Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy

TITLE Improved Formulations and Computational Strategies for the Solution and Nonconvex Generalized Disjunctive Programs

PRESENTED BY Francisco Trespalacios

ACCEPTED BY THE DEPARTMENT OF

Chemical Engineering

IGNACIO GROSSMANN	9/30/15
IGNACIO GROSSMANN, ADVISOR	DATE
LORENZ BIEGLER	9/30/15
LORENZ BIEGLER, DEPARTMENT HEAD	DATE

APPROVED BY THE COLLEGE COUNCIL

VIJAYAKUMAR BHAGAVATULA	9/30/15
DEAN	DATE

Improved Formulations and Computational Strategies for the solution of Convex and Nonconvex Generalized Disjunctive Programs

by

Francisco Trespalacios

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

at

Carnegie Mellon University
Department of Chemical Engineering
Pittsburgh, Pennsylvania

September, 2015

Acknowledgments

First of all I would like to thank my thesis advisor Prof. Ignacio E. Grossmann. I am very grateful for his continuous encouragement and knowledge, which had allowed me to explore exciting new areas. While his academic excellence guided my research these past years, it is his personal qualities that had genuinely left a mark in me. His approachability, hard work, ethic standards, and passion for knowledge, set an example that has made deep impact in my professional and personal life. In addition to his extraordinary academic and personal qualities, what really makes him a great mentor, is that he truly cares.

I am grateful to my thesis committee, Prof. Egon Balas, Prof. Larry Biegler, Prof. John Hooker, Dr. Dimitri Papageorgiou, and Prof. Nick Sahinidis, for their wisdom and patience. It was an honor for me to have this opportunity.

I want to thank my industrial collaborators in ExxonMobil, in particular Dr. Myun-Seok Cheon and Dr. Dimitri Papageorgiou. I appreciate their helpful feedback and their patience to listen and discuss my doubts and concerns. I also want to thank GAMS Development Corp., in particular Alex Meeraus and Michael Bussieck. Many of the results in this thesis are possible because of their support. The financial support from the Center for Advanced Process Decision-making (CAPD) is gratefully acknowledged.

During my time at Carnegie Mellon University I was fortunate to collaborate with many students and researchers in different projects. I thank Dr. John Siirola and Dr. Mahdi Sharifzadeh for believing in our work and implementing some of our strategies in Pyomo. I thank Irene Lotero for allowing me to be part of the extraordinary research she did during her Master's degree at CMU. I thank Dr. Faram Engineer and Prof. Felipe Diaz Alvarado

for giving me the opportunity to work with them on a challenging optimization problem. I am also grateful to David Esteban Bernal Neira and Dr. Stefan Vigerske for their patience, skill, and hard work for implementing algorithmic developments in DICOPT.

I want to thank the many students and visiting researchers from PSE. The informal discussions and friendship created a relaxed but productive atmosphere at CMU. I especially thank Linlin Yang, Pablo Garcia-Herreros, Sumit Mitra, German Oliveros Patino, Satyajith Amaran, Miguel Zamarripa, Rodrigo Lopez-Negrete, Axel Nyberg and Markus Drouven for inspiration and friendship.

Finally, I want to thank all my family. In particular, my parents and my sister for their teachings and unconditional support.

Abstract

Many optimization problems require the modelling of discrete and continuous variables, giving rise to mixed-integer linear and mixed-integer nonlinear programming (MILP / MINLP). An alternative representation of MINLP is Generalized Disjunctive Programming (GDP)[1]. GDP models are represented through continuous and Boolean variables, and involve algebraic equations, disjunctions, and logic propositions. This higher level representation facilitates the modelling process while keeping the logic structure of the problem. GDP models are typically reformulated as MINLP problems to exploit the developments in these solvers. The two traditional GDP-to-MINLP reformulations are the Big-M (BM) and Hull-reformulation (HR). Alternatively to direct MINLP reformulations, special techniques can help to improve the performance in solving GDP problems.

There are two main contributions in this thesis. The first contribution involves the development of reformulations and methods that generate improved MINLP models form GDP problems. This development is achieved by exploiting the logic-nature of GDP, as well as alternative GDP-to-MINLP reformulations, to obtain relatively small MINLP models with tight continuous relaxations. The second contribution of this thesis is the improvement of existing GDP solution methods by the use of novel concepts. In particular, we improve the linear disjunctive branch and bound through the use of a Lagrangean relaxation of the HR. Also, we extend the logic-based outer-approximation to nonconvex problems, and develop a novel method to obtain cutting planes that improves the linear relaxation of the nonconvex problem.

In the thesis, we first present a new Big-M reformulation of GDPs. Unlike the traditional

Big-M reformulation that uses one M-parameter for each constraint, the new approach uses multiple M-parameters for each constraint. The multiple-parameter Big-M (MBM) reformulation is at least as tight as the traditional BM. Furthermore, it does not require additional variables or constraints. We present the new MBM and analyze the strength in its continuous relaxation compared to that of the traditional Big-M.

We then present two algorithmic approaches to improve mixed-integer models that are originally formulated as convex GDPs. The algorithms seek to obtain an improved continuous relaxation of the MINLP reformulation of the GDP, while limiting the growth in the problem size. Both algorithms make use of the logic operation called basic step. This operation allows the derivation of formulations with continuous relaxations that are stronger than the direct BM and HR reformulations. The two algorithms differ in the method to exploit the advantages of the small problem size of the BM, and the tight continuous relaxation of the HR after the application of basic steps. The first algorithm uses a hybrid reformulation of GDP that seeks to exploit both advantages of the BM and HR. The second algorithm uses the strong formulation to derive cuts for the BM, generating a stronger formulation with small growth in problem size.

In terms of GDP solution methods, we first present an enhancement to the disjunctive branch and bound for linear GDPs. In particular, we present a Lagrangean relaxation of the HR. The proposed Lagrangean relaxation can be applied to any linear GDP, and it always assigns 0-1 values to the binary variables of the HR. Furthermore, this relaxation is much simpler to solve than the continuous relaxation of the HR. The Lagrangean relaxation can be used in different manners to improve GDP solution methods. In this thesis, we explore the use of the Lagrangean relaxation as a primal heuristic to find feasible solutions in a disjunctive branch and bound. We note that the proposed Lagrangean relaxation, and its use in the disjunctive branch and bound, can be extended to nonlinear convex problems.

We then extend the logic-based outer-approximation to the global solution of non-convex GDPs. The general idea of the algorithm is to have a linear master GDP that overestimates the feasible region of the GDP. This master problem provides a valid lower bound (in a minimization problem), and the selection of only one disjunctive term in each of the disjunctions. With the alternative provided by the master problem, an NLP subprob-

lem is solved to global optimality. This NLP subproblem is smaller and simpler than the continuous relaxation of the MINLP reformulation of the original GDP. After solving the subproblem, infeasibility or optimality integer cuts can be added to the master problem. This basic algorithm has the advantage of solving only small NLP problems to global optimality, instead of solving a larger MINLP to global optimality from the beginning. Furthermore, by using GDP as framework the NLP subproblem is smaller and simpler than an equivalent method directly applied to the MINLP reformulation. In order to further improve the performance of this logic-based outer approximation, two main features were implemented: derivation of additional cuts and partition of the algorithm in two stages.

Finally, we apply a modified version of the global logic-based outer-approximation to the multiperiod blending problem. In addition to the proposed solution method, we present an improved problem formulation that makes use of redundant constraints. In order to generate such constraints, an alternative formulation was derived. The main idea in the new formulation is to track sources or commodities in the system, instead of tracking compositions. The main advantage is that it is possible to create redundant constraints in which the sum of individual source flows adds up to the total flow. Similarly, the sum of individual source inventories adds up to the total inventory. These redundant constraints considerably improve the relaxation of the model when linear approximations are used for the bilinear terms. Furthermore, the additional constraints can be included in the original model, strengthening its linear relaxation.

This thesis makes several important contributions. From an aggregated perspective, our most significant contribution is the use of GDP and its logic structure to obtain improved models and develop solution methods. In this thesis we show that GDP is not only an intuitive and structured modeling framework, but it also opens a set of tools that are not accessible when modeling problems using mixed-integer programming. The tools we have developed can help to solve some problems in Process Systems Engineering (PSE). Furthermore, we hope that the advantages of formulating some problems using GDP become apparent. As the PSE community continues to increasingly use GDP as modeling framework, we hope it brings greater attention to the OR community.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Many optimization problems require the modeling of discrete and continuous variables, giving rise to mixed-integer linear and mixed-integer nonlinear programming (MILP / MINLP). Models in which the objective function and the constraints are linear are MILP problems. MINLP problems involve a nonlinear objective function and/or nonlinear constraints. Although MINLP problems are nonconvex, MINLP can be divided in two categories, convex MINLP and nonconvex MINLP. Convex MINLP involves minimizing a convex objective function over a feasible region that is convex when the discrete variables are relaxed as continuous variables. In nonconvex MINLP the objective function and/or the continuous relaxation of the feasible region are not convex.

Many Process Systems Engineering (PSE) applications are modeled using MILP and MINLP. Furthermore, many developments in MINLP and global optimization have been motivated by applications in PSE[2]. Different types of models have been used for different PSE applications. Table 1.1, adapted from Biegler and Grossmann[3], provides an overview of the different types of models used in the different PSE applications. The models in Table 1.1 are: linear programs (LP), mixed-integer linear programs (MILP), quadratic programs

Table 1.1: Applications of mathematical programming in process systems engineering.

	LP	MILP	QP, LCP	NLP	MINLP
Process synthesis					
Process Flowsheet		✓		✓	✓
Reactor Networks	✓			✓	✓
Separations		✓			✓
Heat Exchange Networks	✓	✓		✓	✓
Water Networks	✓	✓		✓	✓
Operations					
Planning	✓	✓		✓	✓
Scheduling	✓	✓			✓
Real-time optimization	✓		✓	✓	
Process control					
Linear MPC	✓		✓		
Nonlinear MPC				✓	
Hybrid		✓		✓	✓
Molecular computing		✓			✓

(QP), linear complimentary problems (LCP), nonlinear programs (NLP), mixed-integer nonlinear programs (MINLP).

MILP theory has been considerably enriched in the past decades, which has been reflected in the advances of methods to solve this type of problems[4]. MINLP methods greatly benefit from advances in MILP and NLP methods, and have also improved considerably in recent years[5]. Despite these advances, the performance of MILP and MINLP solvers strongly depends on the problem formulation. Considering that optimization problems can be formulated in different ways, two main questions still remain in this area: a) How to create good MILP/MINLP models for a discrete-continuous optimization problem?, and b) How to use the logic structure of a problem to improve its solution method?

Motivated by these questions, modeling frameworks such as Generalized Disjunctive Programming have been developed. Generalized Disjunctive Programming (GDP)[1] is a higher-

level representation of MILP/MINLP models. GDP models are represented through continuous and Boolean variables, and involve algebraic equations, disjunctions, and logic propositions. GDP representation facilitates the modeling process while keeping the logic structure of the problem.

The general goal of this thesis is to develop new methods for improved MINLP formulations of GDP problems, as well as for improving GDP solution methods. In particular, for convex GDP problems we seek to obtain MINLP formulations that have strong continuous relaxation and small growth in problem size. We also focus on improving existing GDP solution methods. First, in convex GDP problems by enhancing the primal heuristics in the disjunctive branch and bound. And second, by extending the logic-based outer-approximation to the global solution nonconvex GDP problems.

1.2 Generalized disjunctive programming

Generalized disjunctive programming is a higher-level representation of MINLP problems [1]. The general GDP formulation can be represented as follows:

$$
\begin{aligned}
& & & min\ f(x) & & \\
& s.t. & & g(x) \leq 0 & & \\
& & & \bigvee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ r_{ki}(x) \leq 0 \end{bmatrix} & & k \in K \\
& & & \veebar_{i \in D_k} Y_{ki} & & k \in K \\
& & & \Omega(Y) = True & & \\
& & & x^{lo} \leq x \leq x^{up} & & \\
& & & x \in \mathbb{R}^n & & \\
& & & Y_{ki} \in \{True, False\} & & k \in K, i \in D_k
\end{aligned}
\tag{GDP}
$$

In (GDP), the objective is function of the continuous variables $x \in \mathbb{R}^n$. $g(x) <= 0$, where $g : \mathbb{R}^n \mapsto \mathbb{R}^m$, are the global constraints of the problem (i.e. these constraints must

be satisfied regardless of the discrete decisions). The formulation involves disjunctions $k \in K$, each of which contains disjunctive terms $i \in D_k$. The disjunctive terms in each disjunction are linked together by an "or" operator ($\vee$). A Boolean variable Y_{ki} and a set of constraints $r_{ki}(x) \leq 0$ are assigned to each disjunctive term. Exactly one disjunctive term in each disjunction must be enforced ($\veebar_{i \in D_k} Y_{ki}$). A Boolean variable takes a value of $True$ ($Y_{ki} = True$) when a disjunctive term is active, and the corresponding constraints ($r_{ki}(x) \leq 0$) are enforced. When a term is not active ($Y_{ki} = False$), its corresponding constraints are ignored. The logic constraints $\Omega(Y) = True$ represents the relations between the Boolean variables in propositional logic. Note that this is a general representation of any GDP. If there are equality constraints $g(x) = 0$, they can be represented by $g(x) \leq 0$ and $-g(x) \leq 0$.

1.2.1 MINLP reformulations of GDP

GDP problems are typically reformulated as MILP/MINLP by using either the Big-M (BM) or Hull Reformulation (HR)[6,7]. The (BM) reformulation generates a smaller MINLP, while the (HR) provides a tighter formulation[8].

The (BM) reformulation is as follows:

$$
\begin{aligned}
& & & min\ f(x) & & \\
& s.t. & & g(x) \leq 0 & & \\
& & & r_{ki}(x) \leq M^{ki}(1 - y_{ki}) & & k \in K, i \in D_k \\
& & & \sum_{i \in D_k} y_{ki} = 1 & & k \in K \\
& & & Hy \geq h & & \\
& & & x^{lo} \leq x \leq x^{up} & & \\
& & & x \in \mathbb{R}^n & & \\
& & & y_{ki} \in \{0, 1\} & & k \in K, i \in D_k
\end{aligned}
\tag{BM}
$$

In (BM) the Boolean variables Y_{ki} are transformed into binary variables y_{ki}: $Y_{ki} = True$

is equivalent to $y_{ki} = 1$ and $Y_{ki} = False$ is equivalent to $y_{ki} = 0$. Constraint $\sum_{i \in D_k} y_{ki} = 1$ enforces that exactly one disjunctive term is selected per disjunction. The transformation of logic constraints $\Omega(Y) = True$ to integer linear constraints ($Hy \geq h$) is easily obtained[9,10]. For an active term, the corresponding constraints $r_{ki}(x) \leq 0$ are enforced. For a term that is not active ($y_{ki} = 0$) and a large enough M^{ki}, the corresponding constraints $r_{ki}(x) \leq M^{ki}$ become redundant.

