. "Process Systems Engineering–The Generation Next?" *Computers & Chemical Engineering* (2021), p. 107252.

[2] Egon Balas. *Disjunctive programming*. Springer, 2018.

[3] Leo Liberti and Fabrizio Marinelli. "Mathematical programming: Turing completeness and applications to software analysis". *Journal of Combinatorial Optimization* 28.1 (2014), pp. 82–104.

[4] Lorenz T Biegler and Ignacio E Grossmann. "Retrospective on optimization". *Computers & Chemical Engineering* 28.8 (2004), pp. 1169–1192.

[5] Francisco Trespalacios and Ignacio E. Grossmann. "Review of mixed-integer nonlinear and generalized disjunctive programming methods". *Chemie Ingenieur Technik* 86.7 (2014), pp. 991–1012. DOI: 10.1002/cite.201400037.

[6] Fani Boukouvala, Ruth Misener, and Christodoulos A Floudas. "Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO". *European Journal of Operational Research* 252.3 (2016), pp. 701–727.

[7] T.F. Yee and I.E. Grossmann. "Simultaneous optimization models for heat integration—II. Heat exchanger network synthesis". *Computers & Chemical Engineering* 14.10 (1990), pp. 1165–1184. DOI: 10.1016/0098-1354(90)85010-8.

[8] Radhakrishna Tumbalam Gooty, Rakesh Agrawal, and Mohit Tawarmalani. "An MINLP formulation for the optimization of multicomponent distillation configurations". *Computers & Chemical Engineering* 125 (2019), pp. 13–30.

[9] David A Liñán, David E Bernal, Luis A Ricardez-Sandoval, and Jorge M Gómez. "Optimal design of superstructures for placing units and streams with multiple and ordered available locations. Part II: Rigorous design of catalytic distillation columns". *Computers & Chemical Engineering* 139 (2020), p. 106845.

[10] David A Linan, David E Bernal, Jorge M Gomez, and Luis A Ricardez-Sandoval. "Optimal synthesis and design of catalytic distillation columns: A rate-based modeling approach". *Chemical Engineering Science* 231 (2021), p. 116294.

[11] NV Sahinidis and Ignacio E Grossmann. "MINLP model for cyclic multiproduct scheduling on continuous parallel lines". *Computers & chemical engineering* 15.2 (1991), pp. 85–103.

[12] L Mockus and GV Reklaitis. "Continuous time representation approach to batch and continuous process scheduling. 1. MINLP formulation". *Industrial & Engineering Chemistry Research* 38.1 (1999), pp. 197–203.

[13] Lijie Su, Lixin Tang, David E Bernal, Ignacio E Grossmann, and Bowen Wang. "Integrated scheduling of on-line blending and distribution of oil products in refinery operation". In: *Computer Aided Chemical Engineering*. Vol. 44. Elsevier, 2018, pp. 1213–1218.

[14] Haokun Yang, David E Bernal, Robert E Franzoi, Faramroze G Engineer, Kysang Kwon, Sechan Lee, and Ignacio E Grossmann. "Integration of crude-oil scheduling and refinery planning by Lagrangean Decomposition". *Computers & Chemical Engineering* 138 (2020), p. 106812.

[15] Carl A Schweiger and Christodoulos A Floudas. "Interaction of design and control: Optimization with dynamic models". In: *Optimal Control*. Springer, 1998, pp. 388–435.

[16] Ignacio E. Grossmann and Jorge Santibanez. "Applications of mixed-integer linear programming in process synthesis". *Computers & Chemical Engineering* 4.4 (1980), pp. 205–214. DOI: 10.1016/0098-1354(80)85001-0.

[17] Ignacio E. Grossmann, Jose Antonio Caballero, and Hector Yeomans. "Mathematical programming approaches to the synthesis of chemical process systems". *Korean Journal of Chemical Engineering* 16.4 (1999), pp. 407–426. DOI: 10.1007/BF02698263.

[18] Pierre Bonami and Miguel A Lejeune. "An exact solution approach for portfolio optimization problems under stochastic and integer constraints". *Operations Research* 57.3 (2009), pp. 650–670.

[19] Simon Šilih, Tomaž Žula, and Stojan Kravanja. "The MINLP optimization in civil engineering". In: *Proceedings of the 9th WSEAS International Conference on Mathematical and Computational Methods in Science and Engineering*. 2007, pp. 5–7.

[20] Wenhua Cao and Gino J Lim. "Optimization models for cancer treatment planning". In: *Wiley Encyclopedia of Operations Research and Management Science*. Wiley Online Library, 2011.

[21] Costas D Maranas and Ali R Zomorrodi. *Optimization methods in metabolic networks*. John Wiley & Sons, 2016.

[22] Cristiana L Lara, David E Bernal, Can Li, and Ignacio E Grossmann. "Global optimization algorithm for multi-period design and planning of centralized and dis-

tributed manufacturing networks". *Computers & Chemical Engineering* 127 (2019), pp. 295–310.

[23] Leo Liberti. *Mathematical Programming*. Paris: Ecole Polytechnique, 2017.

[24] Leo Liberti. "Undecidability and hardness in mixed-integer nonlinear programming". *RAIRO-Operations Research* 53.1 (2019), pp. 81–109.

[25] Roger Fletcher and Sven Leyffer. "Solving mixed integer nonlinear programs by outer approximation". *Mathematical Programming* 66.1 (1994), pp. 327–349.

[26] Jan Kronqvist, David E Bernal, Andreas Lundell, and Ignacio E Grossmann. "A review and comparison of solvers for convex MINLP". *Optimization and Engineering* 20.2 (2019), pp. 397–455.

[27] Robert J Dakin. "A tree-search algorithm for mixed integer programming problems". *The Computer Journal* 8.3 (1965), pp. 250–255.

[28] Arthur M Geoffrion. "Generalized Benders decomposition". *Journal of Optimization Theory and Applications* 10.4 (1972), pp. 237–260.

[29] Tapio Westerlund, Hans Skrifvars, Iiro Harjunkoski, and Ray Pörn. "An extended cutting plane method for a class of non-convex MINLP problems". *Computers & Chemical Engineering* 22.3 (1998), pp. 357–365.

[30] Marco A Duran and Ignacio E Grossmann. "An outer-approximation algorithm for a class of mixed-integer nonlinear programs". *Mathematical Programming* 36.3 (1986), pp. 307–339.

[31] Aharon Ben-Tal and Arkadi Nemirovski. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*. SIAM, 2001.

[32] Fatma Kılınç-Karzan. "On minimal valid inequalities for mixed integer conic programs". *Mathematics of Operations Research* 41.2 (2016), pp. 477–510.

[33] Miles Lubin, Ilias Zadik, and Juan Pablo Vielma. "Mixed-integer convex representability". In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2017, pp. 392–404.

[34] Henrik A Friberg. "CBLIB 2014: a benchmark library for conic mixed-integer and continuous optimization". *Mathematical Programming Computation* 8.2 (2016), pp. 191–214.

[35] Mosek ApS. *MOSEK modeling cookbook*. 2018.

[36] Alexander Domahidi, Eric Chu, and Stephen Boyd. "ECOS: An SOCP solver for embedded systems". In: *2013 European Control Conference (ECC)*. IEEE. 2013, pp. 3071–3076.

[37] Chris Coey, Lea Kapelevich, and Juan Pablo Vielma. "Towards practical generic conic optimization". *arXiv preprint arXiv:2005.01136* (2020).

[38] Robert J Vanderbei and Hande Yurttan. "Using LOQO to solve second-order cone programming problems". *Constraints* 1 (), p. 2.

[39] Victor Zverovich, Robert Fourer, and AMPL Optimization. "Automatic Reformulation of Second-Order Cone Programming Problems Second-order cone programming (SOCP)". In: *INFORMS Computing Society Conference*. 2015.

[40] Jared Erickson and Robert Fourer. "Detection and Transformation of Second-Order Cone Programming Problems in a General-Purpose Algebraic Modeling Language" (2019).

[41] Michael Grant, Stephen Boyd, and Yinyu Ye. "Disciplined convex programming". In: *Global optimization*. Springer, 2006, pp. 155–210.

[42] Miles Lubin, Emre Yamangil, Russell Bent, and Juan Pablo Vielma. "Extended Formulations in Mixed-Integer Convex Programming". In: *Integer Programming and Combinatorial Optimization: 18th International Conference, IPCO 2016*. Ed. by Quentin Louveaux and Martin Skutella. Springer. Springer International Publishing, 2016, pp. 102–113.

[43] Miles Lubin, Emre Yamangil, Russell Bent, and Juan Pablo Vielma. "Polyhedral approximation in mixed-integer convex optimization". *Mathematical Programming* 172.1-2 (2018), pp. 139–168.

[44] Chris Coey, Miles Lubin, and Juan Pablo Vielma. "Outer approximation with conic certificates for mixed-integer convex problems". *Mathematical Programming Computation* (2020), pp. 1–45.

[45] Stefan Vigerske. "Decomposition in multistage stochastic programming and a constraint integer programming approach to mixed-integer nonlinear programming" (2013).

[46] Ksenia Bestuzheva, Ambros Gleixner, and Stefan Vigerske. "A Computational Study of Perspective Cuts". *arXiv preprint arXiv:2103.09573* (2021).

[47] Aida Khajavirad and Nikolaos V Sahinidis. "A hybrid LP/NLP paradigm for global optimization relaxations". *Mathematical Programming Computation* 10.3 (2018), pp. 383–421.

[48] Oktay Günlük and Jeff Linderoth. "Perspective reformulations of mixed integer nonlinear programs with indicator variables". *Mathematical programming* 124.1 (2010), pp. 183–205.

[49] RA Waltz and Plantenga TD. *KNITRO user's manual. 2010*. 2017.

[50] Pietro Belotti, Timo Berthold, and Kelligton Neves. "Algorithms for discrete nonlinear optimization in FICO Xpress". In: *2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*. IEEE. 2016, pp. 1–5.

[51] Inc. Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2016.

[52] IBM Corp. and IBM. *V20.1: User's Manual for CPLEX*. 2020.

[53] Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, et al. *Integer programming*. Vol. 271. Springer, 2014.

[54] Mehmet Tolga Çezik and Garud Iyengar. "Cuts for mixed 0-1 conic programming". *Mathematical Programming* 104.1 (2005), pp. 179–202.

[55] Pietro Belotti, Julio C Góez, Imre Pólik, Ted K Ralphs, and Tamás Terlaky. "A conic representation of the convex hull of disjunctive sets and conic cuts for integer second order cone optimization". In: *Numerical Analysis and Optimization*. Springer, 2015, pp. 1–35.

[56] Andrea Lodi, Mathieu Tanneau, and Juan Pablo Vielma. "Disjunctive cuts in Mixed-Integer Conic Optimization". *arXiv preprint arXiv:1912.03166* (2019).

[57] Pierre Bonami, Andrea Lodi, Andrea Tramontani, and Sven Wiese. "On mathematical programming with indicator constraints". *Mathematical programming* 151.1 (2015), pp. 191–223.