The (HR) formulation is given as follows:

$$
\begin{aligned}
& min\ f(x) && \\
s.t.\quad & g(x) \leq 0 && \\
& x = \sum_{i \in D_k} \nu^{ki} && k \in K \\
& y_{ki} r_{ki}(\nu^{ki}/y_{ki}) \leq 0 && k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ki} = 1 && k \in K \\
& Hy \geq h && \\
& x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} && k \in K, i \in D_k \\
& x \in \mathbb{R}^n && \\
& \nu^{ki} \in \mathbb{R}^n && k \in K, i \in D_k \\
& y_{ki} \in \{0, 1\} && k \in K, i \in D_k
\end{aligned}
\tag{HR}
$$

In (HR), similarly to (BM), the Boolean variables Y_{ki} are transformed into 0-1 variables y_{ki}, $\Omega(Y) = True$ is transformed into $Hy \geq h$, and $\sum_{i \in D_k} y_{ki} = 1$ enforces that only one disjunctive term is selected per disjunction. In (HR), the continuous variables x are disaggregated into variables ν^{ki}, for each disjunctive term $i \in D_k$ in each disjunction $k \in K$. The constraint $x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki}$ enforces that when a term is active ($y_{ki} = 1$), the corresponding disaggregated variables lie within their bounds. When it is not selected, they take a value of zero. The constraint $x = \sum_{i \in D_k} \nu^{ki}$ enforces that the original variables x have the same value as the disaggregated variables of the active terms. The functions in the constraints of a disjunctive term ($r_{ki}(x) \leq 0$) are represented by the perspective

function $y_{ki}r_{ki}(\nu^{ki}/y_{ki})$[11]. When a term is active ($y_{ki} = 1$) the constraint is enforced for the disaggregated variable ($r_{ki}(\nu^{ki}) \leq 0$). When it is not active ($y_{ki} = 0$), the constraint is trivially satisfied ($0 \leq 0$). When the constraints in the disjunction are linear ($A^{ki}x \leq a^{ki}$), the perspective function becomes $A^{ki}\nu^{ki} \leq a^{ki}y_{ki}$, which is a well-known representation in disjunctive programming[12]. To avoid singularities in the perspective function, the following approximation can be used[13]:

$$y_{ki}r_{ki}(\nu^{ki}/y_{ki}) \approx ((1-\epsilon)y_{ki}+\epsilon)r_{ki}\left(\frac{\nu^{ki}}{(1-\epsilon)y_{ki}+\epsilon}\right) - \epsilon r_{ki}(0)(1-y_{ki}) \qquad \text{(APP)}$$

where ϵ is a small finite number (e.g. 10^-5). This approximation yields an exact value at $y_{ki} = 0$ and $y_{ki} = 1$ irrespective of the value of ϵ, and is convex if r_{ki} is convex.

Note that (HR) involves more variables and constraints than (BM). However, (HR) provides a stronger formulation[8]. The (HR) reformulation represents the intersection of the convex hulls of each disjunction. This representation of convex hulls of disjunctions, using the perspective function, has been previously presented for convex MINLP[11,14].

Figure 1.1 illustrates, for both reformulations, the projection over $x1$ and $x2$ of the feasible region defined by two disjunctions. The first disjunction represents the selection of rectangle $A1$ or rectangle $A2$, and the second one the selection of circle $B1$ or circle $B2$. The dashed region defines the feasible region, and the shaded area represents the continuous relaxation of the (BM) and (HR). It is clear that the (HR) has a tighter relaxation than the (BM).

It is important to note that even though the (HR) is the intersection of the convex hulls of the individual disjunctions, this in general does not mean that it is the convex hull of the feasible region as can be seen in Figure 1.1. In order to further improve the tightness of the (HR) it is possible to make use of the logic operation called basic step.

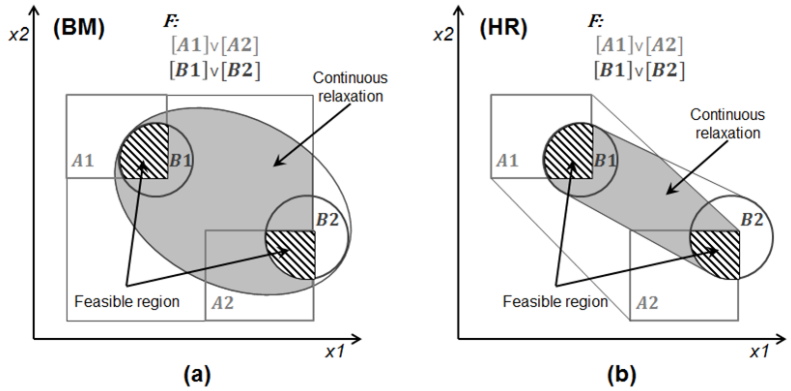

Figure 1.1: Illustration of (BM) and (HR) reformulations

1.2.2 Convex GDP and basic steps

In the particular case in which $f(x)$, $g(x)$, and $r_{ki}(x), i \in D_k, k \in K$ are convex, (GDP) becomes a convex GDP. Any convex GDP is equivalent to a disjunctive convex program[15]. Therefore, some of the rich theory behind disjunctive convex programming can be extended to GDP. Of particular interest for this thesis is the concept of a basic step.

Disjunctive convex programming can be defined as the optimization over a disjunctive convex set. A disjunctive set can be described as the union ($\cup$) and intersection ($\cap$) of a collection of inequalities. Consider the following definitions:

Convex inequality: $C = \{x \in \mathbb{R}^n | \Phi(x) \leq 0\}$, where $\Phi(x) : \mathbb{R}^n \to \mathbb{R}^1$ is a convex function.

Convex set: $P = \bigcap_{m \in M} C_m$

Elementary disjunctive set: $H = \bigcup_{m \in M} C_m$

Disjunction: $S_k = \bigcup_{i \in D_k} P_i = \bigcup_{i \in D_k} \bigcap_{m \in M_i} C_m$

A disjunction such that $S_k = P_i$ for some $i \in D_K$ is called improper disjunction, otherwise it is called a proper disjunction. Note that if D_k is a singleton then S_k is improper.

There are alternative forms to represent disjunctive convex sets. In particular:

Regular form: $F = \bigcap_{k \in K} S_k$

Disjunctive normal form (DNF): $F = S = \bigcup_{i \in D} P_i$

A basic step is the intersection between two disjunctions to form a new disjunction. Basic steps bring a disjunctive set in regular form closer to its DNF. The definition of basic step is as follows[12,15]:

Theorem 1.2.1 (Basic Step) *Let F be a disjunctive set in regular form. Then F can be brought to DNF by $|K| - 1$ recursive applications of the following basic step which preserves regularity. For some $k, l \in K$, bring $S_k \cap S_l$ to DNF by replacing it with:* $S_{kl} = S_k \cap S_l = \bigcup_{i \in D_k, j \in D_l} (P_i \cap P_j)$

It is possible to use the concept of a basic step to strengthen GDP formulations. In particular, there are two main consequences of the basic steps for GDP problems[13,15]: a) The continuous relaxation of the (HR) of a disjunctive set after a basic step is at least as tight as the one before the basic step; and b) The (HR) of the DNF describes the convex hull of the problem.

The results indicate that we can improve the strength of the continuous relaxation of the (HR) by applying basic steps. Furthermore, in the extreme case in which we intersect all of the disjunctions and all of the global constraints into a single disjunction, we obtain the convex hull of the problem. The drawback in the application of basic steps is the growth in the number of disjunctive terms, which is exponential when applying proper basic steps.

The application of proper basic steps not only increases the problem size, but also results in an exponential growth of disjunctive terms. As described in Section 1.2.1, each disjunctive term is associated with a binary variable in any GDP reformulation. Therefore, the growth of disjunctive terms implies an exponential increase in the number of binary variables. However, it is possible to avoid the exponential growth in binary variables by using the following theorem[12]:

Theorem 1.2.2 *Consider MILP/MINLP representation of two disjunctions $k, l \in K$, whose disjunctive terms are represented by the 0-1 variables $y_{ki}, y_{lj}, i \in D_k, j \in D_l$. If a basic step is applied between disjunction k and disjunction l, the variables representing the*

disjunctive terms of the resulting disjunction $\hat{y}_{ij} \in \{0,1\}$ *can be equivalently represented by:*

$$
\begin{aligned}
y_{ki} &= \sum_{j \in D_l} \hat{y}_{ij} && i \in D_k \\
y_{lj} &= \sum_{i \in D_k} \hat{y}_{ij} && j \in D_l \\
\sum_{i \in D_k, j \in D_l} \hat{y}_{ij} &= 1 \\
\sum_{i \in D_k} y_{ki} &= 1 \\
\sum_{j \in D_l} y_{lj} &= 1 \\
0 \leq \hat{y}_{ij} &\leq 1 \\
y_{ki}, y_{lj} \in \{0,1\} & \qquad i \in D_k, j \in D_l
\end{aligned}
$$

Proof. The proof follows from Theorem 4.4 of Balas[12] ■.

Theorem 1.2.2 relates the new terms after a basic step to those before the basic step. Only the variables associated to the original disjunctive terms are required to be binary, while the ones related to the new terms can be continuous between 0 and 1.

Figure 1.2 illustrates tightness of relaxation of the (HR) before and after the application of a basic step. The illustration shows a feasible region described by two disjunctions with two disjunctive terms each, that is $([A_1] \vee [A_2]) \wedge ([B_1] \vee [B_2])$. Figure 1.2.b shows that, after a basic step, the two disjunctions are intersected to form a new single disjunction $([A_1] \wedge [B_1]) \vee ([A_2] \wedge [B_2])$. Thus, the basic step not only improves the tightness of the relaxation, but it brings the problem into DNF. The (HR) of the DNF, as expressed earlier, describes the convex hull of the feasible region, as this can also be seen in 1.2.b. Finally, it is important to note that some of the resulting terms after the application of a basic step might become infeasible. In this example, since A1 and B2, and A2 and B1 do not intersect, the corresponding new terms are not feasible.

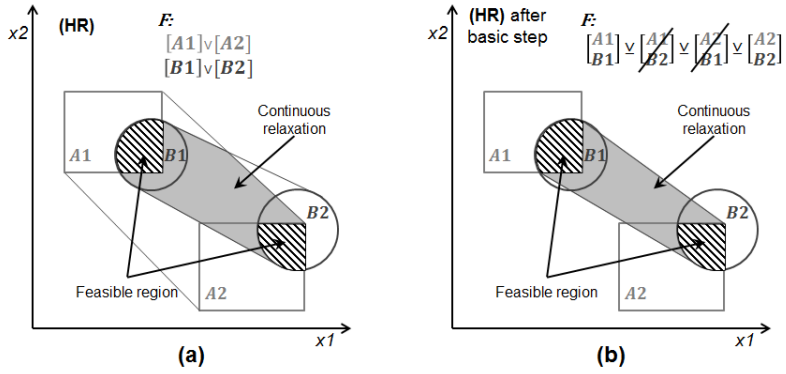

Figure 1.2: Illustration of (HR) (a) before, and (b) after the application of a basic step

1.2.3 Logic-based solution methods for GDP

GDP problems can be reformulated as MINLP, or solved with specialized algorithms[16]. In particular, the disjunctive branch and bound[7,17] and logic-based outer-approximation[18] have proven to be successful in some problems.

Disjunctive branch and bound

The idea behind the disjunctive branch and bound[17,7] is to branch directly on the disjunctions, while using the continuous relaxation of the BM or HR of the remaining disjunctions. Let (R-BM) and (R-HR) be the continuous relaxation of (BM) and (HR), respectively. The disjunctive branch and bound is as follows:

For a node N_p, let z^p denote the optimal value of (R-BM) or (R-HR) of the corresponding GDP_p, and (x^p, y^p) its solution. Let L be the set of nodes to be solved, and GDP_0 be the original GDP. Let z^{up} be an upper bound for the optimal value of the objective function z^*.

0. Initialize. Set $L = N_0$, $z^{up} = \infty$, $(x^*, y^*) = \emptyset$.

1. Terminate. If $L = \emptyset$, then (x^*, y^*) is optimal and algorithm terminates.

2. Node selection. Choose a node $N_p \in L$, and delete it from L. Go either to 3a or to 3b.

3a. Bound. Solve the (R-HR) of GDP_p corresponding to N_p. If it is infeasible, go to step 1. Else, let z^p be its objective function and (x^p, y^p) its solution.

3b. Bound. Solve the (R-BM) of GDP_p corresponding to N_p. If it is infeasible, go to step 1. Else, let z^p be its objective function and (x^p, y^p) its solution.

4. Prune. If $z^p \geq z^{up}$, go to step 1.

If $y^p \in \mathbb{Z}^q$ let $z^{up} = z^p$ and $(x^*, y^*) = (x^p, y^p)$. Delete all nodes $N_r \in L$ in which $z^r \geq z^{up}$, and go to step 1. Else, go to step 5.

5. Branch. Select a disjunction $k \in K$ such that $y_{ki} \notin \{0, 1\}$ for some $i \in D_k$. For every $i \in D_k$, construct the corresponding GDP (GDP_p^i) by setting the constraints corresponding to the disjunctive term i as global, and removing the Boolean variables and constraints corresponding to term $i^{'} \neq i; i^{'} \in D_k$. Add $|D_k|$ new nodes, corresponding to GDP_p^i, to L. Go to step 1.

It is easy to see that this algorithm terminates finitely, in the worst case evaluating every possible node. This algorithm can be trivially modified to consider a tolerance for termination $\epsilon > 0$. The HR disjunctive branch and bound makes use of Step 3a at every node, while the BM disjunctive branch and bound makes use of Step 3b at every node. It is also possible to have a hybrid disjunctive branch and bound in which some nodes are solved using the BM and others using the HR. It is important to note that, as the disjunctive branch and bound progresses, the GDP problems that correspond to each node become smaller. In particular, the constraints that correspond to the disjunctive terms that were not selected are removed from the problem formulation in the subsequent nodes.

Note that the worst case involves $\prod_{k \in D_k} |D_k|$ leaf nodes, which is fewer than the worst case number of leaf nodes in a binary branch and bound algorithm (bounded by $2^{\sum_{k \in K}(|D_k|-1)}$) except when $|D_k| = 2, \forall k \in K$ (in which case the maximum number of leave nodes for both algorithms is $2^{|K|}$). The worst case for number of evaluated nodes in the disjunctive branch and bound depends on the sequence in which the disjunctions were branched, but it is smaller than the binary branch and bound except when $|D_k| = 2, \forall k$ (in which case it

is the same: $2^{|K|+1} - 1$).

It has been shown that the disjunctive branch and bound has advantages over the binary branch and bound[17]. Also, the disjunctive branch and bound can be used for linear, convex or nonconvex GDP problems. In the case of linear GDP problems, an LP is solved at every node. In nonlinear GDP problems, and NLP is solved to global optimality at every node.

Logic-based outer-approximation for nonlinear convex GDP

The logic-based outer-approximation[18] iteratively solves a master problem and a subproblem. The master problem is a linear GDP relaxation of the original GDP that seeks to find a lower bound and an alternative for the vector of Boolean variables (Y). The master problem in the first iteration is obtained by outer-approximating the nonlinear functions at certain solutions (x^p). The subproblem is an NLP in which the Boolean variables are fixed (i.e. setting $Y_{ki} = True$ for the terms selected by the master problem). The subproblem provides an upper bound when a feasible solution is found. If the subproblem is infeasible, then an alternative NLP subproblem is solved (the feasibility subproblem). The solution of the subproblem (x^p) is used to perform further linearizations of the constraints, which are added to the master linear GDP. Note that for a given solution x^P, the linearizations are performed for the global constraints and for the constraints that correspond to the selected active terms ($Y_{ki} = True$).

In the outer-approximation method, the linearization is performed by generating a first-order Taylor series approximation of the constraints. For a vector of nonlinear constraint ($g(x) \leq 0$) and a given set of solutions ($x^p; p = 1, ..., P$), the linearization is: $g(x^p) + \nabla g(x^p)^T (x - x^p) \leq 0$. The main limitation of this linearization is that it provides a valid linear relaxation only for convex functions. If the function $g(x)$ is nonconvex, this linearization can cut off regions that are feasible for $g(x) \leq 0$. In such a case, the master linear GDP is no longer a linear relaxation of the problem.

The convex logic-based outer-approximation guarantees convergence in finite iterations because the master problem and subproblem are equivalent for the discrete solutions in

which the subproblem has been evaluated $Y^p; p = 1, ..., P$. This means that if the master problem selects an alternative Y^p that was already evaluated in the subproblem (i.e. the linearization of this alternative is already included in the master problem), then the optimal objective values of the master problem and subproblem are the same. Clearly, if all the alternatives $Y^p; p = 1, ..., P$ that are feasible for the master problem are evaluated, then the lower bound of the master problem and the upper bound of the subproblem are the same. The proof of convergence can be found in the original work by Turkay and Grossmann[18].

1.3 Outline of the thesis

The general goal of this thesis is to develop solution methods for GDP problems, as well as computational strategies for obtaining improved MINLP formulations of GDP problems. Figure 1.3 presents an overview for the outline the thesis. The figure shows that chapter 2 is a new direct MINLP reformulation for GDP problems. Chapters 3 and 4 seek to obtain improved MINLP reformulations for convex GDP problems, using an algorithmic approach. Chapter 5 is an improvement to the disjunctive branch and bound for linear GDP problems (that can be generalized to nonlinear convex GDP). Chapter 6 is an extension of the logic-based outer-approximation for the global optimization of nonconvex GDP problems. Chapter 7 is an application of the logic-based outer-approximation for the global optimization of the multiperiod blending problem.

Chapter 2 presents a new Big-M reformulation for Generalized Disjunctive Programs. Unlike the traditional BM reformulation that uses one M-parameter for each constraint, the new approach uses multiple M-parameters for each constraint. Each of these M-parameters is associated with each alternative in the disjunction to which the constraint belongs. In this way, the proposed MINLP reformulation is at least as tight as the traditional Big-M, and it does not require additional variables or constraints. We present in this chapter the new Big-M, and analyze the strength in its continuous relaxation compared to that of the traditional Big-M. The new formulation is tested by solving several instances with an NLP-based branch and bound method. The results show that in most cases the

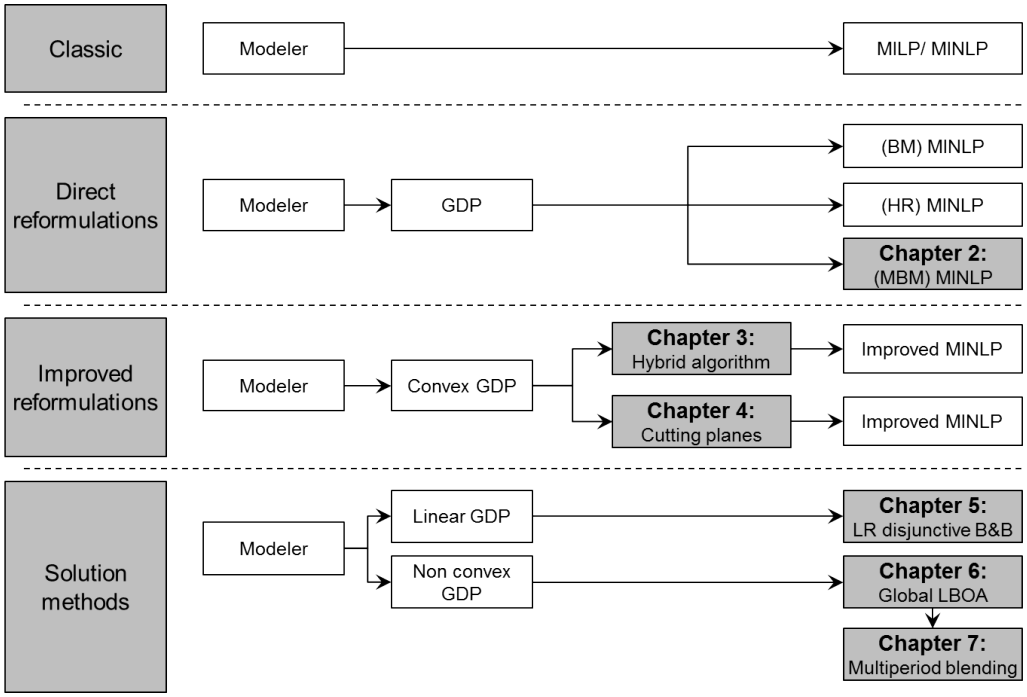

Figure 1.3: Outline of the thesis

new reformulation requires fewer nodes and less time to find the optimal solution.

In **chapter 3**, we propose an algorithmic approach to improve mixed-integer models that are originally formulated as convex GDP problems. The algorithm seeks to obtain an improved continuous relaxation of the MINLP reformulation of the GDP while limiting the growth in the problem size. There are three main stages that form the basis of the algorithm. The first one is a presolve, consequence of the logic nature of GDP, which allows us to reduce the problem size, find good relaxation bounds, and identify properties that help us determine where to apply a basic step. The second stage is the iterative application of basic steps, selecting where to apply them and monitoring the improvement of the formulation. Finally, we use a hybrid reformulation of GDP that seeks to exploit the advantages of the BM and the HR. The results show the improvement in the problem formulations by generating models with improved relaxed solutions and relatively small growth of continuous variables and constraints. The algorithm generally leads to reduction in the solution times.

Chapter 4 presents an alternative algorithm for improved MINLP reformulations through the use of cutting planes. The algorithm presented in this chapter uses the strengthened formulation of the HR after basic steps to derive cuts for the Big-M formulation. This method generates a stronger formulation than the BM with small growth in problem size. The results show that the algorithm improves GDP convex models, in the sense of providing formulations with stronger continuous relaxations than the (BM) with few additional constraints. In most cases, the algorithm also leads to a reduction in the solution time of the problems.

In **chapter 5**, we present a Lagrangean relaxation of the HR for linear GDP problems. The proposed Lagrangean relaxation has three important properties. The first property is that it can be applied to any linear GDP. The second property is that the solution to its continuous relaxation always yields 0-1 values for the binary variables of the HR for linear GDP problems. Finally, it is simpler to solve than the continuous relaxation of the HR. The proposed Lagrangean relaxation can be used in different GDP solution methods. In this chapter, we explore its use as primal heuristic to find feasible solutions in a disjunctive branch and bound algorithm. The modified disjunctive branch and bound is tested with

several instances. The results show that the modified disjunctive branch and bound performs better than other versions of the algorithm that do not include this primal heuristic. This work can be extended to nonlinear convex GDP problems, using the theory developed by Ruiz and Grossmann[15].

Chapter 6 presents a global logic-based outer-approximation method (GLBOA) for the solution of nonconvex GDP problems. The GLBOA allows the solution of nonconvex GDP models, and is particularly useful for optimizing the synthesis of process networks. Two enhancements to the basic GLBOA are presented. The first enhancement seeks to obtain feasible solutions faster by dividing the basic algorithm into two stages. The second enhancement seeks to tighten the lower bound of the algorithm by the use of cutting planes. The proposed method for obtaining cutting planes, the main contribution of this chapter, is a separation problem based in the convex hull of the feasible region of a subset of the constraints. Results show that the enhancements improve the performance of the algorithm, and that the algorithm is more effective at finding better feasible solutions than general purpose global solvers in the tested problems.

In **chapter 7** we present an application for the GLBOA, namely the multiperiod blending problem. The multiperiod blending problem involves bilinear terms, yielding a nonconvex GDP. In this chapter we present two major contributions for the global solution of the problem. The first one is an alternative formulation of the problem. This formulation makes use of redundant constraints that improve the linear relaxation of the GDP. The second contribution is a modified version of the GLBOA that decomposes the GDP model into two levels. The first level, or master problem, is a linear relaxation of the original GDP. The second level, or subproblem, is a smaller nonconvex GDP in which some of the Boolean variables of the original problem are fixed. The results show that the new formulation can be solved faster than alternative models, and that the decomposition method can solve the problems faster than state of the art general purpose solvers.

Finally, **chapter 8** summarizes the main findings of the thesis and lists its novel contributions. We also discuss additional future work directions that are worth investigating.

Chapter 2

Improved Big-M reformulation for GDP problems

2.1 Introduction

In this chapter, we present an alternative reformulation to GDP problems which is an improved version of the Big-M. The multiple-parameter Big-M (MBM) is as tight, and usually tighter, than the traditional (BM). This is achieved by assigning more than one big-M term in the constraints involved in each disjunction. The idea of using multiple big-M parameters was previously presented by Vielma[19] to formulate the union of polyhedra as MILPs. In this context, the multiple Big-M formulation is applied to a single disjunction containing linear constraints. The formulation we present can be regarded as an extension of the idea of Vielma (in the context of GDP), that is applied to formulations involving multiple disjunctions containing nonlinear constraints, Boolean variables, and logic relations among the Boolean variables.

This chapter is organized as follows. Section 2.2 provides an overview of the Big-M reformulation. The section also provides a method for obtaining the tightest (BM) reformulation of a GDP. Section 2.3 first presents the alternative reformulation (MBM), and a

method for obtaining the tightest possible (MBM) for a GDP. Second, it shows that this reformulation is at least as tight as the (BM). Finally, it provides a simple example to illustrate the (MBM) reformulation, comparing it to the traditional (BM). The new reformulation is tested with design of process networks and multi-product batch plant problems, and the results are presented in Section 2.4

2.2 Background: Big-M reformulation

The (BM) reformulation, presented in the introduction of the thesis, is as follows:

$$
\begin{aligned}
& min\ f(x) \\
s.t.\quad & g(x) \leq 0 \\
& r_{ki}(x) \leq M^{ki}(1 - y_{ki}) \quad k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ki} = 1 \qquad\qquad k \in K \\
& Hy \geq h \\
& x^{lo} \leq x \leq x^{up} \\
& x \in \mathbb{R}^n \\
& y_{ki} \in \{0,1\} \qquad\qquad k \in K, i \in D_k
\end{aligned}
\tag{BM}
$$

Note that a formulation with smaller M-parameters (M^{ki}) is at least as tight as a formulation with larger M-parameters (the right hand side in the constraints is smaller, which means that the feasible region of the continuous could be smaller). For this reason, the ideal M-parameter of a constraint is the smallest number that makes such a constraint redundant when required (i.e. when a disjunctive term that does not correspond to such constraint is selected). In many problems, the M-parameters can be obtained from knowledge of the meaning of the constraints. An alternative method is to obtain the M-parameters by solving optimization problems, although this approach may be impractical in many cases. Two methods can be used to obtain the M-parameters through the solution of optimization problems: one involves solving several GDP problems (2.1) and the other involves solving

several NLPs (2.2).

Consider E_{ki} to be the set of constraints corresponding to a disjunctive term ki. In order to find the M-parameter, it is possible to solve the following GDP to obtain the maximum value of the constraint $r_{kie}(x), k \in K, i \in D_k, e \in E_{ki}$ such that constraints $r_{ki'}(x) \leq 0, i' \neq i, i' \in D_k$ and all other disjunctions are satisfied:

$$
\begin{aligned}
& max\ r_{kie}(x) \\
s.t. \quad & r_{ki'}(x) \leq 0 \\
& g(x) \leq 0 \\
& \bigvee_{\hat{i} \in D_{\hat{k}}} \begin{bmatrix} Y_{\hat{k}\hat{i}} \\ r_{\hat{k}\hat{i}}(x) \leq 0 \end{bmatrix} \qquad \hat{k} \in K, \hat{k} \neq k \\
& \veebar_{\hat{i} \in D_{\hat{k}}} Y_{\hat{k}\hat{i}} \qquad \hat{k} \in K, \hat{k} \neq k \\
& \Omega(Y) = True \\
& x^{lo} \leq x \leq x^{up} \\
& x \in \mathbb{R}^n \\
& Y_{\hat{k}\hat{i}} \in \{True, False\} \quad \hat{k} \in K, \hat{k} \neq k, \hat{i} \in D_{\hat{k}}
\end{aligned}
\tag{2.1}
$$

Problem (2.1) seeks to maximize the value of a constraint ($r_{kie}(x)$), over the feasible region of the complete problem. It considers that a disjunctive term from the same disjunction different from i ($i' \in D_k, i' \neq i$) was selected (i.e. $Y_{ki'} = True$, so $r_{ki'}(x) \leq 0$ is enforced).

For a given $k \in K, i \in D_k, e \in E_{ki}, i' \neq i, i' \in D_k$, let x^* be the optimal solution of (2.1) and let $M^{kiei'} = r_{kie}(x^*)$. Then, the M-parameter of a constraint is $M^{kie} = \max_{\substack{i' \in D_k \\ i' \neq i}} \{M^{kiei'}\}$.

Note that solving (2.1) can be very challenging. Furthermore, it has to be solved several times for every single constraint inside a disjunction. For this reason, a more practical approach is to find an M-parameter that is optimal for the feasible region of its corresponding disjunction (instead of the feasible region of the complete problem). This can be achieved

by solving the following NLP problem:

$$\begin{aligned} & max\ r_{kie}(x) \\ s.t. \quad & r_{ki'}(x) \leq 0 \\ & x^{lo} \leq x \leq x^{up} \end{aligned} \tag{2.2}$$

For a given $k \in K, i \in D_k, e \in E_{ki}, i^{'} \neq i, i^{'} \in D_k$, let x^* be the optimal solution of (2.2) and let $M^{kiei^{'}} = r_{kie}(x^*)$. In a similar manner as with (2.1), a valid M-parameter M^{kie} for a constraint $(r_{kie}(x), k \in K, i \in D_k, e \in E_{ki})$ is $M^{kie} = \max_{\substack{i^{'} \in D_k \\ i^{'} \neq i}} \{M^{kiei^{'}}\}$. Note that the M-parameters obtained using (2.2) can be larger than the M-parameters obtained with (2.1). There are a few things to note about problem (2.2):

1. If the problem is linear, then the M-parameters are obtained through the solution of LPs.

2. Even if the original problem has convex constraints, if $r_{kie}(x)$ is nonlinear then (2.2) is non-convex. However, there is no need for a global optimal solution in order to obtain a valid M-parameter. Any upper bound to problem (2.2) is a valid M-parameter. Since problem (2.2) will normally be a small problem with few constraints, it is expected to quickly provide a good upper bound.

3. It is important to include the bounds of the variables to avoid an unbounded problem. However, (2.2) is not restricted to only include these constrains. It is possible to include additional constraints that correspond to the continuous relaxation of (2.1). The addition of these constraints to (2.2) will provide better M-parameters, but the NLP will become larger. Considering that in general the NLP is non-convex, finding a good trade-off between good M-parameter and the speed to obtain it is an important consideration.

4. Some conclusions might be drawn from (2.2). At least two of them provide valuable information. First, if for a given constraint $r_{kie}(x)$, $M^{kiei^{'}} \leq 0$ for all $i^{'} \neq i$, then $r_{kie}(x) \leq 0$ is satisfied by all of the disjunctive terms $i^{'} \neq i$ in that disjunctions. Therefore, it can be removed from the disjunctive term and be regarded as a global constraint. Note that it cannot simply be removed, otherwise when $Y_{ki} = True$ (i.e. disjunctive term i from disjunction k is selected) there is no guarantee that $r_{kie}(x) \leq 0$. The second conclusion is

that if (2.2) is infeasible, then $Y_{ki'} = True$ is infeasible. Therefore, the variable $Y_{ki'}$ and the set of constraints $r_{ki'}(x) \leq 0$ can be removed from the problem.