[58] Ignacio E Grossmann and Sangbum Lee. "Generalized convex disjunctive programming: Nonlinear convex hull relaxation". *Computational optimization and applications* 26.1 (2003), pp. 83–100.

[59] Qi Chen, Emma S Johnson, David E Bernal, Romeo Valentin, Sunjeev Kale, Johnny Bates, John D Siirola, and Ignacio E Grossmann. "Pyomo. GDP: an ecosystem for

logic based modeling and optimization development". *Optimization and Engineering* (2021), pp. 1–36. DOI: 10.1007/s11081-021-09601-7.

[60] Francisco Trespalacios and Ignacio E Grossmann. "Review of mixed-integer nonlinear and generalized disjunctive programming methods". *Chemie Ingenieur Technik* 86.7 (2014), pp. 991–1012.

[61] Oktay Günlük and Jeff Linderoth. "Perspective reformulation and applications". In: *Mixed Integer Nonlinear Programming*. Springer, 2012, pp. 61–89.

[62] Kevin C Furman, Nicolas W Sawaya, and Ignacio E Grossmann. "A computationally useful algebraic representation of nonlinear disjunctive convex sets using the perspective function". *Computational Optimization and Applications* (2020), pp. 1–26.

[63] Hassan Hijazi, Pierre Bonami, Gérard Cornuéjols, and Adam Ouorou. "Mixed-integer nonlinear programs featuring "on/off" constraints". *Computational Optimization and Applications* 52.2 (2012), pp. 537–558.

[64] Sangbum Lee and Ignacio E Grossmann. "New algorithms for nonlinear generalized disjunctive programming". *Computers & Chemical Engineering* 24.9-10 (2000), pp. 2125–2141.

[65] Robert A Stubbs and Sanjay Mehrotra. "A branch-and-cut method for 0-1 mixed convex programming". *Mathematical programming* 86.3 (1999), pp. 515–532.

[66] Antonio Frangioni and Claudio Gentile. "Perspective cuts for a class of convex 0–1 mixed integer programs". *Mathematical Programming* 106.2 (2006), pp. 225–236.

[67] M Selim Aktürk, Alper Atamtürk, and Sinan Gürel. "A strong conic quadratic reformulation for machine-job assignment with controllable processing times". *Operations Research Letters* 37.3 (2009), pp. 187–191.

[68] John Von Neumann. "The principles of large-scale computing machines". *Annals of the History of Computing* 3.3 (1981), pp. 263–273.

[69] Cristian S Calude. "Unconventional computing: A brief subjective history". In: *Advances in Unconventional Computing*. Springer, 2017, pp. 855–864.

[70] Sridhar Tayur. *Unconventional Computing: Applications, Hardware, Algorithms*. 2021. DOI: 10.1287/orms.2021.01.14.

[71] Eleanor G Rieffel and Wolfgang H Polak. *Quantum computing: A gentle introduction*. MIT Press, 2011.

[72] Peter W Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th annual symposium on foundations of computer science*. Ieee. 1994, pp. 124–134.

[73] Lov K Grover. "Quantum mechanics helps in searching for a needle in a haystack". *Physical review letters* 79.2 (1997), p. 325.

[74] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[75] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[76] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum algorithm for linear systems of equations". *Physical review letters* 103.15 (2009), p. 150502.

[77] Joran Van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. "Quantum SDP-solvers: Better upper and lower bounds". In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2017, pp. 403–414.

[78] Iordanis Kerenidis and Anupam Prakash. "A quantum interior point method for LPs and SDPs". *ACM Transactions on Quantum Computing* 1.1 (2020), pp. 1–32.

[79] Andrew Lucas. "Ising formulations of many NP problems". *Frontiers in Physics* 2 (2014), p. 5.

[80] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. "Quantum computation by adiabatic evolution". *arXiv preprint quant-ph/0001106* (2000).

[81] Tameem Albash and Daniel A Lidar. "Adiabatic quantum computation". *Reviews of Modern Physics* 90.1 (2018), p. 015002.

[82] Erica K Grant and Travis S Humble. "Adiabatic quantum computing and quantum annealing". In: *Oxford Research Encyclopedia of Physics*. 2020.

[83] Catherine C McGeoch. "Adiabatic quantum computation and quantum annealing: Theory and practice". *Synthesis Lectures on Quantum Computing* 5.2 (2014), pp. 1–93.

[84] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. "Next-generation topology of D-Wave quantum processors". *arXiv preprint arXiv:2003.00133* (2020).

[85] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. "A quantum approximate optimization algorithm". *arXiv preprint arXiv:1411.4028* (2014).

[86] Vijay V Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[87] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. "A variational eigenvalue solver on a photonic quantum processor". *Nature communications* 5 (2014), p. 4213.

[88] Michael R Bussieck and Stefan Vigerske. "MINLP solver software". In: *Wiley encyclopedia of operations research and management science*. Wiley Online Library, 2010.

[89] Jan Kronqvist, David E Bernal, and Ignacio E Grossmann. "Using regularization and second order information in outer approximation for convex MINLP". *Mathematical Programming* 180.1 (2020), pp. 285–310.

[90] Ignacio Quesada and Ignacio E Grossmann. "An LP/NLP based branch and bound algorithm for convex MINLP optimization problems". *Computers & chemical engineering* 16.10-11 (1992), pp. 937–947. DOI: 10.1016/0098-1354(92)80028-8.

[91] Raouf Dridi, Hedayat Alghassi, and Sridhar Tayur. "A novel algebraic geometry compiling framework for adiabatic quantum computations". *arXiv preprint arXiv:1810.01440* (2018).

[92] Farid Alizadeh and Donald Goldfarb. "Second-order cone programming". *Mathematical programming* 95.1 (2003), pp. 3–51.

[93] Johannes Bisschop. *AIMMS optimization modeling*. Lulu.com, 2006.

[94] Robert Fourer, David Gay, and Brian Kernighan. *AMPL*. Boyd & Fraser Danvers, MA, 1993.

[95] Anthony Brook, David Kendrick, and Alexander Meeraus. "GAMS, a user's guide". *ACM Signum Newsletter* 23.3-4 (1988), pp. 10–11.

[96] Jeff Bezanson, Stefan Karpinski, Viral B Shah, and Alan Edelman. *Julia: A fast dynamic language for technical computing*. arXiv preprint:1209.5145. 2012.

[97] Iain Dunning, Joey Huchette, and Miles Lubin. "JuMP: A modeling language for mathematical optimization". *SIAM Review* 59.2 (2017), pp. 295–320.

[98] William E Hart, Carl D Laird, Jean-Paul Watson, David L Woodruff, Gabriel A Hackebeil, Bethany L Nicholson, and John D Siirola. *Pyomo-optimization modeling in Python*. Vol. 67. Springer, 2012.

[99] Jonathan Currie and David I. Wilson. "OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User". In: *Foundations of Computer-Aided Process Operations*. Ed. by Nick Sahinidis and Jose Pinto. Savannah, Georgia, USA, 2012.

[100] Michael R Bussieck, Arne Stolbjerg Drud, and Alexander Meeraus. "MINLPLib—a collection of test models for mixed-integer nonlinear programming". *INFORMS Journal on Computing* 15.1 (2003), pp. 114–119.

[101] Ville-Pekka Eronen, Jan Kronqvist, Tapio Westerlund, Marko M Mäkelä, and Napsu Karmitsa. "Method for solving generalized convex nonsmooth mixed-integer nonlinear programming problems". *Journal of Global Optimization* 69.2 (2017), pp. 443–459.

[102] Ville-Pekka Eronen, Marko M Mäkelä, and Tapio Westerlund. "On the generalization of ECP and OA methods to nonsmooth convex MINLP problems". *Optimization* 63.7 (2014), pp. 1057–1073.

[103] Claudia D'Ambrosio and Andrea Lodi. "Mixed integer nonlinear programming tools: an updated practical overview". *Annals of Operations Research* 204.1 (2013), pp. 301–320. DOI: 10.1007/s10479-012-1272-5.

[104] Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. "Mixed-integer nonlinear optimization". *Acta Numerica* 22 (2013), pp. 1–131. DOI: 10.1017/S0962492913000032.

[105] Ignacio E Grossmann. "Review of nonlinear mixed-integer and disjunctive programming techniques". *Optimization and Engineering* 3.3 (2002), pp. 227–252. DOI: 10.1023/A:1021039126272.

[106] Christodoulos A Floudas. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.

[107] Ailsa H Land and Alison G Doig. "An automatic method of solving discrete programming problems". *Econometrica: Journal of the Econometric Society* (1960), pp. 497–520.

[108] Omprakash K Gupta and A Ravindran. "Branch and bound experiments in convex nonlinear integer programming". *Management science* 31.12 (1985), pp. 1533–1546.

[109] Mustafa R Kılınç, Jeff Linderoth, and James Luedtke. "Lift-and-project cuts for convex mixed integer nonlinear programs". *Mathematical Programming Computation* 9.4 (2017), pp. 499–526.

[110] Yushan Zhu and Takahito Kuno. "A disjunctive cutting-plane-based branch-and-cut algorithm for 0-1 mixed-integer convex nonlinear programs". *Industrial & Engineering Chemistry Research* 45.1 (2006), pp. 187–196.

[111] Egon Balas, Sebastián Ceria, and Gérard Cornuéjols. "A lift-and-project cutting plane algorithm for mixed 0–1 programs". *Mathematical programming* 58.1-3 (1993), pp. 295–324.

[112] Ralph E Gomory et al. "Outline of an algorithm for integer solutions to linear programs". *Bulletin of the American Mathematical society* 64.5 (1958), pp. 275–278.

[113] Brian Borchers and John E Mitchell. "An improved branch and bound algorithm for mixed integer nonlinear programs". *Computers & Operations Research* 21.4 (1994), pp. 359–367.

[114] Sven Leyffer. "Integrating SQP and branch-and-bound for mixed integer nonlinear programming". *Computational optimization and applications* 18.3 (2001), pp. 295–309.

[115] M. Tawarmalani and N. V. Sahinidis. "A polyhedral branch-and-cut approach to global optimization". *Mathematical Programming* 103 (2 2005), pp. 225–249.

[116] Pierre Bonami, Jon Lee, Sven Leyffer, and Andreas Wächter. "More branch-and-bound experiments in convex nonlinear integer programming". *Preprint, Optimization online* (2011).

[117] Christodoulos A Floudas. *Deterministic Global Optimization, vol. 37 of Nonconvex Optimization and its Applications*. 2000.

[118] T. Westerlund and F. Petterson. "An Extended Cutting Plane Method For Solving Convex MINLP Problems". *Computers & Chemical Engineering* 19 (1995), pp. 131–136.

[119] James E Kelley Jr. "The cutting-plane method for solving convex programs". *Journal of the Society for Industrial & Applied Mathematics* 8.4 (1960), pp. 703–712.

[120] Tapio Westerlund and Ray Pörn. "Solving pseudo-convex mixed integer optimization problems by cutting plane techniques". *Optimization and Engineering* 3.3 (2002), pp. 253–280.