2.3 New Big-M reformulation

In disjunctions that contain more than two disjunctive terms, it is possible to have tighter (BM) formulations by assigning more than one big-M term in the constraints (i.e. one big-M term for each other disjunctive term in the disjunction). This formulation is as follows:

$$
\begin{aligned}
& min\ f(x) \\
s.t.\quad & g(x) \leq 0 \\
& r_{ki}(x) \leq \sum_{\substack{i' \in D_k \\ i' \neq i}} M^{kii'} y_{ki'} \quad k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ki} = 1 \qquad\qquad k \in K \qquad\qquad \text{(MBM)} \\
& Hy \geq h \\
& x^{lo} \leq x \leq x^{up} \\
& x \in \mathbb{R}^n \\
& y_{ki} \in \{0,1\} \qquad\qquad k \in K, i \in D_k
\end{aligned}
$$

The idea behind (MBM) is similar to that of (BM). When a disjunctive term is selected ($y_{ki} = 1,\ i \in D_k$), the other terms in the corresponding disjunction are not ($y_{ki'} = 0,\ \forall i' \in D_k, i' \neq i$). Therefore, $\sum_{\substack{i' \in D_k \\ i' \neq i}} M^{kii'} y_{ki'} = 0$, and the corresponding constraints $r_{ki}(x) \leq 0$ are enforced. If it is not selected ($y_{ki} = 0,\ k \in D_k$), then another term must be selected ($y_{ki'} = 1,\ i' \in D_k, i' \neq i$). If $M^{kii'}$ is large enough, then $r_{ki}(x) \leq M^{kii'}$ becomes redundant.

The value of $M^{kii'}$ can be directly obtained as with (BM). $M^{kii'}$ is then the vector of M-parameters directly obtained from (2.1) or (2.2), with the aforementioned advantages of using either (2.1) or (2.2): $M^{kii'} = [..., M^{kiei'}, ...]^T$, where $e \in E_{ki}$ are the constraints

corresponding to a disjunctive term ki.

2.3.1 Tightness of continuous relaxation of (MBM)

In this section we show that (MBM) is at least as tight as (BM), and can be tighter.

Theorem 2.3.1 *Let $M^{kii'}$ be valid M-parameters for (MBM), and let $M^{ki} = \max_{\substack{i' \in D_k \\ i' \neq i}} \{M^{kii'}\}$ be the M-parameters of (BM). Using such parameters in the reformulation, let (F-BM) and (F-MBM) be the feasible region of the continuous relaxations of (BM) and (MBM), respectively. Then (F-MBM) $\subseteq$ (F-BM).*

Proof. The only difference between (BM) and (MBM) are the two following constraints:

$$r_{ki}(x) \leq M^{ki}(1 - y_{ki}) \quad k \in K, i \in D_k \tag{2.3}$$

$$r_{ki}(x) \leq \sum_{\substack{i' \in D_k \\ i' \neq i}} M^{kii'} y_{ki'} \quad k \in K, i \in D_k \tag{2.4}$$

In (2.3), $(1 - y_{ki})$ can be substituted by $\sum_{\substack{i' \in D_k \\ i' \neq i}} y_{ki'}$, since $\sum_{i \in D_k} y_{ki} = 1$. (2.3) then becomes the following constraint:

$$r_{ki}(x) \leq M^{ki} \sum_{\substack{i' \in D_k \\ i' \neq i}} y_{ki'} \quad k \in K, i \in D_k \tag{2.5}$$

Introducing M^{ki} into the summation, we obtain:

$$r_{ki}(x) \leq \sum_{\substack{i' \in D_k \\ i' \neq i}} M^{ki} y_{ki'} \quad k \in K, i \in D_k \tag{2.6}$$

By definition $M^{ki} = \max_{\substack{i' \in D_k \\ i' \neq i}} \{M^{kii'}\}$. Therefore, the right hand side of (2.4) dominates the right hand side of (2.6).

Since the RHS of (2.4) is smaller than the RHS of (2.6), then any x that satisfies (2.4) also satisfies (2.6). Therefore, (2.4) $\subseteq$ (2.6). Since the only difference between (BM) and (MBM) are (2.3) and (2.4), and since (2.6) represents the same feasible region than (2.3) in (BM), then (F-MBM) $\subseteq$ (F-BM). ■.

Remark 2.3.2 *(F-MBM) and (F-BM) represent the same feasible region when, for all* $k \in K, i \in D_k$*:*

$$M^{kii'} = M^{kij} \; \forall j \in D_k, j \neq i, j \neq i, j \neq i'$$

Note that when a disjunction has only two disjunctive terms, there is only one M-parameter, and (2.3) and (2.4) are equivalent for that disjunction. If all disjunctions have only two terms, then (F-MBM) and (F-BM) represent the same feasible region.

Note that Theorem 2.3.1 holds true for any valid $M^{kii'}$. This includes the cases in which $M^{kii'}$ is obtained through (2.1) or (2.2).

2.3.2 Illustration of (MBM) reformulation

The improved Big-M reformulation is illustrated with the following example:

$$
\begin{aligned}
& min \ -2x_1 + x_2 \\
s.t. \quad & \begin{bmatrix} Y_1 \\ (x_1)^2 + (x_2)^2 \leq 1 \end{bmatrix} \vee \begin{bmatrix} Y_2 \\ (x_1 - 1)^2 + (x_2 - 5)^2 \leq 2 \end{bmatrix} \\
& \vee \begin{bmatrix} Y_3 \\ (x_1 - 4)^2 + (x_2 - 3)^2 \leq 4 \end{bmatrix} \\
& Y_1 \veebar Y_2 \veebar Y_3 \\
& -1 \leq x_1 \leq 6 \\
& -1 \leq x_2 \leq 7 \\
& Y_1, Y_2, Y_3 \in \{True, False\}
\end{aligned}
\tag{2.7}
$$

The traditional Big-M reformulation of (2.7), using the M-parameters directly obtained from (2.2) and selecting the largest corresponding values, is the following:

$$
\begin{aligned}
& min \ -2x_1 + x_2 \\
s.t. \quad & (x_1)^2 + (x_2)^2 \leq 1 + 48(1 - y_1) \\
& (x_1 - 1)^2 + (x_2 - 5)^2 \leq 2 + 35.1981(1 - y_2) \\
& (x_1 - 4)^2 + (x_2 - 3)^2 \leq 4 + 32(1 - y_3) \\
& y_1 + y_2 + y_3 = 1 \\
& -1 \leq x_1 \leq 6 \\
& -1 \leq x_2 \leq 7 \\
& y_1, y_2, y_3 \in \{0, 1\}
\end{aligned}
\tag{2.8}
$$

The (MBM) reformulation, using the M-parameters obtained from (2.2), is the following:

$$\begin{aligned} & min \ -2x_1 + x_2 \\ s.t. \quad & (x_1)^2 + (x_2)^2 \leq 1 + 41.42221y_2 + 48y_3 \\ & (x_1 - 1)^2 + (x_2 - 5)^2 \leq 2 + 35.1981y_1 + 29.4223y_3 \\ & (x_1 - 4)^2 + (x_2 - 3)^2 \leq 4 + 32y_1 + 21.1981y_2 \\ & y_1 + y_2 + y_3 = 1 \\ & -1 \leq x_1 \leq 6 \\ & -1 \leq x_2 \leq 7 \\ & y_1, y_2, y_3 \in \{0, 1\} \end{aligned} \tag{2.9}$$

In the traditional Big-M reformulation (2.8), the M-parameter of the first constraint is $M_1 = max\{41.42221, 48\} = 48$. The RHS of the first constraint in (2.8) is $1+48(1-y_1)$. Therefore, the first constraint in (2.9) is tighter than its corresponding constraint in (2.8). This can be easily seen if $(1 - y_1)$ is substituted by $y_2 + y_3$ in (2.9). This would yield $1 + 48y_2 + 48y_3$ as RHS of the first constraint of (2.9). Clearly, the RHS of the constraint in (2.9) dominates the RHS of the one in (2.9). The same holds true for the other two Big-M constraints, which indicates that (2.9) is at least as tight than (2.8) as proved in Theorem 2.3.1. The tightness of the formulation can also be reflected in the optimal objective value of the continuous relaxation. In this example, the optimal objective value of this problem is $\mathbf{-9.472}$. The optimal objective value of the continuous relaxation of (2.8) is $\mathbf{-10.493}$, and the optimal objective value of the continuous relaxation of (2.9) is $\mathbf{-9.735}$. Clearly, the optimal objective value of the continuous relaxation of (2.9) is better.

2.4 Examples and results

The new reformulation was tested with several instances of two problems: the process network problem and the design of multi-product batch plant problem. Both of these problems can be found in the Appendix B. For the case of the multi-product batch plant problem, it is possible to obtain the M-parameters from the physical meaning of the prob-

lem. For this reason, we present the formulation and M-parameters of this problem in this section.

2.4.1 Design of multi-product batch plant problem formulation

The problem, presented in detail in Appendix B, is as follows:

$$
\begin{aligned}
& min\ \alpha_1 \sum_j \exp(n_j + m_j + \beta_1 v_j) + \alpha_2 \sum_{Tj} \exp(\beta_2 v_{Tj}) \\
s.t.\quad & v_j \geq \ln(S_{ij}) + b_{ij} - n_j && \forall i, j \\
& e_i \geq \ln(T_{ij}) - b_{ij} - m_j && \forall i, j \\
& H \geq \sum_i (Q_i exp(e_i)) \\
& \begin{bmatrix} YS_j \\ v_{Tj} \geq \ln(S_j^*) + b_{ij+1} \quad \forall i \\ v_{Tj} \geq \ln(S_j^*) + b_{ij} \quad \forall i \\ b_{ij} - b_{ij+1} \leq \ln(S_{ij}^*) \quad \forall i \\ b_{ij} - b_{ij+1} \geq -\ln(S_{ij}^*) \quad \forall i \end{bmatrix} \vee \begin{bmatrix} \neg YS_j \\ v_{Tj} = 0 \\ b_{ij} - b_{ij+1} = 0 \quad \forall i \end{bmatrix} && \forall j < |J| \\
& \begin{bmatrix} YM_{j,1} \\ m_j = \ln(1) \end{bmatrix} \vee \ldots \vee \begin{bmatrix} YM_{j,maxp} \\ m_j = \ln(maxp) \end{bmatrix} && \forall j \\
& \begin{bmatrix} YN_{j,1} \\ n_j = \ln(1) \end{bmatrix} \vee \ldots \vee \begin{bmatrix} YN_{j,maxp} \\ n_j = \ln(maxp) \end{bmatrix} && \forall j \\
& YM_{j,1} \veebar \ldots \veebar YM_{j,maxp} && \forall j \\
& YN_{j,1} \veebar \ldots \veebar YN_{j,maxp} && \forall j \\
& YS_j, YM_{j,p}, YN_{j,p} \in \{True, False\} \quad \forall j, p = 1, \ldots, maxp
\end{aligned}
\tag{2.10}
$$

M-parameters in the multiproduct batch reactor problem

Problem (2.10) contains three sets of disjunctions. The first set has only two terms, so the M-parameters are the same for the (BM) and the (MBM). The other two sets of disjunc-

tions have to be written as inequalities to perform the reformulation:

$$\begin{bmatrix} YM_{j,1} \\ m_j \leq \ln(1) \\ m_j \geq \ln(1) \end{bmatrix} \vee ... \vee \begin{bmatrix} YM_{j,maxp} \\ m_j \leq \ln(maxp) \\ m_j \geq \ln(maxp) \end{bmatrix} \quad \forall j \tag{2.11}$$

$$\begin{bmatrix} YN_{j,1} \\ n_j \leq \ln(1) \\ n_j \geq \ln(1) \end{bmatrix} \vee ... \vee \begin{bmatrix} YN_{j,maxp} \\ n_j \leq \ln(maxp) \\ n_j \geq \ln(maxp) \end{bmatrix} \quad \forall j \tag{2.12}$$

In disjunction (2.11) and (2.12) the M-parameters of the (BM) and (MBM) are different.

For $p = 1, ..., maxp; \quad q = 1, ..., maxp; \quad q \neq p$, the M-parameters of (MBM) are:

$M^{jpq1} = \ln(q) - \ln(p)$

$M^{jpq2} = -\ln(q) + \ln(p)$

The M-parameters of the (BM) are:

$$M^{jp1} = \max_{\substack{q=1,...,maxp \\ q \neq p}} \{\ln(q) - \ln(p)\} = \ln(maxp) - \ln(p)$$

$$M^{jp1} = \max_{\substack{q=1,...,maxp \\ q \neq p}} \{-\ln(q) + \ln(p)\} = -\ln(1) + \ln(p)$$

The (BM) of constraints (2.11) are:

$$\begin{aligned} m_j &\leq \ln(p) + (\ln(maxp) - \ln(p))(1 - ym_{j,p}) \quad p = 1, ..., maxp \\ m_j &\geq \ln(p) - (\ln(p) - \ln(1))(1 - ym_{j,p}) \quad p = 1, ..., maxp \end{aligned} \tag{2.13}$$

The (MBM) of constraints (2.11) are:

$$\begin{aligned} m_j &\leq \ln(p) + \sum_{\substack{q=1,...,maxp \\ q \neq p}} (\ln(q) - \ln(p))ym_{j,q} = 1 \quad p = 1, ..., maxp \\ m_j &\geq \ln(p) - \sum_{\substack{q=1,...,maxp \\ q \neq p}} (-\ln(q) + \ln(p))ym_{j,q} = 1 \quad p = 1, ..., maxp \end{aligned} \tag{2.14}$$

Table 2.1: Solution of multi-product batch plant instances.

Instance	Solutions	Continuous relaxation		Time (s)		Nodes	
		(BM)	(MBM)	(BM)	(MBM)	(BM)	(MBM)
Proc-1-21	17.2	1.7	**10.0**	8.4	**4.0**	906	**404**
Proc-1-31	12.2	5.9	**7.9**	6.5	**1.2**	639	**86**
Proc-1-36	12.1	5.6	**7.8**	14.2	**1.5**	1,408	**112**
Proc-1-48	12.1	5.5	**7.4**	63.7	**5.7**	6,102	**405**
BatchS101006	769,440	734,943	734,943	237	**40**	10,894	**1,595**
BatchS121208	1,241,125	1,202,365	1,202,365	657	**365**	23,890	**10,587**
BatchS141208	1,487,664	1,440,995	1,440,995	1,148	**1,018**	38,643	**29,083**
BatchS151208	1,543,472	1,499,913	1,499,913	**121**	1,872	**3,729**	47,958
BatchS181210	2,042,327	2,006,860	2,006,860	145	145	3,719	**3,088**
BatchS201210	2,295,349	2,255,304	2,255,304	525	**502**	13,774	**10,158**

2.4.2 Results

The new reformulation was tested with 10 instances. 4 of these benchmark examples for the multi-product batch plant problem[20]. Two additional multi-product batch problems were generated using data of the benchmark problems. The instances were solved with SBB (NLP-based branch and bound) from GAMS 24.3.3[21], using an Intel(R) Core(TM) i7 CPU 2.93 GHz and 4 GB of RAM. SBB provides the optimal solution to MINLP problems with convex continuous relaxations (such as (B.1) and (2.10)). In NLP-based branch and bound methods, such as SBB, tighter relaxations will typically require the evaluation of fewer nodes to find and prove optimality. The M-parameters for the multi-product batch problems are presented in 2.4.1. The M-parameters for the process network problems were obtained by solving (2.2).

Table 2.1 shows the relaxation, number of nodes and time required to solve each of the instances. It is clear that for the process problems (MBM) provides a much better continuous relaxation. The continuous relaxation of both the (BM) and (MBM) is the same for the batch problems (although (MBM) is at least as tight than (BM) as can bee seen from the constraints described in Section 2.4.1). Fewer number of nodes are required for the (MBM) in 9 of the 10 instances, which generally yields reductions in the solution times. There is only one instance in which the (BM) performs better than the (MBM).

2.5 Conclusions

In this chapter we have presented an enhanced Big-M reformulation for GDP problems. The new reformulation uses multiple M-parameters in each constraint, instead of the single M-parameter the traditional big-M uses. In many cases, the new reformulation allows the multiple M-parameters to be smaller than the single M-parameter of the traditional Big-M. This translates into smaller RHS in the constraints, which can lead to tighter formulations. We have proved that the proposed reformulation is at least as tight as the traditional Big-M, and it does not require any additional variables or constraints. We reviewed two general methods for obtaining M-parameters in the traditional Big-M, and adapted it for the new proposed reformulation. The reformulation was tested with 10 instances, 4 of which are the design of a process network and 6 are multiproduct batch reactor problems. Compared to the traditional Big-M, the new reformulation requires fewer nodes to find the optimal solution in 9 out of 10 instances. The solution time of the new reformulation is smaller in 8 out of 10 cases when compared to the traditional Big-M reformulation.

Chapter 3

Algorithmic approach for improved mixed-integer reformulations of convex GDP problems

3.1 Introduction

In this chapter, we make use of the logic structure of a GDP. Instead of directly reformulating it as an MILP/MINLP, we apply a pre-analysis, a logic operation called basic step, and a hybrid reformulation. This logic manipulation allows us to obtain an improved formulation in comparison to the one obtained by a traditional reformulation. The resulting model can be solved by a GDP or an MILP/MINLP algorithm. Figure 3.1 outlines the main idea of this chapter. We should note that the proposed pre-analysis has some similarities to the MILP "fixing variables" pre-solve technique[22,23,24,25], but applied in the GDP space for convex nonlinear GDP models.

This chapter is organized as follows. Section 3.2 first presents the proposed algorithm, and second describes in detail each step of the method. Section 3.3 provides an example of the application of the algorithm. Section 3.4 presents the statistics, results and performance of

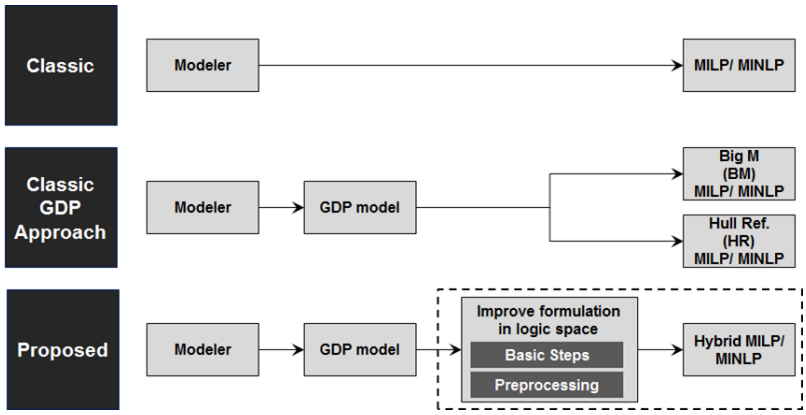

Figure 3.1: Different modeling approaches.

different test examples.

3.2 Algorithm to improve GDP formulations

In order to improve GDP formulations, we iteratively apply basic steps. In this section we first describe the algorithm, and afterwards we explain in detail each of its steps. Figure 3.2 provides an outline of the algorithm, where the main idea is to first perform a preanalysis for pre-solving, then repeatedly apply basic steps over one single disjunction, and finally use the (HR) in that disjunction and (BM) in all the remaining ones.

3.2.1 Algorithm

For the description of the algorithm, we will define the global constraints as individual inequalities, such that $g(x) \leq 0$ is represented with $g_e(x) \leq 0$, $e \in E$.

Step 1. Initialize z^*, GDP^* and z^{lo} from pre-solve. Set $iter = 1$.

Goal. Use pre-solve to initialize the algorithm; improving (GDP), finding better bounds, and providing a value that characterizes each disjunction.

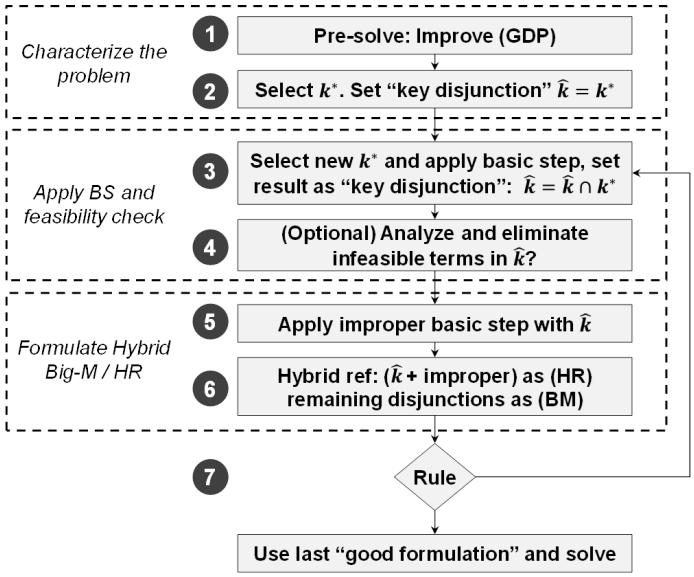

Figure 3.2: Outline of algorithm

Step 2. Select disjunction $k^* \in K$. Set $\hat{k}_1 = k^*$; $\hat{K} = \{\hat{k}_1\}$; $\hat{D}_{k_1} = D_{k^*}$.

Goal. Select the first disjunction to which basic steps will be applied, and set this disjunction as the "key disjunction".

Step 3. Set $iter = iter + 1$. Select $k^* \in K \backslash \hat{K}$. Set $\hat{k}_{iter} = k^*$; $\hat{K} = \{\hat{k}_1, ..., \hat{k}_{iter}\}$; $\hat{D}_{k_{iter}} = D_{k^*}$; $\hat{D} = \{\hat{D}_{k_1}, ..., \hat{D}_{k_{iter}}\}$.

Goal. Select the next disjunction, and apply a basic step between this disjunction and the "key disjunction". Set the resulting disjunction of this basic step as "key disjunction".

Step 4 *(Optional).* For all $\hat{i} = (\hat{i}_1, ..., \hat{i}_{iter}) \in \hat{D}$, such that: $(GDP^*) \bigcap_{\substack{\hat{k}_s \in \hat{K} \\ s=1,...,iter}} (y_{\hat{k}_s \hat{i}_s} = 1)$ becomes infeasible, set $\hat{i} \in INEAS_{iter}$.

Goal. Identify which terms in the "key disjunction" are infeasible.

Step 5. Select global equations to which apply a basic step $\hat{E} \in E$.

Step 6. Solve the continuous relaxation of (GDPH).

$$
\begin{aligned}
& & & min\ z = f(x) & & \\
& s.t. & & g_e(x) \leq 0 & & e \in E \backslash \hat{E} \\
& & & r_{ki}(x) \leq M^{ki}(1 - y_{ki}) & & k \in K \backslash \hat{K},\ i \in D_k \\
& & & x = \sum_{\hat{i} \in \hat{D}} \nu^{\hat{i}} & & \\
& & & \hat{y}_{\hat{i}} g_e(\nu^{\hat{i}} / \hat{y}_{\hat{i}}) \leq 0 & & e \in \hat{E},\ \hat{i} \in \hat{D} \\
& & & \hat{y}_{\tilde{i}_s} r_{\hat{k}_s i}(\nu^{\tilde{i}_s} / \hat{y}_{\tilde{i}_s}) \leq 0 & & s = 1, ..., iter,\ i \in \hat{D}_{k_s},\ \tilde{i}_s \in \tilde{D}_{si} \\
& & & y_{\hat{k}_s i} = \sum_{\substack{\hat{i} \in \hat{D} \\ \hat{i}_s = i}} \hat{y}_{\hat{i}} & & s = 1, ..., iter,\ i \in \hat{D}_{k_s} \\
& & & x^{lo} \hat{y}_{\hat{i}} \leq \nu^{\hat{i}} \leq x^{up} \hat{y}_{\hat{i}} & & \hat{i} \in \hat{D} \\
& & & \sum_{i \in D_k} y_{ki} = 1 & & k \in K \\
& & & \sum_{\hat{i}} \hat{y}_{\hat{i}} = 1 & & \\
& & & Hy \geq h & & \\
& & & \hat{y}_{\hat{i}} = 0 & & \hat{i} \in INEAS_{iter} \\
& & & z^{lo} \leq z & & \\
& & & x^{lo} \leq x \leq x^{up} & & \\
& & & x \in \mathbb{R}^n & & \\
& & & y_{ki} \in \{0, 1\} & & k \in K, i \in D_k \\
& & & 0 \leq \hat{y}_{\hat{i}} \leq 1 & & \hat{i} \notin INEAS_{iter}
\end{aligned}
\tag{GDPH}
$$

Step 7. If relaxed $(GDPH) > z^*$, set $z^* = relaxation(GDPH)$, $GDP^* = GDPH$. If the relaxation has not improved after a specified maximum number of iterations, or the GDPH problem size is greater than a specified limit, solve GDP^*. Else, go back to step 3.

(GDPH) is a hybrid reformulation in which the objective function $f(x)$ is the same as in the original formulation. The global constraints that were not selected for the application of a basic step ($e \in E \backslash \hat{E}$) remain unchanged. The disjunctions that were not selected to apply basic steps are reformulated using (BM) ($r_{ki}(x) \leq M^{ki}(1 - y_{ki})$). The disjunctions

that were intersected with basic steps now form a single disjunction, which we will denote "key disjunction", and that contains all terms $\hat{i} \in \hat{D}$. The corresponding variable to this new terms is $\hat{y}_{\hat{i}}$. Note that $|\hat{D}| = |\hat{D}_{k_1}| * ... * |\hat{D}_{k_{iter}}|$, which indicates an exponential growth of the disjunction with the number of iterations. The "key disjunction" is reformulated using (HR). The equation $x = \sum_{\hat{i} \in \hat{D}} \nu^{\hat{i}}$ relates the continuous variables x, to the disaggregated variables in all terms $\nu^{\hat{i}}$. The constraint $\hat{y}_{\hat{i}} g_e(\nu^{\hat{i}}/\hat{y}_{\hat{i}})$ is the (HR) reformulation of the global constraints that were intersected with the "key disjunction" $e \in \hat{E}$. Note that these constraints are present in all terms of the "key disjunction" ($\hat{i} \in \hat{D}$). Equation $\hat{y}_{\tilde{i}_s} r_{\hat{k}_s i}(\nu^{\tilde{i}_s}/\hat{y}_{\tilde{i}_s})$ is the (HR) reformulation of all the constraints in the terms of the disjunctions to which basic steps where applied. Each term of the "key disjunction" contains $iter$ sets of constraints, each one related to one of the disjunctions $\hat{k}_s \in \hat{K}$. The original set of constraints in a certain term i of a selected disjunction $\hat{k}_s$ will be present in all terms $\hat{i}$ of the "key disjunction", as long as its corresponding element $\hat{i}_s$ is equal to i ($\tilde{i}_s \in \tilde{D}_{si}$, where $\tilde{D}_{si} = \{\hat{D}_{k_1}, ..., i, ..., \hat{D}_{k_{iter}}\}$ and i is the s^{th} element of $\tilde{D}_{si}$). $\tilde{D}_{si}$ is a map that assigns the constraints in the original disjunctive terms $r_{\hat{k}_s i}$ to the terms $\hat{i}$ in the "key disjunction". Equation $y_{\hat{k}_s i} = \sum_{\substack{\hat{i} \in \hat{D} \\ \hat{i}_s = i}} \hat{y}_{\hat{i}}$ relates the original binary variables y_{ki} to the new variables $\hat{y}_{\hat{i}}$, and it allows the new variables $\hat{y}_{\hat{i}}$ to be continuous while enforcing them to always take a $\{0, 1\}$ value[12,15].

It should be noted that the algorithm applies basic steps, which are valid logic operations for GDP. Also, the hybrid reformulation is a valid MILP/MINLP representation of the problem, since the (BM) and (HR) are valid reformulations for individual disjunctions. Therefore, the algorithm provides a valid MILP/MINLP representation of the original GDP.

Section 3.3 provides an illustration of each of the steps in the algorithm. The remaining of this section describes each of the steps of the algorithm with detail.

3.2.2 Step 1: Pre-solve

The pre-solve has three purposes: eliminate infeasible terms, find better bounds, and provide a value that will characterize each disjunction. This pre-solve can be performed due to the logic nature of GDP formulations. The pre-solve can be regarded as a strong branching over every disjunction, and only in the root node.

The pre-solve procedure is as follows:

0. Set $k = 1$ and $i = 1$

1. Set $Y_{ki} = True$ and, as consequence, $Y_{ki'} = False$ for all $i' \neq i, i' \in D_k$

2. Formulate MILP/MINLP by using the (HR) formulation of the remaining disjunctions $k' \neq k, k' \in K$ (note that, since $Y_{ki} = True$, the constraints associated with Y_{ki} will be enforced as global constraints, while the ones associated with $Y_{ki'}, i' \neq i, i' \in D_k$ will be removed from the formulation).

3. Solve the continuous relaxation of this problem to optimality, and define z^{ki} as the solution of this problem.

4. Repeat this for every $k \in K$ and $i \in D_k$.

Definition 3.2.1 *For a minimization problem, we define the characteristic value of a disjunction k as follows:*

$$charv_k = min\{z^{ki}\}, \forall i \in D_k$$

It is possible to reduce the problem size and find better bounds of the problem considering the following remarks:

Remark 3.2.2 *$charv_k$ is a lower bound of z.*

Proof. Solving the (HR) reformulation of the original GDP, and relaxing all integrality constraints except the ones corresponding to the disjunction k also yields $charv_k$ (i.e. $charv_k$ is the value of the solution of a relaxation of (GDP)) ■.

Remark 3.2.3 *If z^{ki} is $infeasible$ for a disjunctive term $k \in K, i \in D_k$, then $Y_{ki} = False$ in the original formulation.*

Proof. From the definition, z^{ki} is a continuous relaxation of the problem with $Y_{ki} = True$. If this relaxation is infeasible, then that particular disjunctive term can not be selected (i.e. $Y_{ki} = False$) ■.

As consequence of Remark 3.2.3, the terms and constraints associated to a $k \in K, i \in D_k$ term such that z^{ki} is $infeasible$ can be removed from the original (GDP) formulation.

Remark 3.2.4 *If z^{ki} is $infeasible$ for all $i \in D_k$ for any $k \in K$, then the problem is infeasible.*

Proof. If for any disjunction $k \in K$, all its disjunctive terms $i \in D_k$ are infeasible, then there is no alternative in that disjunction that will make the problem feasible, and therefore, the problem is $infeasible$ ■.

Remark 3.2.5 *A lower bound to the problem z^{lo} that is at least as large as the continuous relaxation of the original (GDP) can be obtained with as follows:*

$$z^{lo} = \max_{k \in K} \{charv_k\}$$

Proof. Trivially follows from 3.2.2 ■.