[121] Jan Kronqvist, Andreas Lundell, and Tapio Westerlund. "The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming". *Journal of Global Optimization* 64.2 (2016), pp. 249–272.

[122] M. A. Duran and I. E. Grossmann. "An outer-approximation algorithm for a class of mixed-integer nonlinear programs". *Mathematical Programming* 36.3 (1986), p. 307. DOI: 10.1007/BF02592064.

[123] Gary R Kocis and Ignacio E Grossmann. "Global optimization of nonconvex mixed-integer nonlinear programming (MINLP) problems in process synthesis". *Industrial & engineering chemistry research* 27.8 (1988), pp. 1407–1421.

[124] J. Viswanathan and I. E. Grossmann. "A combined penalty function and outer-approximation method for MINLP optimization". *Computers & Chemical Engineering* 14.7 (1990), pp. 769–782. DOI: 10.1016/0098-1354(90)87085-4.

[125] Jacques F Benders. "Partitioning procedures for solving mixed-variables programming problems". *Numerische mathematik* 4.1 (1962), pp. 238–252.

[126] Ignacio E Grossmann and Zdravko Kravanja. "Mixed-integer nonlinear programming: A survey of algorithms and applications". In: *Large-scale optimization with applications*. Ed. by Lorenz T. Biegler, Thomas F. Coleman, Andrew R. Conn, and Fadil N. Santosa. Springer, 1997, pp. 73–100.

[127] Sven Leyffer. "Deterministic methods for mixed integer nonlinear programming". *PhD University of Dundee* (1993).

[128] Kumar Abhishek, Sven Leyffer, and Jeff Linderoth. "FilMINT: An outer approximation-based solver for convex mixed-integer nonlinear programs". *INFORMS Journal on Computing* 22.4 (2010), pp. 555–567.

[129] Pierre Bonami, Lorenz T. Biegler, Andrew R. Conn, Gérard Cornuéjols, Ignacio E. Grossmann, Carl D. Laird, Jon Lee, Andrea Lodi, François Margot, Nicolas Sawaya, and Andreas Wächter. "An algorithmic framework for convex mixed integer nonlinear programs". *Discrete Optimization* 5.2 (2008), pp. 186–204. DOI: 10.1016/j.disopt.2006.10.011.

[130] Ashutosh Mahajan, Svenn Leyffer, Jeff Linderoth, James Luedtke, and Todd Munson. "Minotaur: A Mixed-Integer Nonlinear Optimization Toolkit". *Preprint, Optimization Online* (2017).

[131] Pierre Bonami. "Lift-and-project cuts for mixed integer convex programs". In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2011, pp. 52–64.

[132] Jeff T Linderoth and Martin WP Savelsbergh. "A computational study of search strategies for mixed integer programming". *INFORMS Journal on Computing* 11.2 (1999), pp. 173–187.

[133] Tobias Achterberg, Thorsten Koch, and Alexander Martin. "Branching rules revisited". *Operations Research Letters* 33.1 (2005), pp. 42–54.

[134] J Parker Shectman and Nikolaos V Sahinidis. "A finite algorithm for global minimization of separable concave programs". *Journal of Global Optimization* 12.1 (1998), pp. 1–36.

[135] Pietro Belotti, Sonia Cafieri, Jon Lee, and Leo Liberti. "Feasibility-based bounds tightening via fixed points". In: *International Conference on Combinatorial Optimization and Applications*. Ed. by W. Wu and O. Daescu. Springer, 2010, pp. 65–76.

[136] Leo Liberti and Nelson Maculan. *Global optimization: from theory to implementation*. Vol. 84. Springer Science & Business Media, 2006.

[137] Laurence A Wolsey. *Integer Programming. Series in Discrete Mathematics and Optimization*. Wiley-Interscience New Jersey, 1998.

[138] Jan Kronqvist, Andreas Lundell, and Tapio Westerlund. "Reformulations for utilizing separability when solving convex MINLP problems". *Journal of Global Optimization* (2018), pp. 1–22.

[139] Hassan Hijazi, Pierre Bonami, and Adam Ouorou. "An outer-inner approximation for separable mixed-integer nonlinear programs". *INFORMS Journal on Computing* 26.1 (2013), pp. 31–44.

[140] Leo Liberti. *Reformulation techniques in mathematical programming*. HDR thesis. 2009.

[141] Matteo Fischetti and Andrea Lodi. "Heuristics in mixed integer programming". In: *Wiley Encyclopedia of Operations Research and Management Science*. Wiley Online Library, 2011.

[142] Timo Berthold and Ambros M Gleixner. "Undercover: a primal MINLP heuristic exploring a largest sub-MIP". *Mathematical Programming* 144.1-2 (2014), pp. 315–346.

[143] Pierre Bonami, Gérard Cornuéjols, Andrea Lodi, and François Margot. "A feasibility pump for mixed integer nonlinear programs". *Mathematical Programming* 119.2 (2009), pp. 331–352.

[144] Timo Berthold. "RENS — The optimal rounding". *Mathematical Programming Computation* 6.1 (2014), pp. 33–54.

[145] Jan Kronqvist, David E Bernal, Andreas Lundell, and Tapio Westerlund. "A center-cut algorithm for quickly obtaining feasible solutions and solving convex MINLP problems". *Computers & Chemical Engineering* 122 (2019), pp. 105–113.

[146] Timo Berthold. "Heuristic algorithms in global MINLP solvers". PhD thesis. Technische Universität Berlin, 2014, p. 366.

[147] Claudia D'Ambrosio, Antonio Frangioni, Leo Liberti, and Andrea Lodi. "A storm of feasibility pumps for nonconvex MINLP". *Mathematical programming* 136.2 (2012), pp. 375–402.

[148] David E Bernal, Stefan Vigerske, Francisco Trespalacios, and Ignacio E Grossmann. "Improving the performance of DICOPT in convex MINLP problems using a feasibility pump". *Preprint, Optimization Online* (2017).

[149] *Gurobi optimizer reference manual*. Gurobi Optimization, LLC. 2018.

[150] FICO. *Xpress-optimizer reference manual*. 2017.

[151] Andrew Makhorin. *GLPK (GNU linear programming kit)*. 2008.

[152] John Forrest. *Cbc User's guide*. 2005.

[153] Robin Lougee-Heimer. "The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community". *IBM Journal of Research and Development* 47.1 (2003), pp. 57–66.

[154] Arne Drud. "CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems". *Mathematical Programming* 31.2 (1985), pp. 153–191. DOI: 10.1007/BF02591747.

[155] Richard H Byrd, Jorge Nocedal, and Richard A Waltz. "KNITRO: An integrated package for nonlinear optimization". In: *Large-scale nonlinear optimization*. Springer, 2006, pp. 35–59.

[156] Erling D Andersen and Knud D Andersen. *The MOSEK optimization software*. EKA Consulting ApS, Denmark. 2000.

[157] Philip E Gill, Walter Murray, and Michael A Saunders. "SNOPT: An SQP algorithm for large-scale constrained optimization". *SIAM review* 47.1 (2005), pp. 99–131.

[158] Andreas Wächter and Lorenz T Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". *Mathematical programming* 106.1 (2006), pp. 25–57.

[159] Lorenz T Biegler. *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.

[160] Toni Lastusilta. *GAMS MINLP solver comparisons and some improvements to the AlphaECP algorithm*. PhD thesis, bo Akademi University. 2011.

[161] Ruth Misener and Christodoulos A Floudas. "ANTIGONE: Algorithms for continuous/integer global optimization of nonlinear equations". *Journal of Global Optimization* 59.2-3 (2014), pp. 503–526.

[162] Ruth Misener and Christodoulos A Floudas. "GloMIQO: Global mixed-integer quadratic optimizer". *Journal of Global Optimization* 57.1 (2013), pp. 3–50.

[163] Marcel Hunting. *The AIMMS outer approximation algorithm for MINLP*. Tech. rep. AIMMS B.V., 2011.

[164] Marcel Roelofs and Johannes Bisschop. *AIMMS – the language reference*. 2018.

[165] Hong S Ryoo and Nikolaos V Sahinidis. "A branch-and-reduce approach to global optimization". *Journal of global optimization* 8.2 (1996), pp. 107–138.

[166] Kai Zhou, Mustafa R Kılınç, Xi Chen, and Nikolaos V Sahinidis. "An efficient strategy for the activation of MIP relaxations in a multicore global MINLP solver". *Journal of Global Optimization* 70.3 (2018), pp. 497–516.

[167] Mustafa R Kılınç and Nikolaos V Sahinidis. "Exploiting integrality in the global optimization of mixed-integer nonlinear programming problems with BARON". *Optimization Methods and Software* 33.3 (2018), pp. 540–562.

[168] Aida Khajavirad and Nikolaos V. Sahinidis. "A hybrid LP/NLP paradigm for global optimization relaxations". *Mathematical Programming Computation* 10.3 (2018), pp. 383–421.

[169] Pierre Bonami and Jon Lee. "BONMIN user's manual". *Numer Math* 4 (2007), pp. 1–32.

[170] Pietro Belotti, Jon Lee, Leo Liberti, François Margot, and Andreas Wächter. "Branching and bounds tightening techniques for non-convex MINLP". *Optimization Methods and Software* 24 (2009), pp. 597–634.

[171] Pietro Belotti. *Couenne: a user's manual*. 2010.

[172] Ole Kröger, Carleton Coffrin, Hassan Hijazi, and Harsha Nagarajan. *Juniper: An Open-Source Nonlinear Branch-and-Bound Solver in Julia*. arXiv preprint: 1804.07332. 2018.

[173] Artelys. *Artelys Knitro User's Manual*. 2018.

[174] Youdong Lin and Linus Schrage. "The global solver in the LINDO API". *Optimization Methods & Software* 24.4-5 (2009), pp. 657–668.

[175] LINDO Systems Inc. *LINDO User's Manual*. 2017.

[176] Wendel Melo, Marcia Fampa, and Fernanda Raupp. "An overview of MINLP algorithms and their implementation in Muriqui Optimizer". *Annals of Operations Research* (2018), pp. 1–25.

[177] Wendel Melo, Marcia Fampa, and Fernanda Raupp. *First steps to solve MINLP problems with Muriqui Optimizer*. [Accessed 18-May-2018]. 2018.

[178] Chris Coey, Miles Lubin, and Juan Pablo Vielma. *Outer Approximation With Conic Certificates For Mixed-Integer Convex Problems*. arXiv preprint:1808.05290. 2018.

[179] GAMS. *Branch-and-Cut-and-Heuristic Facility*. [Accessed 18-May-2018]. 2018.

[180] Tobias Achterberg. "SCIP: solving constraint integer programs". *Mathematical Programming Computation* 1.1 (2009), pp. 1–41.

[181] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. *The SCIP Optimization Suite 6.0*. Technical Report. Optimization Online, 2018.

[182] Stefan Vigerske and Ambros Gleixner. "SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework". *Optimization Methods and Software* 33.3 (2018), pp. 563–593.

[183] Andreas Lundell, Jan Kronqvist, and Tapio Westerlund. "The Supporting Hyperplane Optimization Toolkit – A Polyhedral Outer Approximation Based Convex

MINLP Solver Utilizing a Single Branching Tree Approach". *Preprint, Optimization Online* (2018).

[184] Andreas Lundell, Jan Kronqvist, and Tapio Westerlund. "SHOT – A global solver for convex MINLP in Wolfram Mathematica". In: *Computer Aided Chemical Engineering*. Vol. 40. Elsevier, 2017, pp. 2137–2142.

[185] FICO. *FICO Xpress-SLP manual*. 2017.

[186] George L Nemhauser, Martin WP Savelsbergh, and Gabriele C Sigismondi. "MINTO, a MIxed INTeger optimizer". *Operations Research Letters* 15.1 (1994), pp. 47–58.

[187] Roger Fletcher and Sven Leyffer. *User manual for filterSQP*. Numerical Analysis Report NA/181, Department of Mathematics, University of Dundee. 1998.

[188] Tapio Westerlund. *User's Guide for GAECP, An Interactive Solver for Generalized Convex MINLP-Problems using Cutting Plane and Supporting Hyperplane Techniques*. 2018.

[189] Hande Y Benson. "Mixed integer nonlinear programming using interior-point methods". *Optimization Methods and Software* 26.6 (2011), pp. 911–931.

[190] David E Bernal, Qi Chen, Felicity Gong, and Ignacio E Grossmann. "Mixed-integer nonlinear decomposition toolbox for Pyomo (MindtPy)". In: *Computer Aided Chemical Engineering*. Vol. 44. Elsevier, 2018, pp. 895–900. DOI: 10.1016/B978-0-444-64241-7.50144-0.

[191] Sven Leyffer. *User manual for MINLP BB*. Tech. rep. University of Dundee numerical analysis report, 1999.

[192] Kenneth Holmström. "The TOMLAB optimization environment in Matlab". *AMO – Advanced Modeling and Optimization* 1.1 (1999).

[193] CA Schweiger and CA Floudas. "MINOPT: A modeling language and algorithmic framework for linear, mixed-integer, nonlinear, dynamic, and mixed-integer nonlinear optimization". *Princeton University,(http://titan. princeton. edu/MINOPT)* (1998).

[194] Oliver Exler and Klaus Schittkowski. "A trust region SQP algorithm for mixed-integer nonlinear programming". *Optimization Letters* 1.3 (2007), pp. 269–280.

[195] Ivo Nowak, Norman Breitfeld, Eligius MT Hendrix, and Grégoire Njacheun-Njanzoua. "Decomposition-based Inner-and Outer-Refinement Algorithms for Global Optimization". *Journal of Global Optimization* (2018), pp. 1–17.

[196] Sébastien Le Digabel. "Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm". *ACM Transactions on Mathematical Software (TOMS)* 37.4 (2011), p. 44.

[197] Harsha Nagarajan, Mowen Lu, Site Wang, Russell Bent, and Kaarthik Sundar. *An Adaptive, Multivariate Partitioning Algorithm for Global Optimization of Nonconvex Programs*. arXiv preprint: 1707.02514. 2017.

[198] Ivo Nowak, Hernán Alperin, and Stefan Vigerske. "LaGO—an object oriented library for solving MINLPs". In: *International Workshop on Global Optimization and Constraint Satisfaction*. Springer. 2002, pp. 32–42.

[199] Mohit Tawarmalani and Nikolaos V Sahinidis. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: Theory, algorithms, software, and applications*. Vol. 65. Springer Science & Business Media, 2002.

[200] Pierre Bonami, Mustafa Kilinç, and Jeff Linderoth. "Algorithms and software for convex mixed integer nonlinear programs". In: *Mixed integer nonlinear programming*. Springer, 2012, pp. 1–39.

[201] HSL. *A collection of Fortran codes for large-scale scientific computation*. http://www.hsl.rl.ac.uk. 2018.

[202] Michael R. Bussieck, Steven P. Dirkse, and Stefan Vigerske. "PAVER 2.0: an open source environment for automated performance analysis of benchmarking data". *Journal of Global Optimization* 59.2 (2014), pp. 259–275.

[203] MINLPLib. *Mixed-Integer Nonlinear Programming Library*. [Accessed 27-May-2018]. 2018.

[204] Elizabeth D. Dolan and Jorge J. Moré. "Benchmarking optimization software with performance profiles". *Mathematical Programming, Series B* 91.2 (2002), pp. 201–213. DOI: 10.1007/s101070100263. arXiv: 0102001 [cs].

[205] Nicholas Gould and Jennifer Scott. "A note on performance profiles for benchmarking software". *ACM Transactions on Mathematical Software (TOMS)* 43.2 (2016), p. 15.

[206] David E. Bernal, Stefan Vigerske, Francisco Trespalacios, and Ignacio E. Grossmann. "Improving the performance of DICOPT in convex MINLP problems using a feasibility pump". *Optimization Methods and Software* 0.0 (2019), pp. 1–20. DOI: 10.1080/10556788.2019.1641498.

[207] Ignacio E. Grossmann and Zdravko Kravanja. "Mixed-Integer Nonlinear Programming: A Survey of Algorithms and Applications". In: *Large-Scale Optimization with Applications: Part II: Optimal Design and Control*. Ed. by Lorenz T. Biegler, Thomas F. Coleman, Andrew R. Conn, and Fadil N. Santosa. Springer New York, 1997, pp. 73–100. DOI: 10.1007/978-1-4612-1960-6_5.

[208] Ignacio E. Grossmann and Jon Lee. "CyberInfrastructure for Mixed-Integer Nonlinear Programming". *SIAG/OPT Views-and-News* 22.1 (2011), pp. 8–12.

[209] Matteo Fischetti, Fred Glover, and Andrea Lodi. "The feasibility pump". *Mathematical Programming* 104.1 (2005), pp. 91–104. DOI: 10.1007/s10107-004-0570-3.

[210] Pierre Bonami, Gérard Cornuéjols, Andrea Lodi, and François Margot. "A Feasibility Pump for mixed integer nonlinear programs". *Mathematical Programming* 119.2 (2009), pp. 331–352. DOI: 10.1007/s10107-008-0212-2.

[211] Roger Fletcher and Sven Leyffer. "Solving mixed integer nonlinear programs by outer approximation". *Mathematical Programming* 66.1-3 (1994), pp. 327–349. DOI: 10.1007/BF01581153.

[212] Egon Balas and Robert G. Jeroslow. "Strengthening cuts for mixed integer programs". *European Journal of Operational Research* 4.4 (1980), pp. 224–234. DOI: 10.1016/0377-2217(80)90106-X.

[213] G. R. Kocis and I. E. Grossmann. "Computational experience with DICOPT solving MINLP problems in process systems engineering". *Computers & Chemical Engineering* 13.3 (1989), pp. 307–315. DOI: 10.1016/0098-1354(89)85008-2.

[214] Ignacio E Grossmann, Jagadisan Viswanathan, Aldo Vecchietti, Ramesh Raman, Erwin Kalvelagen, et al. "GAMS/DICOPT: A discrete continuous optimization package". *GAMS Corporation Inc* 37 (2002), p. 55.

[215] Livio Bertacco, Matteo Fischetti, and Andrea Lodi. "A feasibility pump heuristic for general mixed-integer problems". *Discrete Optimization* 4.1 (2007), pp. 63–76. DOI: 10.1016/j.disopt.2006.10.001.

[216] Timo Berthold. "Heuristic algorithms in global MINLP solvers". PhD thesis. TU Berlin, 2014.

[217] Claudia D'Ambrosio, Antonio Frangioni, Leo Liberti, and Andrea Lodi. "A storm of feasibility pumps for nonconvex MINLP". *Mathematical Programming* 136.2 (2012), pp. 375–402. DOI: 10.1007/s10107-012-0608-x.

[218] Björn Geißler, Antonio Morsi, Lars Schewe, and Martin Schmidt. "Penalty Alternating Direction Methods for Mixed-Integer Optimization: A New View on Feasibility Pumps". *SIAM Journal on Optimization* 27.3 (2017), pp. 1611–1636. DOI: 10.1137/16M1069687.

[219] Tobias Achterberg and Timo Berthold. "Improving the feasibility pump". *Discrete Optimization* 4.1 (2007), pp. 77–86. DOI: 10.1016/j.disopt.2006.10.004.

[220] Stefan Vigerske. "MINLPLib 2". In: *Proceedings of the XII global optimization workshop MAGO 2014*. Ed. by L. G. Casado, I. García, and E. M. T. Hendrix. 2014, pp. 137–140.

[221] Michael R. Bussieck, Steven P. Dirkse, and Stefan Vigerske. "PAVER 2.0: an open source environment for automated performance analysis of benchmarking data". *Journal of Global Optimization* 59.2-3 (2014), pp. 259–275. DOI: 10.1007/s10898-013-0131-5.

[222] Russell D Meller and Kai-Yin Gau. "The facility layout problem: Recent and emerging trends and perspectives". *Journal of Manufacturing Systems* 15.5 (1996), pp. 351–366. DOI: 10.1016/0278-6125(96)84198-7. arXiv: arXiv:1011.1669v3.

[223] Ignacio Castillo, Joakim Westerlund, Stefan Emet, and Tapio Westerlund. "Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods". *Computers & Chemical Engineering* 30.1 (2005), pp. 54–69. DOI: 10.1016/j.compchemeng.2005.07.012.

[224] Jan Kronqvist, Andreas Lundell, and Tapio Westerlund. "A center-cut algorithm for solving convex mixed-integer nonlinear programming problems". In: *Computer Aided Chemical Engineering*. Vol. 40. Elsevier, 2017, pp. 2131–2136.

[225] Jack Elzinga and Thomas G Moore. "A central cutting plane algorithm for the convex programming problem". *Mathematical Programming* 8.1 (1975), pp. 134–145.

[226] Morton Slater et al. *Lagrange multipliers revisited*. Tech. rep. Cowles Foundation for Research in Economics, Yale University, 1959.

[227] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[228] George L Nemhauser and WB Widhelm. "A modified linear program for columnar methods in mathematical programming". *Operations Research* 19.4 (1971), pp. 1051–1060.

[229] Eligius MT Hendrix, Carmen J Mecking, and Theo HB Hendriks. "Finding robust solutions for product design problems". *European Journal of Operational Research* 92.1 (1996), pp. 28–36.

[230] Toni Lastusilta, Michael R Bussieck, and Tapio Westerlund. "An experimental study of the GAMS/AlphaECP MINLP solver". *Industrial & Engineering Chemistry Research* 48.15 (2009), pp. 7337–7345.

[231] GAMSWorld. *Mixed-Integer Nonlinear Programming Library*. [Online; accessed 27-December-2017]. 2017.

[232] Ignacio Castillo, Joakim Westerlund, Stefan Emet, and Tapio Westerlund. "Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods". *Computers & Chemical Engineering* 30.1 (2005), pp. 54–69.

[233] Nicolas Sawaya. *Reformulations, relaxations and cutting planes for generalized disjunctive programming*. Vol. 67. 12. Carnegie Mellon University, 2006.