It is important to note that this process requires the evaluation of $\sum_{k \in K} |D_k|$ LP/NLP problems. Although this preprocessing might be expensive for only eliminating terms and finding good bounds for the problem, the characteristic value of the disjunctions has a major role in the algorithm as will be described in section 3.2.3. Additionally, since most of the structure of the problem does not change while evaluating every term, it might be possible to solve the many LP/NLP problems in a more efficient manner. This issue is not addressed in this thesis. The resulting (GDP) after eliminating infeasible terms is set as (GDP^*), and its continuous relaxation z^*.

Illustration of pre-solve.

In order to illustrate the pre-solve procedure, consider the formulation shown in (3.1).

$$
\begin{aligned}
& min\ z = x_1 + x_2 \\
s.t.\ & \\
& \begin{bmatrix} Y_{11} \\ x_2 \geq 8 + x_1 \\ x_2 = 12 - x_1 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ x_1 \leq 5 \\ x_2 \geq 6 \\ x_2 \leq x_1 + 5 \end{bmatrix} \vee \begin{bmatrix} Y_{13} \\ x_1 \geq 9 \\ x_2 \leq 5 \\ x_2 \geq x_1 - 8 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ 4 \leq x_1 \leq 7 \\ 7 \leq x_2 \leq 8 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ 7 \leq x_1 \leq 11 \\ 2 \leq x_2 \leq 4 \end{bmatrix} \\
& Y_{11} \veebar Y_{12} \veebar Y_{13} \\
& Y_{21} \veebar Y_{22} \\
& x_1, x_2 \in \mathbb{R}^1 \\
& Y_{11}, Y_{12}, Y_{13}, Y_{21}, Y_{22} \in \{True, False\}
\end{aligned}
\tag{3.1}
$$

Problem (3.1) has an optimal solution of $\mathbf{z = 11}$, in which $Y_{12} = True$ and $Y_{21} = True$. The (BM) provides a relaxation of $\mathbf{z^{BM} = 3}$, and the (HR) a relaxation of $\mathbf{z^{HR} = 9.16}$. Following the pre-solve procedure described in 3.2.2, we first set $Y_{11} = True, Y_{12} = False, Y_{13} = False$, and then perform the (HR) reformulation of the remaining of the

problem. Relaxing the integrality constraints, this yields the LP shown in (3.2).

$$
\begin{aligned}
& & min\ z &= x_1 + x_2 \\
& s.t. & x_2 &\geq 8 + x_1 \\
& & x_2 &= 12 - x_1 \\
& & x_1 &= (x_1)_{21} + (x_1)_{22} \\
& & x_2 &= (x_2)_{21} + (x_2)_{22} \\
& & 4 * y_{21} &\leq (x_1)_{21} \leq 7 * y_{21} \\
& & 7 * y_{21} &\leq (x_2)_{21} \leq 8 * y_{21} \\
& & 7 * y_{22} &\leq (x_1)_{22} \leq 11 * y_{22} \\
& & 2 * y_{22} &\leq (x_2)_{22} \leq 4 * y_{22} \\
& & y_{21} + y_{22} &= 1 \\
& & x_1, x_2 &\in \mathbb{R}^1 \\
& & 0 &\leq y_{21}, y_{22} \leq 1
\end{aligned}
\tag{3.2}
$$

Problem (3.2) is infeasible. Performing the same calculation for $Y_{12} = True$ yields a $z^{12} = 10.6$. Repeating this for the remaining disjunctive terms we obtain $z^{13} = 11$, $z^{21} = 11$, $z^{21} = 9.25$. With these values, we assign the characteristic values for each disjunction:

$$charv_1 = min\{z^{11}, z^{12}, z^{13}\} = min\{infeas, 10.6, 11\} = 10.6$$

$$charv_2 = min\{z^{21}, z^{22}\} = min\{11, 9.25\} = 9.25$$

With these two characteristic values, it is possible to set the new lower bound: $z^{lo} = max\{charv_1, charv_2\} = 10.6$. Also, since $z^{11} = infeas$, the term associated with Y_{11} can be eliminated from the original (GDP) formulation. Therefore, problem (3.1) after the

pre-solve becomes (3.3).

$$
\begin{aligned}
& min\ z = x_1 + x_2 \\
& s.t. \\
& \begin{bmatrix} Y_{12} \\ x_1 \leq 5 \\ x_2 \geq 6 \\ x_2 \leq x_1 + 5 \end{bmatrix} \vee \begin{bmatrix} Y_{13} \\ x_1 \geq 9 \\ x_2 \leq 5 \\ x_2 \geq x_1 - 8 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ 4 \leq x_1 \leq 7 \\ 7 \leq x_2 \leq 8 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ 7 \leq x_1 \leq 11 \\ 2 \leq x_2 \leq 4 \end{bmatrix} \\
& Y_{11} \veebar Y_{12} \veebar Y_{13} \\
& Y_{21} \veebar Y_{22} \\
& x_1, x_2 \in \mathbb{R}^1 \\
& Y_{11}, Y_{12}, Y_{13}, Y_{21}, Y_{22} \in \{True, False\}
\end{aligned}
\tag{3.3}
$$

It is clear that (3.3) is a smaller problem than the original problem (3.1), and a new lower bound of $\mathbf{z^{lo} = 10.6}$ has been found. Note that the solution to the problem is **z = 11**, and the (HR) of (3.1) provides a lower bound of $\mathbf{z^{HR} = 9.16}$, so the new lower bound $\mathbf{z^{lo} = 10.6}$ is stronger.

3.2.3 Step 2, 3: Selection of k^*

The algorithm selects over which disjunctions to apply basic steps (k^*). As described by Balas[12] there are heuristics to estimate which basic step will generate the best improvement on a formulation. Furthermore, even if the best first basic step were selected, this does not necessarily imply that after a sequence of basic steps this first one is the best choice. Sawaya and Grossmann[26] and Ruiz and Grossmann[15] propose some heuristics as to when to apply basic steps. For the algorithm proposed in this chapter, we consider three main factors for the selection of these disjunctions:

1) A consequence of basic steps described in Balas' work[12]: A basic step between two

disjunctions that do not share variables in common will not improve the tightness of the formulation.

2) The number of terms in the new disjunction increases exponentially with the number of terms of the selected disjunctions ($|\hat{D}| = |\hat{D}_{k_1}| * ... * |\hat{D}_{k_{iter}}|$). Since the algorithm applies basic steps iteratively over the same disjunction, it is desired to keep the problem size as small as possible.

3) The characteristic value of the disjunctions, obtained in the first step of the algorithm, provides a heuristic on the "tightness" associated which each disjunction. Therefore, disjunctions with higher characteristic values are preferred.

There are several heuristics that can be used for the selection of k^*. In our experience, the best performance was achieved by applying many basic steps with small growth in size. Therefore, the heuristic we found to work the best was to select disjunctions with fewest terms that share variables in common with many other disjunctions (preferably also small ones). This evaluation is done as follows:

0. Initialize $W^k = 0, \forall k \in K$. Set $m = 1$, $n = 2$.

1. If disjunction m shares a variable in common with disjunction n, then $W^m = W^m + \frac{1}{(|D_m|)*(|D_n|)}$; $W^n = W^n + \frac{1}{(|D_m|)*(|D_n|)}$.

2. Set $n = n + 1$. If $n \leq |K|$ go back to 1, else go to 3.

3. Set $m = m + 1$. If $m \leq |K| - 1$ set $n = m + 1$ and go back to 1, else go to 4.

4. Select the disjunction k^* with largest W^k value. If there is a tie, choose the disjunction with the highest characteristic value.

Note that this algorithm gives priority to the number of basic steps that can be applied to a certain disjunction, giving more weight to the basic steps with smaller increase in number of disjunctive terms. If there is a tie with this parameter, it selects the one with highest $charv_k$.

The method for the selection of k^* in step 3 is almost the same as the one for the selection of k^* in step 2. The difference is that a basic step will be applied between this disjunc-

tion k^* and "key disjunction". Therefore, if a disjunction m does not share a variable in common with any $\hat{k}_s \in \hat{K}$ then $W^m = 0$.

3.2.4 Step 4: Analyze and eliminate resulting disjunctive terms

As shown in Section 3.2.1, the basic steps will be applied iteratively over the same disjunction. This means that the number of terms of such a disjunction will grow exponentially after each basic step. Eliminating terms after each basic step helps to maintain small formulations. However, there are two things to consider: first is that this step can become computationally expensive as the algorithm iterates, and second is that eliminating terms has less impact after each iteration. There are two methods that the algorithm uses to eliminate infeasible terms.

First, the terms that result from intersecting a term that was infeasible in the previous iteration, will be infeasible. Second, in order to eliminate resulting disjunctive terms, one LP/NLP is solved for each term in the new disjunction, similarly to the pre-solve. In the initial iterations, $|\hat{D}|$ is small, so the "key disjunction" has a small number of terms. Furthermore, every term that is eliminated reduces the exponential growth of the problem size in the subsequent iterations. As the algorithm iterates and $|\hat{D}|$ becomes larger, the number of LP/NLP evaluation increases, each LP/NLP becomes more expensive, and there will not be many more basic steps, so its impact is limited. For these reasons, step 4 is only applied to the initial iterations, but dropped as the algorithm progresses.

In addition to this, a third method that identifies infeasibility implied by the logic expressions when two of the disjunctive terms are intersected can be used. Such cases are computationally cheap, but there are not necessarily many terms that can be eliminated in this way. Tools such as constraint programming can further improve the performance of this method[27]. This third method is not addressed in this thesis.

3.2.5 Step 5

In step 5 an improper basic step is applied between the "key disjunction" and the global constraints. Using the same concepts described in 3.2.3, the improper basic step is only applied with the global constraints that share at least one variable in common with any $\hat{k}_s \in \hat{K}$.

3.2.6 Step 6: Hybrid reformulation of (GDP)

As presented in the introduction, the MILP/MINLP reformulation of the (GDP) is typically performed through either (BM) or (HR). The reformulation, however, needs not to be strictly one of these; some disjunctions can be reformulated through (BM) while others (HR). The advantage of doing this is that, if the correct disjunctions are selected for the different reformulations, we can obtain a tight relaxation but with a smaller problem size than (HR). Note that in the hybrid reformulation the smallest possible MILP/MINLP is the complete (BM), while the tightest continuous relaxation is achieved through the (HR). Any hybrid lies between the two formulations, both in size of problem and tightness of continuous relaxation.

For the algorithm, we apply convex hull to the "key disjunction" (i.e. we formulate it using (HR)) since the tightness improvement of the basic steps does not hold true for the (BM). The rest of the disjunctions are formulated through (BM), considering that the tightness improvement comes from the basic steps applied in disjunctions $\hat{k}_s \in \hat{K}$.

Figure 3.3 illustrates the idea of the hybrid reformulation. In the (BM) reformulation the problem size is small, but the continuous relaxation (represented by the shaded region) provides a solution Z^{BM} that is far from the optimal solution Z^*. The (HR) provides a tighter continuous relaxation, and therefore the solution to the relaxation Z^{HR} is closer to the optimal solution. As expected, the problem size is considerably larger than (BM). Lastly, in the hybrid reformulation the continuous relaxation is not as tight as the (HR). However, its relaxation provides the same optimal solution as the relaxation of the (HR) $Z^{HY} = Z^{HR}$. This hybrid reformulation is larger than the (BM), but not as large as (HR).

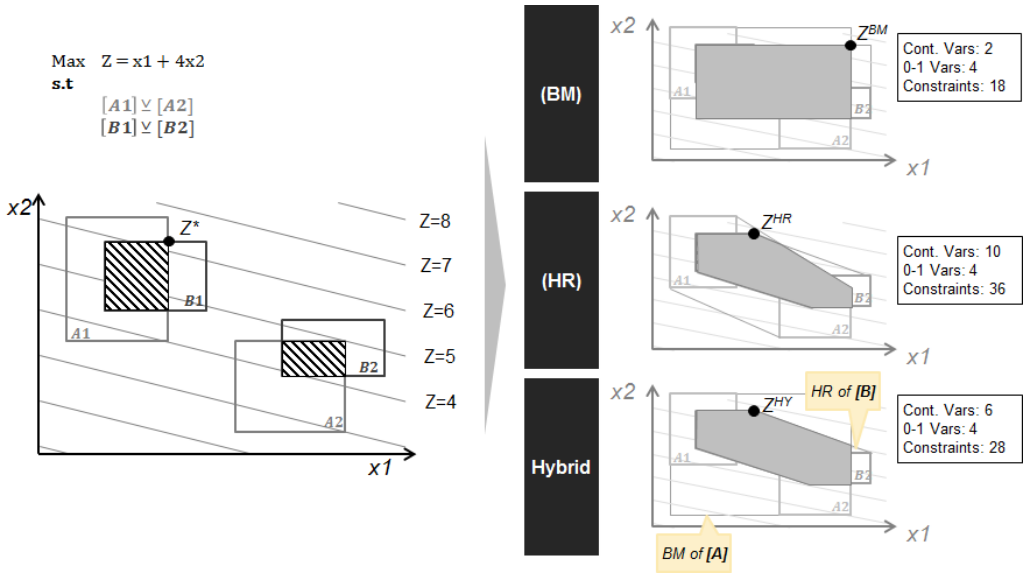

Figure 3.3: Illustration of (BM), (HR) and Hybrid reformulation

3.2.7 Step 7: Rule for iterating

Many different rules can be applied to decide whether or not the algorithm keeps iterating. The most intuitive rules are size of the problem and value of continuous relaxation, since these are the two properties we are trying to improve. The last formulation that was considered to improve the GDP (GDP^*), is the one that is then solved as an MILP/MINLP.

It is also important to note that, in order to identify if the relaxation is improving or not, we check the relaxation of ($GDPH$) and (GDP^*) without using the lower bound found in the pre-solve z^{lo}. This lower bound is added in the last iteration, when (GDP^*) is solved as an MILP/MINLP.

3.3 Illustration of algorithm

In this section we provide an illustration of the algorithm with a simple example. Consider the linear (GDP) in (3.4):

$$
\begin{aligned}
& min\ lt \\
s.t.\quad & lt \geq x_1 + 6 \\
& lt \geq x_2 + 5 \\
& lt \geq x_3 + 4 \\
& lt \geq x_4 + 3 \\
& \begin{bmatrix} Y_{11} \\ x_1 + 6 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ x_2 + 5 \leq x_1 \end{bmatrix} \vee \begin{bmatrix} Y_{13} \\ h_1 - 6 \geq h_2 \end{bmatrix} \vee \begin{bmatrix} Y_{14} \\ h_2 - 7 \geq h_1 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ x_1 + 6 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ x_3 + 4 \leq x_1 \end{bmatrix} \vee \begin{bmatrix} Y_{23} \\ h_1 - 6 \geq h_3 \end{bmatrix} \vee \begin{bmatrix} Y_{24} \\ h_3 - 5 \geq h_1 \end{bmatrix} \\
& \begin{bmatrix} Y_{31} \\ x_1 + 6 \leq x_4 \end{bmatrix} \vee \begin{bmatrix} Y_{32} \\ x_4 + 3 \leq x_1 \end{bmatrix} \vee \begin{bmatrix} Y_{33} \\ h_1 - 6 \geq h_4 \end{bmatrix} \vee \begin{bmatrix} Y_{34} \\ h_4 - 3 \geq h_1 \end{bmatrix} \\
& \begin{bmatrix} Y_{41} \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} Y_{42} \\ x_3 + 4 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} Y_{43} \\ h_2 - 7 \geq h_3 \end{bmatrix} \vee \begin{bmatrix} Y_{44} \\ h_3 - 5 \geq h_2 \end{bmatrix} \\
& \begin{bmatrix} Y_{51} \\ x_2 + 5 \leq x_4 \end{bmatrix} \vee \begin{bmatrix} Y_{52} \\ x_4 + 3 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} Y_{53} \\ h_2 - 7 \geq h_4 \end{bmatrix} \vee \begin{bmatrix} Y_{54} \\ h_4 - 3 \geq h_2 \end{bmatrix} \\
& \begin{bmatrix} Y_{61} \\ x_3 + 4 \leq x_4 \end{bmatrix} \vee \begin{bmatrix} Y_{62} \\ x_4 + 3 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} Y_{63} \\ h_3 - 5 \geq h_4 \end{bmatrix} \vee \begin{bmatrix} Y_{64} \\ h_4 - 3 \geq h_3 \end{bmatrix} \\
& Y_{k1} \veebar Y_{k2} \veebar Y_{k3} \veebar Y_{k4} \qquad k = 1, ..., 6 \\
& 0 \leq x_1 \leq 12;\ 0 \leq x_2 \leq 13;\ 0 \leq x_3 \leq 14;\ 0 \leq x_4 \leq 15 \\
& 6 \leq h_1 \leq 10;\ 7 \leq h_2 \leq 10;\ 5 \leq h_3 \leq 10;\ 3 \leq h_4 \leq 10 \\
& x_j, h_j \in \mathbb{R}^1 \qquad j = 1, 2, 3, 4 \\
& Y_{ki} \in \{True, False\} \qquad k = 1, ..., 6, i = 1, 2, 3, 4
\end{aligned}
\tag{3.4}
$$

This problem has an optimal solution of **lt = 15**, in which $Y_{12}, Y_{21}, Y_{32}, Y_{41}, Y_{53}, Y_{63} = True$. The continuous relaxation of its (BM) reformulation has a value of $\mathbf{z^{BM} = 6.0}$, and the (HR) provides a relaxation of $\mathbf{z^{HR} = 8.3}$.