[234] Iiro Harjunkoski, Tapio Westerlund, Ray Pörn, and Hans Skrifvars. "Different transformations for solving non-convex trim-loss problems by MINLP". *European Journal of Operational Research* 105.3 (1998), pp. 594–603.

[235] Lijie Su, Lixin Tang, David E Bernal, and Ignacio E Grossmann. "Improved quadratic cuts for convex mixed-integer nonlinear programs". *Computers & Chemical Engineering* 109 (2018), pp. 77–95.

[236] Carlos A. Méndez, Jaime Cerdá, Ignacio E. Grossmann, Iiro Harjunkoski, and Marco Fahl. "State-of-the-art review of optimization methods for short-term scheduling of batch processes". *Computers & Chemical Engineering* 30.6-7 (2006), pp. 913–946. DOI: 10.1016/j.compchemeng.2006.02.008.

[237] Lixin Tang, Peter B Luh, Jiyin Liu, and Lei Fang. "Steel-making process scheduling using Lagrangian relaxation". *International Journal of Production Research* 40.1 (2002), pp. 55–70.

[238] Mohit Tawarmalani and Nikolaos V Sahinidis. "Global optimization of mixed-integer nonlinear programs: A theoretical and computational study". *Mathematical programming* 99.3 (2004), pp. 563–591.

[239] Claire S Adjiman, Stefan Dallwig, Christodoulos A Floudas, and Arnold Neumaier. "A global optimization method, αBB, for general twice-differentiable constrained NLPs — I. Theoretical advances". *Computers & Chemical Engineering* 22.9 (1998), pp. 1137–1158.

[240] Padmanaban Kesavan, Russell J Allgor, Edward P Gatzke, and Paul I Barton. "Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs". *Mathematical Programming* 100.3 (2004), pp. 517–535.

[241] Xiang Li, Asgeir Tomasgard, and Paul I Barton. "Nonconvex generalized Benders decomposition for stochastic separable mixed-integer nonlinear programs". *Journal of optimization theory and applications* 151.3 (2011), pp. 425–454.

[242] Ioannis P Androulakis, Costas D Maranas, and Christodoulos A Floudas. "αBB: A global optimization method for general constrained nonconvex problems". *Journal of Global Optimization* 7.4 (1995), pp. 337–363.

[243] Lijie Su, Lixin Tang, and Ignacio E. Grossmann. "Computational strategies for improved MINLP algorithms". *Computers and Chemical Engineering* 75 (2015), pp. 40–48. DOI: 10.1016/j.compchemeng.2015.01.015.

[244] Christian Bliek1ú, Pierre Bonami, and Andrea Lodi. "Solving mixed-integer quadratic programming problems with IBM-CPLEX: a progress report". In: *Proceedings of the twenty-sixth RAMP symposium*. 2014, pp. 16–17.

[245] Christoph Buchheim and Long Trieu. "Quadratic outer approximation for convex integer programming with box constraints". In: *International Symposium on Experimental Algorithms*. Springer. 2013, pp. 224–235.

[246] Timo Berthold, Stefan Heinz, and Stefan Vigerske. "Extending a CIP Framework to Solve MIQCP s". In: *Mixed integer nonlinear programming*. Springer, 2012, pp. 427–444.

[247] G.B. Folland. *Advanced Calculus*. Featured Titles for Advanced Calculus Series. Prentice Hall, 2002.

[248] Michael R Bussieck and Alex Meeraus. "General algebraic modeling system (GAMS)". In: *Modeling languages in mathematical optimization*. Springer, 2004, pp. 137–157.

[249] Lijie Su, Lixin Tang, David E. Bernal, and Ignacio E. Grossmann. "Improved quadratic cuts for convex mixed-integer nonlinear programs". *Computers & Chemical Engineering* 109 (2018), pp. 77–95. DOI: 10.1016/j.compchemeng.2017.10.011.

[250] Michael R Bussieck and Arne Drud. *SBB: A new solver for mixed integer nonlinear programming*. GAMS. 2001. URL: https://old.gams.com/presentations/present_sbb.pdf.

[251] Jon Lee and Sven Leyffer. *Mixed Integer Nonlinear Programming*. Ed. by Jon Lee and Sven Leyffer. Vol. 154. Springer Science & Business Media, 2011.

[252] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Vol. 87. Springer, 2004.

[253] Dick den Hertog, J Kaliski, C Roos, and T Terlaky. "A logarithmic barrier cutting plane method for convex programming". *Annals of Operations Research* 58.2 (1995), pp. 67–98.

[254] Adil Bagirov, Napsu Karmitsa, and Marko M Mäkelä. *Introduction to Nonsmooth Optimization: Theory, Practice and Software*. Springer, 2014.

[255] Sofia Zaourar and Jérôme Malick. "Quadratic stabilization of Benders decomposition". working paper or preprint. 2014.

[256] Welington de Oliveira. "Regularized optimization methods for convex MINLP problems". *TOP* 24.3 (2016), pp. 665–692.

[257] Claude Lemaréchal, Arkadii Nemirovskii, and Yurii Nesterov. "New variants of bundle methods". *Mathematical Programming* 69.1-3 (1995), pp. 111–147.

[258] Krzysztof C Kiwiel. "Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities". *Mathematical Programming* 69.1-3 (1995), pp. 89–109.

[259] Zhou Wei and M. Montaz Ali. "Outer Approximation Algorithm for One Class of Convex Mixed-Integer Nonlinear Programming Problems with Partial Differentiability". *Journal of Optimization Theory and Applications* 167.2 (2015), pp. 644–652. DOI: 10.1007/s10957-015-0715-y.

[260] Philip Wolfe. "A duality theorem for non-linear programming". *Quarterly of applied mathematics* 19.3 (1961), pp. 239–244.

[261] Semyon Aranovich Gershgorin. "Uber die Abgrenzung der Eigenwerte einer Matrix". *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na* 6 (1931), pp. 749–754.

[262] Jan Kronqvist, Andreas Lundell, and Tapio Westerlund. "Reformulations for utilizing separability when solving convex MINLP problems". *Submitted for publication* (2017).

[263] David E Bernal, Stefan Vigerske, Francisco Trespalacios, and Ignacio E Grossmann. "Improving the performance of DICOPT in convex MINLP problems using a feasibility pump". *Optimization Methods and Software* 35.1 (2020), pp. 171–190.

[264] Pavlo Muts, Ivo Nowak, and Eligius MT Hendrix. "The decomposition-based outer approximation algorithm for convex mixed-integer nonlinear programming". *Journal of Global Optimization* (2020), pp. 1–22.

[265] Massimo De Mauri, Joris Gillis, Jan Swevers, and Goele Pipeleers. "A proximal-point outer approximation algorithm". *Computational Optimization and Applications* 77.3 (2020), pp. 755–777.

[266] A Lundell, J Kronqvist, and T Westerlund. "The supporting hyperplane optimization toolkit—a polyhedral outer approximation based convex MINLP solver utilizing a single branching tree approach. Optimization". *Preprint, Optimization Online* (2018).

[267] Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, 2000.

[268] Adriano Delfino and Welington de Oliveira. "Outer-approximation algorithms for nonsmooth convex MINLP problems". *Optimization* 67.6 (2018), pp. 797–819.

[269] Mohit Tawarmalani and Nikolaos V Sahinidis. "A polyhedral branch-and-cut approach to global optimization". *Mathematical programming* 103.2 (2005), pp. 225–249.

[270] Paul T Boggs and Jon W Tolle. "Sequential quadratic programming". *Acta numerica* 4 (1995), pp. 1–51.

[271] Erwin Kreyszig. *Introductory functional analysis with applications*. Vol. 1. wiley New York, 1978.

[272] Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2013.

[273] William E Hart, Carl D Laird, Jean-Paul Watson, David L Woodruff, Gabriel A Hackebeil, Bethany L Nicholson, John D Siirola, et al. *Pyomo-optimization modeling in python*. Vol. 67. Springer, 2017.

[274] HSL. *A collection of Fortran codes for large scale scientific computation*. 2007.

[275] Nicolas Sawaya and Ignacio E Grossmann. *Reformulations, relaxations and cutting planes for linear generalized disjunctive programming*. 2008.

[276] Mohit Tawarmalani and Nikolaos V Sahinidis. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*. Vol. 65. Springer Science & Business Media, 2013.

[277] Shih-Ping Han. “Superlinearly convergent variable metric algorithms for general nonlinear programming problems”. *Mathematical Programming* 11.1 (1976), pp. 263–282.

[278] Leo Liberti, Sonia Cafieri, and Fabien Tarissan. “Reformulations in mathematical programming: A computational approach”. In: *Foundations of Computational Intelligence Volume 3*. Springer, 2009, pp. 153–234.

[279] Robert Chares. “Cones and interior-point algorithms for structured convex optimization involving powers andexponentials”. PhD thesis. Ph. D. Thesis, UCL-Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 2009.

[280] Hande Y Benson and Robert J Vanderbei. “Solving problems with semidefinite and related constraints using interior-point methods for nonlinear programming”. *Mathematical Programming* 95.2 (2003), pp. 279–302.

[281] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of convex analysis*. Springer Science & Business Media, 2004.

[282] Aldo Vecchietti and Ignacio E Grossmann. “LOGMIP: a disjunctive 0–1 non-linear optimizer for process system models”. *Computers & chemical engineering* 23.4-5 (1999), pp. 555–565.

[283] Egon Balas. “Disjunctive programming”. In: *Annals of Discrete Mathematics*. Vol. 5. Elsevier, 1979, pp. 3–51.

[284] Sebastián Ceria and João Soares. “Convex programming for disjunctive convex optimization”. *Mathematical Programming* 86.3 (1999), pp. 595–614.

[285] Egon Balas. “Disjunctive programming and a hierarchy of relaxations for discrete optimization problems”. *SIAM Journal on Algebraic Discrete Methods* 6.3 (1985), pp. 466–486.

[286] Egon Balas. "Disjunctive programming: Properties of the convex hull of feasible points". *Discrete Applied Mathematics* 89.1-3 (1998), pp. 3–44.

[287] Hassan Hijazi, Pierre Bonami, and Adam Ouorou. "An outer-inner approximation for separable mixed-integer nonlinear programs". *INFORMS Journal on Computing* 26.1 (2014), pp. 31–44.

[288] Jan Kronqvist, Andreas Lundell, and Tapio Westerlund. "Reformulations for utilizing separability when solving convex MINLP problems". *Journal of Global Optimization* 71.3 (2018), pp. 571–592.

[289] Robert G Jeroslow. "Representability in mixed integer programmiing, I: Characterization results". *Discrete Applied Mathematics* 17.3 (1987), pp. 223–243.

[290] Ramesh Raman and Ignacio E Grossmann. "Modelling and computational techniques for logic based integer programming". *Computers & Chemical Engineering* 18.7 (1994), pp. 563–578.

[291] Ignacio E Grossmann and Juan P Ruiz. "Generalized disjunctive programming: A framework for formulation and alternative algorithms for MINLP optimization". In: *Mixed Integer Nonlinear Programming*. Springer, 2012, pp. 93–115.

[292] Nicolas Sawaya and Ignacio Grossmann. "A hierarchy of relaxations for linear generalized disjunctive programming". *European Journal of Operational Research* 216.1 (2012), pp. 70–82.

[293] Juan P Ruiz and Ignacio E Grossmann. "A hierarchy of relaxations for nonlinear convex generalized disjunctive programming". *European Journal of Operational Research* 218.1 (2012), pp. 38–47.

[294] H Paul Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

[295] Juan P Ruiz and Ignacio E Grossmann. "Global optimization of non-convex generalized disjunctive programs: a review on reformulations and relaxation techniques". *Journal of Global Optimization* 67.1-2 (2017), pp. 43–58.

[296] Francisco Trespalacios and Ignacio E Grossmann. "Cutting plane algorithm for convex generalized disjunctive programs". *INFORMS Journal on Computing* 28.2 (2016), pp. 209–222.

[297] Alper Atamtürk and Andrés Gómez. "Strong formulations for quadratic optimization with M-matrices and indicator variables". *Mathematical Programming* 170.1 (2018), pp. 141–176.

[298] Venkat Chandrasekaran and Parikshit Shah. "Relative entropy optimization and its applications". *Mathematical Programming* 161.1-2 (2017), pp. 1–32.

[299] Laurent El Ghaoui and Hervé Lebret. "Robust solutions to least-squares problems with uncertain data". *SIAM Journal on matrix analysis and applications* 18.4 (1997), pp. 1035–1064.

[300] Michael R Bussieck and Arne Drud. "SBB: A new solver for mixed integer nonlinear programming". *Talk, OR* (2001).

[301] *CMU-IBM Cyber-Infrastructure for MINLP*.

[302] Patrick Flaherty, Pitchaya Wiratchotisatian, Ji Ah Lee, Zhou Tang, and Andrew C Trapp. "MAP Clustering under the Gaussian Mixture Model via Mixed Integer Nonlinear Optimization". *arXiv preprint arXiv:1911.04285* (2019).

[303] Yihua Chen and Maya R Gupta. "Em demystified: An expectation-maximization tutorial". In: *Electrical Engineering*. Citeseer. 2010.

[304] Yu-Feng Li, Ivor W Tsang, James T Kwok, and Zhi-Hua Zhou. "Convex and scalable weakly labeled SVMs." *Journal of Machine Learning Research* 14.7 (2013).

[305] Dimitri J Papageorgiou and Francisco Trespalacios. "Pseudo basic steps: bound improvement guarantees from Lagrangian decomposition in convex disjunctive programming". *EURO Journal on Computational Optimization* 6.1 (2018), pp. 55–83.

[306] Jan Kronqvist, Ruth Misener, and Calvin Tsay. "Between steps: Intermediate relaxations between big-M and convex hull formulations". *arXiv preprint arXiv:2101.12708* (2021).

[307] G.R. Kocis and I.E. Grossmann. "A modelling and decomposition strategy for the minlp optimization of process flowsheets". *Computers & Chemical Engineering* 13.7 (1989), pp. 797–819. DOI: 10.1016/0098-1354(89)85053-7.

[308] Francisco Trespalacios and Ignacio E Grossmann. "Improved Big-M reformulation for generalized disjunctive programs". *Computers & Chemical Engineering* 76 (2015), pp. 98–103.

[309] Jennifer R Jackson and Ignacio E Grossmann. "High-level optimization model for the retrofit planning of process networks". *Industrial & engineering chemistry research* 41.16 (2002), pp. 3762–3770.

[310] Ashutosh Mahajan and Todd Munson. "Exploiting second-order cone structure for global optimization" (2010).

[311] Rodolfo Quintero, David E. Bernal, Tamás Terlaky, and Luis F Zuluaga. "Characterization of QUBO reformulations for the maximum k-colorable subgraph problem". *arXiv preprint arXiv:2101.09462* (2021).

[312] I. Semeniuk. "Understanding the Quantum Computing Revolution". *The Globe and Mail* (Sept., 2017). https://www.theglobeandmail.com/report-on-business/rob-magazine/quantum-computing-technology-explained/article36397793/.

[313] S. Cole. "Ready or not, the quantum computing revolution is here". *Military embedded systems* (Mar. 20, 2018). http://mil-embedded.com/articles/ready-not-quantum-computing-revolution-here/.

[314] C. Metz. "The Next Tech Talent Shortage: Quantum Computing Researchers". *New York Times* (Oct. 21, 2018). https://www.nytimes.com/2018/10/21/technology/quantum-computing-jobs-immigration-visas.html.

[315] Ashley Montanaro. "Quantum algorithms: an overview". *npj Quantum Information* 2.1 (2016), pp. 1–8.

[316] Richard M Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[317] Stephen G Brush. "History of the Lenz-Ising model". *Reviews of Modern Physics* 39.4 (1967), p. 883.

[318] Satya Pal Singh. "The Ising Model: Brief Introduction and Its Application". In: *Solid State Physics-Metastable, Spintronics Materials and Mechanics of Deformable Bodies-Recent Progress*. IntechOpen, 2020.

[319] Barry A Cipra. "The Ising model is NP-complete". *SIAM News* 33.6 (2000), pp. 1–3.

[320] Foad Mahdavi Pajouh, Balabhaskar Balasundaram, and Oleg A Prokopyev. "On characterization of maximal independent sets via quadratic optimization". *Journal of Heuristics* 19.4 (2013), pp. 629–644.

[321] Vicky Choi. "Minor-embedding in adiabatic quantum computation: I. The parameter setting problem". *Quantum Information Processing* 7.5 (2008), pp. 193–209.

[322] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, Trevor Lanting, Firas Hamze, Neil Dickson, Richard Harris, Andrew J Berkley, Jan Johansson, Paul

Bunyk, et al. “Quantum annealing with manufactured spins”. *Nature* 473.7346 (2011), pp. 194–198.

[323] Edward Farhi and Aram W Harrow. “Quantum supremacy through the quantum approximate optimization algorithm”. *arXiv preprint arXiv:1602.07674* (2016).

[324] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G Rieffel. “Quantum approximate optimization algorithm for MaxCut: A fermionic view”. *Physical Review A* 97.2 (2018), p. 022304.

[325] Andrew D King and Catherine C McGeoch. “Algorithm engineering for a quantum annealing platform”. *arXiv preprint arXiv:1410.2628* (2014).

[326] Giacomo Nannicini. “Performance of hybrid quantum-classical variational heuristics for combinatorial optimization”. *Physical Review E* 99.1 (2019), p. 013304.

[327] Cristian S Calude, Michael J Dinneen, and Richard Hua. “QUBO formulations for the graph isomorphism problem and related problems”. *Theoretical Computer Science* 701 (2017), pp. 54–69.

[328] Eleanor G Rieffel, Davide Venturelli, Bryan O’Gorman, Minh B Do, Elicia M Prystay, and Vadim N Smelyanskiy. “A case study in programming a quantum annealer for hard operational planning problems”. *Quantum Information Processing* 14.1 (2015), pp. 1–36.

[329] Davide Venturelli, D Marchand, and Galo Rojo. “Job shop scheduling solver based on quantum annealing”. In: *Proc. of ICAPS-16 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling (COPLAS)*. 2016, pp. 25–34.

[330] Tobias Stollenwerk, Bryan O’Gorman, Davide Venturelli, Salvatore Mandrà, Olga Rodionova, Hokkwan Ng, Banavar Sridhar, Eleanor Gilbert Rieffel, and Rupak Biswas. “Quantum annealing applied to de-conflicting optimal trajectories for air

traffic management". *IEEE transactions on intelligent transportation systems* 21.1 (2019), pp. 285–297.

[331] Christos Papalitsas, Theodore Andronikos, Konstantinos Giannakis, Georgia Theocharopoulou, and Sofia Fanarioti. "A QUBO model for the traveling salesman problem with time windows". *Algorithms* 12.11 (2019), p. 224.

[332] Fred Glover, Gary Kochenberger, and Yu Du. "Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models". *4OR* 17.4 (2019), pp. 335–371.

[333] Tomáš Vyskočil, Scott Pakin, and Hristo N Djidjev. "Embedding inequality constraints for quantum annealing optimization". In: *International workshop on quantum technology and optimization problems*. Springer. 2019, pp. 11–22.

[334] Yue Ruan, Samuel Marsh, Xilin Xue, Xi Li, Zhihao Liu, and Jingbo Wang. "Quantum approximate algorithm for NP optimization problems with constraints". *arXiv preprint arXiv:2002.00943* (2020).

[335] Alexander Fowler. "Improved QUBO formulations for D-Wave quantum computing". PhD thesis. University of Auckland, 2017.

[336] Gian Giacomo Guerreschi and Anne Y Matsuura. "QAOA for Max-Cut requires hundreds of qubits for quantum speed-up". *Scientific reports* 9.1 (2019), p. 6903.

[337] Richard Hua and Michael J Dinneen. "Improved QUBO formulation of the graph isomorphism problem". *SN Computer Science* 1.1 (2020), p. 19.

[338] Amit Verma and Mark Lewis. "Optimal quadratic reformulations of fourth degree Pseudo-Boolean functions". *Optimization Letters* 14.6 (2020), pp. 1557–1569.

[339] Amit Verma and Mark Lewis. "Penalty and partitioning techniques to improve performance of QUBO solvers". *Discrete Optimization* (2020), p. 100594.

[340] Olga Kuryatnikova, Renata Sotirov, and Juan Vera. "The maximum k-colorable subgraph problem and related problems". *arXiv preprint arXiv:2001.09644* (2020).

[341] Anand Prabhu Subramanian, Himanshu Gupta, Samir R Das, and Milind M Buddhikot. "Fast spectrum allocation in coordinated dynamic spectrum access based cellular networks". In: *2007 2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks*. IEEE. 2007, pp. 320–330.

[342] Magnús M Halldórsson, Joseph Y Halpern, Li Erran Li, and Vahab S Mirrokni. "On spectrum sharing games". *Distributed computing* 22.4 (2010), pp. 235–248.

[343] Pierre Fouilhoux and A Ridha Mahjoub. "Solving VLSI design and DNA sequencing problems using bipartization of graphs". *Computational Optimization and Applications* 51.2 (2012), pp. 749–781.

[344] Ross Lippert, Russell Schwartz, Giuseppe Lancia, and Sorin Istrail. "Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem". *Briefings in bioinformatics* 3.1 (2002), pp. 23–31.

[345] László Lovász. "On the Shannon capacity of a graph". *IEEE Transactions on Information theory* 25.1 (1979), pp. 1–7.

[346] Piotr Berman and Andrzej Pelc. "Distributed probabilistic fault diagnosis for multiprocessor systems". In: *[1990] Digest of Papers. Fault-Tolerant Computing: 20th International Symposium*. IEEE. 1990, pp. 340–346.

[347] Jean B Lasserre. "A MAX-CUT formulation of 0/1 programs". *Operations Research Letters* 44.2 (2016), pp. 158–164.

[348] Jochen Harant. "Some news about the independence number of a graph". *Discussiones Mathematicae Graph Theory* 20.1 (2000), pp. 71–79.