Step 1. After applying the pre-solve as described in 3.2.2, the terms $\{13, 14, 23, 24, 43, 44\}$

Table 3.1: Weight parameter calculation for the disjunctions.

			Accumulated weight					
m	n	common vars?	W^1	W^2	W^3	W^4	W^5	W^6
1	2	yes	0.25	0.25	0	0	0	0
1	3	yes	0.375	0.25	0.125	0	0	0
1	4	yes	0.625	0.25	0.125	0.25	0	0
1	5	yes	0.75	0.25	0.125	0.25	0.125	0
1	6	no	0.75	0.25	0.125	0.25	0.125	0
2	3	yes	0.75	0.375	0.125	0.25	0.125	0
2	4	yes	0.75	0.625	0.25	0.5	0.125	0
2	5	no	0.75	0.625	0.25	0.5	0.125	0
2	6	yes	0.75	0.75	0.25	0.5	0.125	0.125
3	4	no	0.75	0.75	0.25	0.5	0.125	0.125
3	5	yes	0.75	0.75	0.3125	0.5	0.1875	0.125
3	6	yes	0.75	0.75	0.375	0.5	0.1875	0.1875
4	5	yes	0.75	0.75	0.375	0.625	0.3125	0.1875
4	6	yes	0.75	0.75	0.375	0.75	0.3125	0.3125
5	6	yes	0.75	0.75	0.375	0.75	0.375	0.375

are found to be $infeasible$, so they are removed from the original (GDP) formulation. GDP^* is then (3.4) without terms $\{13, 14, 23, 24, 43, 44\}$. Also, the characteristic values of the disjunctions $k = 1, ..., 6$ are, respectively: $charv_k = (11, 10, 8.3, 9.6, 8.3, 8.3)$. A new lower bound z^{lo} is also found, since $max(charv_k) = 11$, which is larger than the relaxation of the (HR) $z^* = 8.3$. Set $iter = 1$.

Step 2. The selection of k^* is performed by assigning a weight to each disjunction as described in 3.2.3. In this case $W = (0.75, 0.75, 0.38, 0.75, 0.38, 0.38)$. Table 3.1 shows the iterations to obtain W, following the procedure described in section 3.2.3. Disjunctions 1, 2 and 4 have the same weight, but the first disjunction has the highest $charv_k$. Therefore, disjunction 1 is chosen as k^*. $\hat{k_1} = 1$; $\hat{K} = \{1\}$; $\hat{D}_1 = \{1, 2\}$.

Step 3. $iter = 2$. To find new k^*, weighting factors W^k for $k \neq 1$ are also assigned $W = (1, 0.5, 1, 0.5, 0)$. Disjunction 2 and 4 have the same W^k, but since $charv_2 > charv_4$, disjunction 2 is selected as k^*. Also note that $W^6 = 0$ since disjunction 6 and disjunction 1 do not share variables in common.

Set $\hat{k}_2 = 2$; $\hat{K} = \{1, 2\}$; $\hat{D}_2 = \{1, 2\}$; $\hat{D} = \{(1, 1), (1, 2), (2, 1), (2, 2)\}$.

For this step, it is possible to represent the "key disjunction" as follows:

$$\begin{bmatrix} \hat{Y}_{1,1} \\ x_1 + 6 \leq x_2 \\ x_1 + 6 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,2} \\ x_1 + 6 \leq x_2 \\ x_3 + 4 \leq x_1 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,1} \\ x_2 + 5 \leq x_1 \\ x_1 + 6 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,2} \\ x_2 + 5 \leq x_1 \\ x_3 + 4 \leq x_1 \end{bmatrix} \tag{3.5}$$

The additional constraints that are added so that the new disjunctive variables $\hat{y}_i$ can be continuous are shown in (3.6):

$$\begin{aligned} y_{11} &= \hat{y}_{1,1} + \hat{y}_{1,2} \\ y_{12} &= \hat{y}_{2,1} + \hat{y}_{2,2} \\ y_{21} &= \hat{y}_{1,1} + \hat{y}_{2,1} \\ y_{22} &= \hat{y}_{1,2} + \hat{y}_{2,2} \\ y_{11}, y_{12}, y_{21}, y_{22} &\in \{0, 1\} \\ 0 \leq \hat{y}_{1,1}, \hat{y}_{1,2}, \hat{y}_{2,1}, \hat{y}_{2,2} &\leq 1 \end{aligned} \tag{3.6}$$

Step 4. All the resulting terms in disjunction (3.5) are feasible, so $INFEAS_2 = \emptyset$.

Step 5. Select global equations to which apply a basic step $\hat{E} \in E$. In the example, the first three global constraints share a variable in common with the "key disjunction". It is possible to represent the resulting disjunction after the application of the improper basic step as shown in (3.7).

$$\begin{bmatrix} \hat{Y}_{1,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_1 + 6 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_3 + 4 \leq x_1 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_1 + 6 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_3 + 4 \leq x_1 \end{bmatrix} \tag{3.7}$$

Step 6. The hybrid reformulation of this example is performed by applying (HR) to the "key disjunction" and (BM) in the remaining disjunctions.

Step 7. The relaxed solution of (3.7) is 11. Since relaxed (3.7)$>$ 8.3, set $z^* = 11$,

GDP^*=(3.7). Note that, as explained in section 3.2.7, the lower bound found in the presolve ($z^{lo} = 11$) is not used to evaluate the improvement in the formulation. This lower bound will be added only in the last iteration when (GDP) is solved as MILP/MINLP.

Since it improved, the algorithm proceeds to the next iteration.

In the next iteration, a basic step between the "key disjunction" and disjunction 4 is applied. The resulting disjunction is illustrated in (3.8).

$$\begin{bmatrix} \hat{Y}_{1,1,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_1 + 6 \leq x_3 \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,1,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_1 + 6 \leq x_3 \\ x_3 + 4 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,2,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_3 + 4 \leq x_1 \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{1,2,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_1 + 6 \leq x_2 \\ x_3 + 4 \leq x_1 \\ x_3 + 4 \leq x_2 \end{bmatrix} \vee$$
$$\vee \begin{bmatrix} \hat{Y}_{2,1,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_1 + 6 \leq x_3 \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,1,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_1 + 6 \leq x_3 \\ x_3 + 4 \leq x_2 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,2,1} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_3 + 4 \leq x_1 \\ x_2 + 5 \leq x_3 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{2,2,2} \\ lt \geq x_1 + 6 \\ lt \geq x_2 + 5 \\ lt \geq x_3 + 4 \\ x_2 + 5 \leq x_1 \\ x_3 + 4 \leq x_1 \\ x_3 + 4 \leq x_2 \end{bmatrix} \tag{3.8}$$

In this case, the term associated with $\hat{Y}_{1,2,1}$ and $\hat{Y}_{2,1,2}$ are infeasible, so $INFEAS_2 = \{(1,2,1),(2,1,2)\}$.

Also, additional constraints need to be added in the MILP reformulation to avoid the increase in binary variables, as described in earlier. These constraints (after setting $\hat{y}_{1,2,1} =$

$\hat{y}_{2,1,2} = 0$;) are shown in (3.9).

$$\begin{aligned}
y_{11} &= \hat{y}_{1,1,1} + \hat{y}_{1,1,2} + \hat{y}_{1,2,2} \\
y_{12} &= \hat{y}_{2,1,1} + \hat{y}_{2,2,1} + \hat{y}_{2,2,2} \\
y_{21} &= \hat{y}_{1,1,1} + \hat{y}_{1,1,2} + \hat{y}_{2,1,1} \\
y_{22} &= \hat{y}_{1,2,2} + \hat{y}_{2,2,1} + \hat{y}_{2,2,2} \\
y_{41} &= \hat{y}_{1,1,1} + \hat{y}_{2,1,1} + \hat{y}_{2,2,1} \\
y_{42} &= \hat{y}_{1,1,2} + \hat{y}_{1,2,2} + \hat{y}_{2,2,2} \\
y_{11}, y_{12}, y_{21}, y_{22}, y_{41}, y_{42} &\in \{0, 1\} \\
0 \leq \hat{y}_{1,1,1}, \hat{y}_{1,1,2}, \hat{y}_{1,2,2}, \hat{y}_{2,1,1}, \hat{y}_{2,2,1}, \hat{y}_{2,2,2} &\leq 1
\end{aligned} \tag{3.9}$$

The continuous relaxation of the hybrid MILP reformulation of this iteration is $\mathbf{z^{iter=2}} = \mathbf{15}$. Since it improved, the algorithm proceeds to another iteration.

The third iteration involves a basic step between the "key disjunction" and disjunction 5. This results in a disjunction with 32 terms, of which 8 of them are infeasible (the 8 terms that result from intersecting terms $\{(1,2,1),(2,1,2)\}$ with the 4 terms of disjunction 5). This $(GDPH)$ also has a relaxation of $\mathbf{z^{iter=3}} = \mathbf{15}$. Since there is no improvement, the formulation obtained in iteration 2 is selected as GDP^* and solved as MILP. Note that the continuous relaxation of this formulation provides a lower bound that has the same value as the optimal solution of the problem $z^{iter=2} = z^* = 15$. However, the values for y_{ki} are not integer, so the MILP solver requires to evaluate some nodes to find the integer solution.

3.4 Results

In this section we present the computational results of applying the algorithm described in section 3.2 to different instances of the examples that can be found in Appendix A and B. The algorithm uses the heuristics for selecting k^* described in 3.2.3. There are many rules that could be set for deciding if the algorithm moves to the next iteration or not. In this study the rule that was used is as follows. If the continuous relaxation does not improve after 3 iterations, or the number of constraints is more than double than the

original, or the number of disjunctive terms in D^k is larger than half of the number of original total disjunctive terms, then proceed to solve GDP^*; else keep iterating. Note that a formulation can in fact more than double its size, but that can occur only in the last iteration. Afterwards the algorithm proceeds to the next step.

Thirty six instances were solved. The instances were generated by defining problem size and randomly generating the parameters of the problems (e.g. in Stpck the number of rectangles was set, but width and length of the rectangles was randomly generated). The ϵ has a value of 10^{-4}, and the Big M parameter was estimated using the most basic solution for the given data (e.g. in the strip packing problem the most basic solution is to pack one rectangle after the other, though is not the optimal). The only exception are Batch instances, where the "optimal" Big M parameter was used, and which can be found in the CMU-IBM MINLP library[20].

The instances are solved using branch and bound methods, Gurobi 5.5 for the linear GDP problems and SBB for the convex GDP with CONOPT as the NLP solver. Cuts and presolve were deactivated in Gurobi for all linear instances. The algorithm and models were implemented in GAMS[21] and solved in an Intel(R) Core(TM) i7 CPU 2.93 GHz and 4 GB of RAM.

We first present and discuss the general performance of the algorithm shown in Figures 3.4, 3.5 and 3.6. We then discuss some characteristics of the different instances presented in Figures 3.7 and 3.8. Finally, we describe the behavior of the algorithm in Figure 3.9.

Figure 3.4 shows the percentage of problems solved vs. time for the (BM), (HR) and the algorithm. The time for the algorithm includes the preprocessing, the steps for improving the formulation, and the solution to the MINLP. The figure shows that the algorithm performs in general better than the (BM) and (HR) reformulations. In the smaller instances this is not true, and one of the main reasons for this is that the algorithm is programmed with a high level language (GAMS). Thus, if the algorithm is implemented in a lower level programming language its performance is expected to improve. Other improvements to the algorithm, such as the ones mentioned earlier in the chapter (taking advantage of problem structure to reduce presolving time, using constraint programming and logic con-

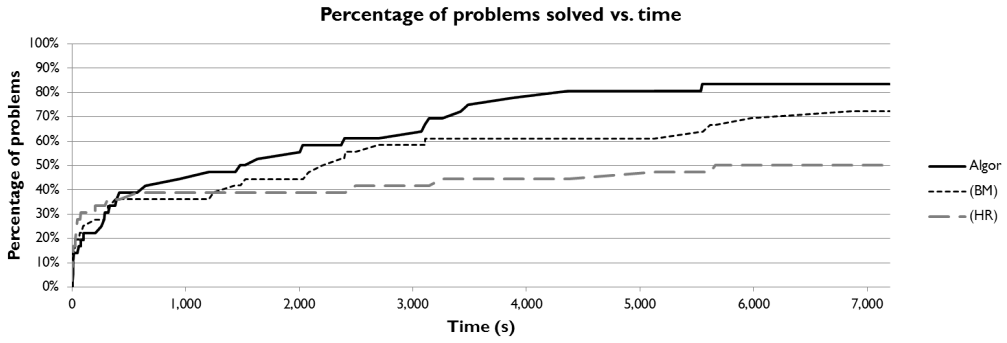

Figure 3.4: Percentage of problems solved vs. time for the test instances

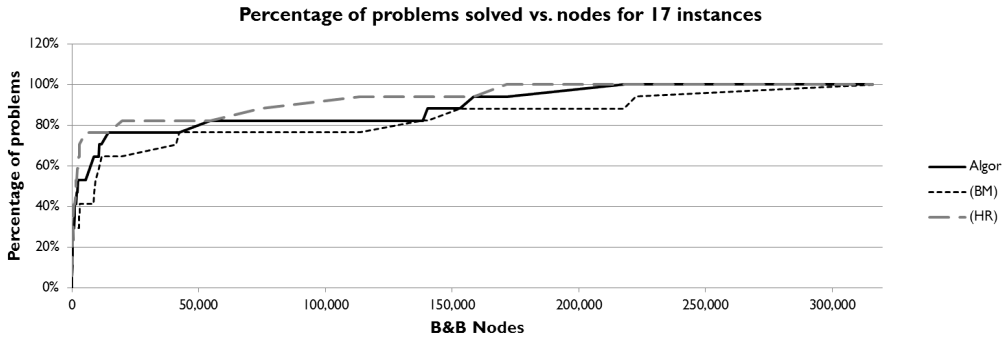

Figure 3.5: Number of nodes evaluated to achieve optimum, for the 17 instances where the three formulations did so. Excludes S-Pck12 for comparison purposes, in order to avoid plotting over 1^6 nodes

cepts to eliminate infeasible terms, and improving heuristics for iterating rule and selection of basic steps) might further improve the algorithm's performance.