[349] James Abello, Sergiy Butenko, Panos M Pardalos, and Mauricio GC Resende. "Finding independent sets in a graph using continuous multivariable polynomial formulations". *Journal of Global Optimization* 21.2 (2001), pp. 111–137.

[350] Endre Boros, Peter L Hammer, and Gabriel Tavares. "Local search heuristics for Quadratic Unconstrained Binary Optimization (QUBO)". *Journal of Heuristics* 13.2 (2007), pp. 99–132.

[351] Michel X Goemans and David P Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". *Journal of the ACM (JACM)* 42.6 (1995), pp. 1115–1145.

[352] Svatopluk Poljak and Zsolt Tuza. "Maximum cuts and large bipartite subgraphs". *DIMACS Series* 20 (1995), pp. 181–244.

[353] Gavin E Crooks. "Performance of the quantum approximate optimization algorithm on the maximum cut problem". *arXiv preprint arXiv:1811.08419* (2018).

[354] James King, Sheir Yarkoni, Mayssam M Nevisi, Jeremy P Hilton, and Catherine C McGeoch. "Benchmarking a quantum annealing processor with the time-to-target metric". *arXiv preprint arXiv:1508.05087* (2015).

[355] Immanuel M Bomze, Marco Budinich, Panos M Pardalos, and Marcello Pelillo. "The maximum clique problem". In: *Handbook of Combinatorial Optimization*. Ed. by D. Z. Du and P. M. Pardalos. Kluwer Academic Publisher, 1999, pp. 1–74.

[356] Guillaume Chapuis, Hristo Djidjev, Georg Hahn, and Guillaume Rizk. "Finding maximum cliques on a quantum annealer". In: *Proceedings of the Computing Frontiers Conference*. 2017, pp. 63–70.

[357] Sheir Yarkoni, Aske Plaat, and Thomas Back. "First results solving arbitrarily structured maximum independent set problems using quantum annealing". In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2018, pp. 1–6.

[358] Pawel Wocjan and Thomas Beth. "The 2-local Hamiltonian problem encompasses NP". *International Journal of Quantum Information* 1.03 (2003), pp. 349–357.

[359] Mihalis Yannakakis and Fanica Gavril. "The maximum k-colorable subgraph problem for chordal graphs". *Information Processing Letters* 24.2 (1987), pp. 133–137.

[360] Carsten Lund and Mihalis Yannakakis. "The approximation of maximum subgraph problems". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 1993, pp. 40–51.

[361] Edwin R van Dam and Renata Sotirov. "New bounds for the max-k-cut and chromatic number of a graph". *Linear Algebra and its Applications* 488 (2016), pp. 216–234.

[362] Manoel Campêlo and Ricardo C Corrêa. "A combined parallel lagrangian decomposition and cutting-plane generation for maximum stable set problems". *Electronic Notes in Discrete Mathematics* 36 (2010), pp. 503–510.

[363] Tim Januschowski and Marc E Pfetsch. "The maximum k-colorable subgraph problem and orbitopes". *Discrete Optimization* 8.3 (2011), pp. 478–494.

[364] Jérémie Roland and Nicolas J Cerf. "Quantum search by local adiabatic evolution". *Physical Review A* 65.4 (2002), p. 042308.

[365] Troels F Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V Isakov, David Wecker, John M Martinis, Daniel A Lidar, and Matthias Troyer. "Defining and detecting quantum speedup". *science* 345.6195 (2014), pp. 420–424.

[366] Harmut Neven, Vasil S Denchev, Marshall Drew-Brook, Jiayong Zhang, William G Macready, and Geordie Rose. "NIPS 2009 demonstration: Binary classification using hardware implementation of quantum annealing" (2009).

[367] M Cullimore, Mark J Everitt, MA Ormerod, JH Samson, Richard D Wilson, and Alexandre M Zagoskin. "Relationship between minimum gap and success probability in adiabatic quantum computing". *Journal of Physics A: Mathematical and Theoretical* 45.50 (2012), p. 505305.

[368] *QPU-Specific Anneal Schedules – D-Wave Systems*.

[369] Andrew D King and William Bernoudy. "Performance benefits of increased qubit connectivity in quantum annealing 3-dimensional spin glasses". *arXiv preprint arXiv:2009.12479* (2020).

[370] MHS Amin. "Effect of local minima on adiabatic quantum optimization". *Physical Review Letters* 100.13 (2008), p. 130503.

[371] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. "A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem". *Science* 292.5516 (2001), pp. 472–475.

[372] Mohammad H Amin, Anatoly Yu Smirnov, Neil G Dickson, and Marshall Drew-Brook. "Approximate diagonalization method for large-scale hamiltonians". *Physical Review A* 86.5 (2012), p. 052314.

[373] David E Bernal, Kyle EC Booth, Raouf Dridi, Hedayat Alghassi, Sridhar Tayur, and Davide Venturelli. "Integer programming techniques for minor-embedding in quantum annealers". In: *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer. 2020, pp. 112–129.

[374] Stefanie Zbinden, Andreas Bärtschi, Hristo Djidjev, and Stephan Eidenbenz. "Embedding algorithms for quantum annealers with chimera and pegasus connection topologies". In: *International Conference on High Performance Computing*. Springer. 2020, pp. 187–206.

[375] Stuart Harwood, Claudio Gambella, Dimitar Trenev, Andrea Simonetto, David E Bernal, and Donny Greenberg. "Formulating and Solving Routing Problems on Quantum Computers". *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–17. DOI: 10.1109/TQE.2021.3049230.

[376] Paolo Toth and Daniele Vigo. *Vehicle routing: problems, methods, and applications*. second. SIAM, 2014.

[377] Laurence A. Wolsey and George L. Nemhauser. *Integer and combinatorial optimization*. New York: John Wiley & Sons, 1999.

[378] Dimitri J. Papageorgiou, George L. Nemhauser, Joel Sokol, Myun-Seok Cheon, and Ahmet B. Keha. "MIRPLib–A library of maritime inventory routing problem instances: Survey, core model, and benchmark results". *European Journal of Operational Research* 235.2 (2014), pp. 350–366.

[379] *Review of Maritime Transport*. Tech. rep. UNCTAD/RMT/2018. United Nations, 2018.

[380] Dimitri J Papageorgiou, Myun-Seok Cheon, Stuart Harwood, Francisco Trespalacios, and George L Nemhauser. "Recent progress using matheuristics for strategic maritime inventory routing". In: *Modeling, computing and data handling methodologies for maritime transportation*. Springer, 2018, pp. 59–94.

[381] Panagiotis Kl Barkoutsos, Giacomo Nannicini, Anton Robert, Ivano Tavernelli, and Stefan Woerner. "Improving Variational Quantum Optimization using CVaR". *arXiv preprint arXiv:1907.04769* (2019).

[382] Anthony V Fiacco and Garth P McCormick. *Nonlinear programming: sequential unconstrained minimization techniques*. Vol. 4. Siam, 1990.

[383] Chang-Yu Wang and Duan Li. "Unified theory of augmented Lagrangian methods for constrained global optimization". *Journal of Global Optimization* 44.3 (2009), p. 433.

[384] L. Zhou, S.-T. Wang, S. Choi, H. Pichler, and M. D. Lukin. "Quantum Approximate Optimization Algorithm: Performance, Mechanism, and Implementation on Near-Term Devices". *arXiv: 1812.01041* (2019).

[385] A. Gilliam, S. Woerner, and C. Gonciulea. "Grover Adaptive Search for Constrained Polynomial Binary Optimization". *arXiv: 1912.04088* (2019).

[386] F.G.S L. Brandão, R. Kueng, and D. Stilck França. "Faster quantum and classical SDP approximations for quadratic binary optimization". *arXiv: 1909.04613* (2019).

[387] Lee Braine, Daniel J Egger, Jennifer Glick, and Stefan Woerner. "Quantum Algorithms for Mixed Binary Optimization applied to Transaction Settlement". *arXiv preprint arXiv:1910.05788* (2019).

[388] Claudio Gambella and Andrea Simonetto. "Multi-block ADMM Heuristics for Mixed-Binary Optimization on Classical and Quantum Computers". *arXiv preprint arXiv:2001.02069* (2020).

[389] Z. Bian, F. Chudak, R. Israel, B. Lackey, W. G. Macready, and A. Roy. "Discrete optimization using quantum annealing on sparse Ising models". *Front. Phys.* (2014).

[390] G. Rosenberg, M. Vazifeh, B. Woods, and E. Haber. "Building an iterative heuristic solver for a quantum annealer". *Computational Optimization and Applications* 65 (2016), pp. 845–869.

[391] H. Karimi, G. Rosenberg, and H. G. Katzgraber. "Effective optimization using sample persistence: A case study on quantum annealers and various Monte Carlo optimization methods". *Physical Review W* 96 (2017).

[392] R. Shaydulin, H. Ushijima-Mwesigwa, C. F. A. Negre, I. Safro, S. M. Mniszewski, and Y. Alexeev. "A Hybrid Approach for Solving Optimization Problems on Small Quantum Computers". *IEEE Computer* 52 (2019).

[393] Sebastian Feld, Christoph Roch, Thomas Gabor, Christian Seidel, Florian Neukart, Isabella Galter, Wolfgang Mauerer, and Claudia Linnhoff-Popien. "A hybrid solution method for the capacitated vehicle routing problem using a quantum annealer". *Frontiers in ICT* 6 (2019), p. 13.

[394] Hirotaka Irie, Goragot Wongpaisarnsin, Masayoshi Terabe, Akira Miki, and Shinichirou Taguchi. "Quantum annealing of vehicle routing problem with time, state and capacity". In: *International Workshop on Quantum Technology and Optimization Problems*. Springer. 2019, pp. 145–156.

[395] Brian Sutton, Kerem Yunus Camsari, Behtash Behin-Aein, and Supriyo Datta. "Intrinsic Optimization Using Stochastic Nanomagnets". *Scientific Reports* 7 (2017).

[396] X. Yin, B. Sedighi, M. Varga, M. Ercsey-Ravasz, Z. Toroczkai, and X. S. Hu. "Efficient Analog Circuits for Boolean Satisfiability". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.1 (2018), pp. 155–167. DOI: 10.1109/TVLSI.2017.2754192.

[397] Peter L. McMahon, Alireza Marandi, Yoshitaka Haribara, Ryan Hamerly, Carsten Langrock, Shuhei Tamate, Takahiro Inagaki, Hiroki Takesue, Shoko Utsunomiya, Kazuyuki Aihara, Robert L. Byer, M. M. Fejer, Hideo Mabuchi, and Yoshihisa Yamamoto. "A fully programmable 100-spin coherent Ising machine with all-to-all con-

nections". *Science* 354.6312 (2016), pp. 614–617. DOI: 10.1126/science.aah5178. eprint: http://science.sciencemag.org/content/354/6312/614.full.pdf.

[398] T. Lanting, A. J. Przybysz, A. Yu. Smirnov, F. M. Spedalieri, M. H. Amin, A. J. Berkley, R. Harris, F. Altomare, S. Boixo, P. Bunyk, N. Dickson, C. Enderud, J. P. Hilton, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, R. Neufeld, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, S. Uchaikin, A. B. Wilson, and G. Rose. "Entanglement in a Quantum Annealing Processor". *Phys. Rev. X* 4 (2 2014), p. 021041. DOI: 10.1103/PhysRevX.4.021041.

[399] Adam Douglass, Andrew D. King, and Jack Raymond. "Constructing SAT Filters with a Quantum Annealer". In: *Theory and Applications of Satisfiability Testing – SAT 2015*. Ed. by Marijn Heule and Sean Weaver. Cham: Springer International Publishing, 2015, pp. 104–120.

[400] Hayato Goto, Kosuke Tatsumura, and Alexander R. Dixon. "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems". *Science advances* 5.4 (2019), eaav2372.

[401] Sanroku Tsukamoto, Motomu Takatsu, Satoshi Matsubara, and Hirotaka Tamura. "An Accelerator Architecture for Combinatorial Optimization Problems". *FUJITSU Science and Technology Journal* 53 (2017), pp. 8–13.

[402] Bettina Heim, Ethan W Brown, Dave Wecker, and Matthias Troyer. "Designing adiabatic quantum optimization: A case study for the traveling salesman problem". *arXiv preprint arXiv:1702.06248* (2017).

[403] Andrew Lucas. "Ising formulations of many NP problems". *Frontiers in Physics* 2 (2014), p. 5.

[404] Brian Kallehauge. "Formulations and exact algorithms for the vehicle routing problem with time windows". *Computers & Operations Research* 35.7 (2008), pp. 2307–2330.

[405] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. "A new optimization algorithm for the vehicle routing problem with time windows". *Operations research* 40.2 (1992), pp. 342–354.

[406] J. Duchi. "Sequential Convex Programming". *Lecture notes for EE364b, Stanford University* (2018).

[407] A. R. Conn, N. I. M. Gould, and P. L. Toint. *Trust Region Methods*. MOS-SIAM Series on Optimization, 2000.

[408] Héctor Abraham *et al. Qiskit: An Open-source Framework for Quantum Computing*. 2019. DOI: 10.5281/zenodo.2562110.

[409] *Qiskit: an open-source quantum computing software development framework.* https://qiskit.org/. 2016.

[410] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M. Chow, and Jay M. Gambetta. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets". *Nature* 549 (2017), p. 242.

[411] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G Rieffel, Davide Venturelli, and Rupak Biswas. "From the quantum approximate optimization algorithm to a quantum alternating operator ansatz". *Algorithms* 12.2 (2019), p. 34.

[412] Zhihui Wang, Nicholas C Rubin, Jason M Dominy, and Eleanor G Rieffel. "XY-mixers: analytical and numerical results for QAOA". *arXiv preprint arXiv:1904.09314* (2019).

[413] James C Spall. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". *IEEE transactions on automatic control* 37.3 (1992), pp. 332–341.

[414] Susana Gomez and Jean Pierre Hennart. *Advances in Optimization and Numerical Analysis*. Vol. 275. Springer Science & Business Media, 2013.

[415] Yu Wang, Wotao Yin, and Jinshan Zeng. "Global convergence of ADMM in nonconvex nonsmooth optimization". *Journal of Scientific Computing* 78.1 (2019), pp. 29–63.

[416] Kate Smith-Miles, Davaatseren Baatar, Brendan Wreford, and Rhyd Lewis. "Towards objective measures of algorithm performance across instance space". *Computers & Operations Research* 45 (2014), pp. 12–24.

[417] David E Bernal, Sridhar Tayur, and Davide Venturelli. "Quantum Integer Programming (QuIP) 47-779: Lecture Notes". *arXiv preprint arXiv:2012.11382* (2020).

[418] Can Li, David E Bernal, Kevin C Furman, Marco A Duran, and Ignacio E Grossmann. "Sample average approximation for stochastic nonconvex mixed integer nonlinear programming via outer-approximation". *Optimization and Engineering* (2020), pp. 1–29.

[419] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, et al. "MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library". *Mathematical Programming Computation* (2021), pp. 1–48.

[420] Hans D Mittelmann. "Benchmarking optimization software-a (hi) story". In: *SN Operations Research Forum*. Vol. 1. 1. Springer. 2020, pp. 1–6.

[421] Francesco Ceccon, Radu Baltean-Lugojan, Michael Bynum, Chun Li, and Ruth Misener. "GALINI: An extensible mixed-integer quadratically-constrained optimization solver". *Optimization Online* (2021).

[422] Michael Bynum, Andrea Castillo, Carl Laird, Jean-Paul Watson, and USDOE. *Coramin v. 0.1 Beta, Version v. 0.1*. 2019. DOI: 10.11578/dc.20190311.1.

[423] Siegfried Schaible. "Fractional programming". In: *Handbook of global optimization*. Springer, 1995, pp. 495–608.

[424] Benoıt Chachuat. "MC++: A versatile library for McCormick relaxations and Taylor models". *URL http://www3. imperial. ac. uk/people/b. chachuat/research, http://www3. imperial. ac. uk/people/b. chachuat/research* (2011).

[425] David A Liñán, David E Bernal, Luis A Ricardez-Sandoval, and Jorge M Gómez. "Optimal design of superstructures for placing units and streams with multiple and ordered available locations. Part I: A new mathematical framework". *Computers & Chemical Engineering* 137 (2020), p. 106794.

[426] Kazuo Murota. "Discrete convex analysis". *Mathematical Programming* 83.1 (1998), pp. 313–371.

[427] Alejandro Perdomo-Ortiz, Alexander Feldman, Asier Ozaeta, Sergei V Isakov, Zheng Zhu, Bryan O'Gorman, Helmut G Katzgraber, Alexander Diedrich, Hartmut Neven, Johan de Kleer, et al. "Readiness of quantum optimization machines for industrial applications". *Physical Review Applied* 12.1 (2019), p. 014004.

[428] Monique Guignard and Siwhan Kim. "Lagrangean decomposition: A model yielding stronger Lagrangean bounds". *Mathematical programming* 39.2 (1987), pp. 215–228.

[429] John N Hooker and Greger Ottosson. "Logic-based Benders decomposition". *Mathematical Programming* 96.1 (2003), pp. 33–60.

[430] Peter L McMahon, Alireza Marandi, Yoshitaka Haribara, Ryan Hamerly, Carsten Langrock, Shuhei Tamate, Takahiro Inagaki, Hiroki Takesue, Shoko Utsunomiya, Kazuyuki Aihara, et al. "A fully programmable 100-spin coherent Ising machine with all-to-all connections". *Science* 354.6312 (2016), pp. 614–617.

[431] William M Kaminsky and Seth Lloyd. "Scalable architecture for adiabatic quantum computing of NP-hard problems". In: *Quantum computing and quantum bits in mesoscopic systems*. Springer, 2004, pp. 229–236.

[432] Vicky Choi. "Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design". *Quantum Information Processing* 10.3 (2011), pp. 343–353.

[433] Jun Cai, William G Macready, and Aidan Roy. "A practical heuristic for finding graph minors". *arXiv:1406.2741* (2014).

[434] Zhengbing Bian, Fabian Chudak, Robert Israel, Brad Lackey, William G Macready, and Aidan Roy. "Discrete optimization using quantum annealing on sparse Ising models". *Frontiers in Physics* 2 (2014), p. 56.

[435] Zongcheng Yang and Michael J Dinneen. *Graph minor embeddings for D-Wave computer architecture*. Tech. rep. Department of Computer Science, The University of Auckland, New Zealand, 2016.

[436] Thiago Serra, Teng Huang, Arvind Raghunathan, and David Bergman. "Template-based Minor Embedding for Adiabatic Quantum Optimization". *arXiv:1910.02179* (2019). arXiv: 1910.02179.

[437] Hiroshi Teramoto, Atsuyoshi Nakamura, Ichigaku Takigawa, Shin-Ichi Minato, Masanao Yamaoka, and Tamiki Komatsuzaki. "Graph Minors from Simulated Annealing for Annealing Machines with Sparse Connectivity". In: *Theory and Practice of*

Natural Computing: 7th International Conference, TPNC 2018, Dublin, Ireland, December 12–14, 2018, Proceedings. Vol. 11324. Springer. 2018, p. 111.

[438] Timothy D Goodrich, Blair D Sullivan, and Travis S Humble. "Optimizing adiabatic quantum program compilation using a graph-theoretic framework". *Quantum Information Processing* 17.5 (2018), p. 118.

[439] Shuntaro Okada, Masayuki Ohzeki, Masayoshi Terabe, and Shinichiro Taguchi. "Improving solutions by embedding larger subproblems in a D-Wave quantum annealer". *Scientific reports* 9.1 (2019), p. 2098.

[440] Kelly Boothby, Paul Bunyk, Jack Raymond, and Aidan Roy. *Next-generation topology of D-wave quantum processors*. Tech. rep. Technical report, 2019.

[441] Robert E Bixby. "A Brief History of Linear and Mixed-Integer Programming Computation". *Documenta Mathematica · Extra* (2012).

[442] Vahid Roshanaei, Kyle E. C. Booth, Dionne M Aleman, David R Urbach, and J Christopher Beck. "Branch-and-check methods for multi-level operating room planning and scheduling". *International Journal of Production Economics* (2019).

[443] Maplesoft. "Algorithms for Groebner Basis, Maple 2017" (2019).

[444] Jean-Charles Faugère. "A new efficient algorithm for computing Gröbner bases (F4)". *Journal of Pure and Applied Algebra* 139.13 (1999), pp. 61–88.

[445] Jean Charles Faugère. "A New Efficient Algorithm for Computing Gröbner Bases Without Reduction to Zero (F5)". In: *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*. ISSAC '02. Lille, France: ACM, 2002, pp. 75–83.

[446] Eiji Oki. "GLPK (GNU Linear Programming Kit)". In: *Linear Programming and Algorithms for Communication Networks*. 2012.

[447] John Forrest and Robin Lougee-Heimer. “CBC User Guide”. In: *Emerging Theory, Methods, and Applications*. 2005.

[448] Nike Dattani and Nick Chancellor. “Embedding quadratization gadgets on Chimera and Pegasus graphs”. *arXiv preprint arXiv:1901.07676* (2019).

[449] Eleanor G Rieffel et al. “From Ansätze to Z-gates: a NASA View of Quantum Computing”. *arXiv:1905.02860* (2019).

[450] Alejandro Perdomo-Ortiz, Neil Dickson, Marshall Drew-Brook, Geordie Rose, and Alan Aspuru-Guzik. “Finding low-energy conformations of lattice protein models by quantum annealing”. *Scientific Reports* (2012). arXiv: 1204.5485.

[451] Raouf Dridi, Hedayat Alghassi, and Sridhar Tayur. “Minimizing Polynominal Functions on Quantum Computers”. *Science and Culture, MAY-JUNE* (2019).

[452] David A. Cox, John B. Little, and Donal O’Shea. *Using algebraic geometry*. Graduate texts in mathematics. New York: Springer, 1998.