Figure 3.5 shows the number of nodes that where evaluated to find and prove optimality (within 0.1 % gap) for the 17 instances in which the three formulations did so. As expected the (HR) requires fewer nodes than the (BM) to achieve optimality. The algorithm needs fewer number of nodes than the (BM), but more than the (HR) in these 17 instances.

Figure 3.6 shows the percentage of problems vs. number of constraints. The figure shows that the number of constraints in the (BM) is in general smaller than (HR) and the formulation obtained through the algorithm. The MILP/MINLP obtained with the algorithm has

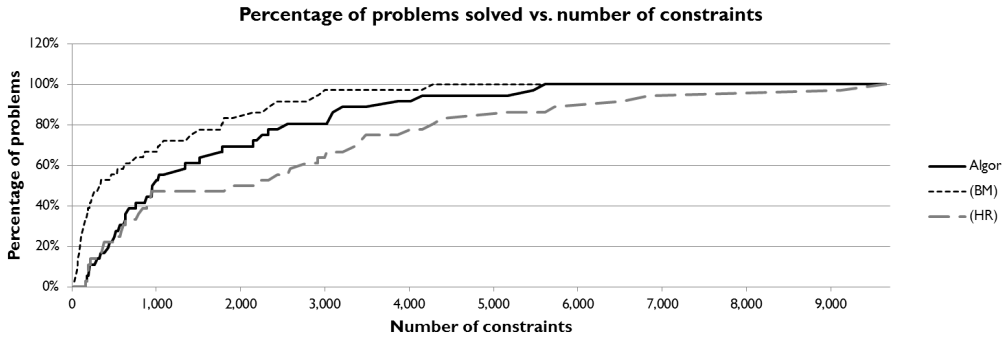

Figure 3.6: Number of constraints for the different formulations

fewer constraints than the (HR). It is important to note that for most problems the continuous relaxation improved after applying the algorithm, as shown in Figure 3.7. Therefore, having not much larger or even smaller problem sizes represents an important improvement in the problem formulation.

Figure 3.7 shows the continuous relaxation, the number of constraints, and the number of variables for the different instances, comparing the (BM) and (HR) formulations of the original problem with the formulation obtained with the proposed algorithm. It can be seen that on 28 of the 36 instances the relaxation improves after applying the algorithm, in the other 8 it has the same value as the (HR). In few cases, such as C-Lay-3-2 (where the solution is **41,573**, the (BM) and (HR) relaxations are **0**, and the relaxation after the algorithm is **2,200**), the improvement in the relaxation is small. In most of the cases the gap improves around 20%-40%, for example C-Lay-5-2 where the solution is **11,472**, the (BM) and (HR) relaxations are **0** and the relaxation after the algorithm is **4,203**. In the extreme case of Process-8, the algorithm provides a relaxation of **1,098**, which is actually the optimal value of the objective function, (HR) provides a good relaxation of **1,079**, but still with some gap, while (BM) provides a very poor relaxation of **0**.

The algorithm produces the largest number of variables and constraints in 11 out of the 36 instances, and the size of these problems is not much larger than that of the (HR). The (HR) produces the largest formulations in the rest of the instances. The number of binary variables in (HR) and (BM) is the same as expected. However, there are fewer

		Relaxation			Constraints			Variables/binary		
Instance	**Solution**	**(BM)**	**(HR)**	**Algorithm**	**(BM)**	**(HR)**	**Algor**	**(BM)**	**(HR)**	**Algor**
Batch101006	769,440	734,943	745,909	**750,769**	1,020	**1,897**	1,004	279/129	**789/129**	276/129
Batch121208	1,241,126	1,202,365	1,217,603	**1,223,599**	1,512	**2,781**	1,514	407/203	**1163/203**	415/203
Batch151208	1,543,472	1,499,913	1,514,948	**1,519,320**	1,782	**3,348**	1,784	446/203	**1334/203**	454/203
Batch181210	2,042,327	2,006,860	**2,021,546**	**2,021,546**	2,148	**4,011**	2,150	533/251	**16001/251**	543/251
Batch201210	2,295,349	2,255,304	2,272,082	**2,277,093**	2,328	**4,389**	2,330	559/251	**1715/251**	569/251
C-Lay-3-2	41,573	0	0	**2,200**	57	195	**568**	32/18	92/18	**240/18**
C-Lay-3-3	26,669	0	0	**2,200**	69	219	**338**	35/21	101/21	**139/20**
C-Lay-4-2	8,469	0	0	**2,769**	93	349	**952**	54/32	166/32	**390/32**
C-Lay-4-3	9,746	0	0	**3,988**	109	381	**1,030**	58/36	178/36	**394/29**
C-Lay-5-2	11,472	0	0	**4,203**	138	530	**1,345**	82/50	262/50	**546/50**
C-Lay-5-3	20,799	0	0	**5,372**	158	588	**1,341**	87/55	277/55	**551/53**
C-Lay-5-4	10,876	0	0	**2,695**	178	608	**952**	92/60	292/60	**392/59**
C-Lay-6-2	11,914	0	0	**2,986**	192	**792**	503	116/72	**380/72**	240/67
C-Lay-6-3	14,523	0	0	**3,546**	216	**840**	518	122/78	**398/78**	246/73
C-Lay-6-4	15,084	0	0	**3,792**	240	888	**1,755**	128/84	416/84	**720/74**
C-Lay-6-5	26,521	0	0	**3,660**	264	**936**	883	134/90	**434/90**	382/82
Dice0605	12	2	**6**	**6**	2,432	**5,162**	3,090	1292/1260	**2912/1260**	2000/1051
Dice0606	12	2	**6**	**6**	1,802	**6,842**	3,090	1766/1728	**3926/1728**	2002/1051
Dice0804	24	4	**8**	**8**	2,914	**6,530**	3,860	1570/1536	**3618/1536**	2676/1249
Dice0805	21	3	**8**	**8**	4,282	**9,122**	5,468	2282/2240	**5162/2240**	3544/1881
Dice1203	56	10	**15**	**15**	2,998	**9,650**	5,610	2198/2160	**5222/2160**	3892/1690
DiceH0605	12	3	**6**	**6**	632	**2,432**	444	1292/1260	**2012/1260**	1294/1250
DiceH0606	12	3	**6**	**6**	758	**2,918**	2,253	1766/1728	**2630/1728**	2766/1718
DiceH0804	24	5	**8**	**8**	866	**3,426**	2,556	1570/1536	**2594/1536**	2706/1522
DiceH0805	21	4	**8**	**8**	1,082	**4,282**	3,205	2282/2240	**3562/2240**	3706/2226
DiceH1203	56	12	**15**	**15**	1,406	**5,726**	4,149	2198/2160	**3926/2160**	4042/2138
F-lay-3	49	31	31	**42**	27	**195**	160	28/12	**124/12**	88/12
F-lay-4	20	11	12	**15**	45	381	**632**	44/24	236/24	**336/24**
F-lay-5	63	35	35	**46**	68	**633**	179	64/40	**389/40**	121/40
F-lay-6	80	37	38	**52**	102	**942**	208	94/60	**574/60**	146/60
Process-12	1,250	0	1,227	**1,248**	121	188	**635**	71/12	151/12	**431/11**
Process-8	1,098	0	1,079	**1,098**	84	163	**201**	48/8	116/8	**120/7**
S-Pck12	35	9	13	**17**	344	**2,192**	299	290/264	**1346/264**	298/204
S-Pck13	59	10	14	**28**	327	**2,589**	675	340/312	**1588/312**	481/202
S-Pck14	48	9	13	**25**	471	**3,019**	759	394/364	**1850/364**	538/272
S-Pck15	40	10	12	**15**	542	**3,482**	433	452/420	**3482/2132**	460/296

Figure 3.7: Relaxation, number of constraints and variables for the (BM), (HR) and algorithm formulations

binary variables in some of the formulations derived from the algorithm. In these cases, the algorithm is able to eliminate some disjunctive terms during the first pre-solving step, so that the binary variables associated with these terms are removed from the formulation.

Figure 3.8 shows solution time, optimality gap and number of nodes evaluated in the branch and bound tree. Note that the proposed algorithm requires fewest number of nodes in 18 out of the 30 instances in which the number of nodes can be compared. The performance is measured as solution time, or gap in the cases where the models did not find and prove the optimal solution after two hours. The algorithm performs the best in 15 of the 36 instances, and worst in 12 instances. However, the algorithm performs relatively close to the best performer in all instances, except in Batch151208. For the Batch problems the

(HR) formulation performs the best. The (BM) and (HR) perform similarly in the first two instances. The third one is the only instance in all test problems in which the algorithm does much worse than the (BM) and (HR). In the last two instances of the Batch problems the algorithm performs much better than the (BM). For the C-Lay problems the (BM) generally performs the best in the smaller instances (C-Lay-3-2 to C-Lay-5-4). In the larger instances the algorithm performs better than the (BM) and (HR). In Dice and DiceH the algorithm performs the best, while the (HR) performs the worst. For the F-Lay problem the (BM) performs the best, while the (HR) performs the worst. For Process (HR) is the best formulation, and the algorithm the worst, but note that the time to solve is very fast, so the presolve and rest of the algorithm takes much more time than actually solving the MINLP. For the S-Pck problems the (BM) performs the best, then the algorithm, and the (HR) the worst. Note that in C-Lay-6-2, C-Lay-6-3, and the larger instances of Dice and DiceH the (BM) and (HR) reformulations do not solve to optimality within two hours while the algorithm does. It is also important to note that in the examples related to process design (Process and Batch) the (HR) performs much better than the (BM), while in the packing (C-Lay, F-Lay and S-Pck) and Dice the (BM) is much better than the (HR).

Figure 3.9 shows the behavior of the algorithm. The first three columns show the time that the algorithm spends in applying the pre-solve, performing the iterative basic steps over the "key disjunction", and how much time the solver takes to solve the resulting MILP/MINLP. The next two columns show how many iterations the algorithm performed, and what was the criteria to stop iterating. From this column it can be seen that in most of the instances the criteria to stop iterating is the resulting problem size, while only in a few is the lack of improvement in the relaxation after three iterations. The last two columns show the number of proper and improper basic steps. It is interesting to note that, in general, few proper basic steps are applied, but many improper ones are. The improvement in the relaxation with few proper basic steps, but many improper ones, is consistent with the suggestions from previous work[12,15].

		Total solution time (s)			Optimality gap (%)			Number of B&B nodes		
Instance	**Type**	**(BM)**	**(HR)**	**Algor**	**(BM)**	**(HR)**	**Algor**	**(BM)**	**(HR)**	**Algor**
Batch101006	MINLP	204	**12**	287	0.1	0.1	0.1	9,424	**300**	8,553
Batch121208	MINLP	286	**77**	391	0.1	0.1	0.1	10,551	**1,554**	6,886
Batch151208	MINLP	319	**314**	2,007	0.0	0.1	0.1	11,619	**5,400**	54,549
Batch181210	MINLP	1,523	**205**	330	0.1	0.0	0.1	41,053	3,020	**983**
Batch201210	MINLP	5,980	**44**	643	0.0	0.1	0.1	153,039	**692**	10,769
C-Lay-3-2	MINLP	**3**	6	12	0.0	0.0	0.0	282	379	**187**
C-Lay-3-3	MINLP	**4**	10	17	0.0	0.0	0.0	462	449	**389**
C-Lay-4-2	MINLP	**24**	35	61	0.0	0.0	0.0	3,039	**2,038**	2,329
C-Lay-4-3	MINLP	71	**31**	78	0.0	0.0	0.0	8,857	1,571	**1,044**
C-Lay-5-2	MINLP	**383**	574	417	0.1	0.1	0.1	42,337	19,614	**14,286**
C-Lay-5-3	MINLP	2,696	2,498	**2,031**	0.1	0.1	0.1	316,060	**74,602**	217,847
C-Lay-5-4	MINLP	**2,084**	5,124	5,549	0.0	0.1	0.1	222,736	**113,571**	158,556
C-Lay-6-2	MINLP	5,534	7,200	**3,073**	0.1	21.8	0.1	578,938	312,185	**233,685**
C-Lay-6-3	MINLP	6,854	7,200	**3,107**	0.1	30.9	0.1	712,777	277,900	**224,824**
C-Lay-6-4	MINLP	7,200	7,200	7,200	34.5	26.2	**10.8**	573,910	236,327	187,354
C-Lay-6-5	MINLP	7,200	7,200	7,200	**97.9**	300.7	107.0	546,010	54,456	182,880
Dice0605	MILP	2,370	7,200	**948**	0.0	350.0	0.0	739,393	596,342	**16,043**
Dice0606	MILP	2,405	7,200	**1,203**	0.0	414.3	0.0	363,657	446,270	**13,104**
Dice0804	MILP	**1,436**	7,200	1,480	0.0	433.3	0.0	164,227	518,659	**14,802**
Dice0805	MILP	7,200	7,200	**3,489**	50.0	540.0	0.0	1,319,930	290,636	**29,728**
Dice1203	MILP	7,200	7,200	**4,371**	12.0	380.0	0.0	570,555	391,152	**81,598**
DiceH0605	MILP	3,108	7,200	**284**	0.0	300.0	0.0	1,904,882	2,220,152	**39,232**
DiceH0606	MILP	5,619	7,200	**2,398**	0.0	414.3	0.0	2,124,442	1,621,400	**831,391**
DiceH0804	MILP	7,200	7,200	**3,879**	433.3	276.5	0.0	5,419,210	1,806,748	**843,383**
DiceH0805	MILP	7,200	7,200	**3,422**	326.7	481.8	0.0	2,884,668	970,273	**487,549**
DiceH1203	MILP	7,200	7,200	7,200	200.0	311.4	**22.2**	1,433,590	953,789	551,692
F-lay-3	MINLP	**1**	2	7	0.0	0.0	0.0	**102**	110	**102**
F-lay-4	MINLP	**27**	48	101	0.0	0.1	0.1	2,834	2,768	**1,956**
F-lay-5	MINLP	**1,247**	5,660	1,629	0.1	0.1	0.1	**138,443**	171,486	140,446
F-lay-6	MINLP	7,200	7,200	7,200	23.4	43.4	**17.5**	546,524	97,396	432,925
Process-12	MINLP	2	**1**	14	0.0	0.0	0.0	168	**48**	232
Process-8	MINLP	1	**0**	9	0.0	0.0	0.0	24	**16**	18
S-Pck12	MILP	**101**	3,257	260	0.0	0.0	0.0	**1,026,258**	2,630,563	1,239,283
S-Pck13	MILP	7,200	7,200	7,200	**13.5**	1.7	**13.5**	28,179,292	4,600,552	14,313,072
S-Pck14	MILP	**2,205**	7,200	3,142	0.0	4.3	0.0	17,864,367	2,918,216	10,168,644
S-Pck15	MILP	7,200	7,200	7,200	11.1	21.2	**8.1**	21,255,244	3,329,851	56,619,228

Figure 3.8: Solution time, optimality gap and B&B nodes for the (BM), (HR) and algorithm formulations

	Time (s)			Algorithm		Basic Steps	
Instance	Pre	Basic S	MINLP	Iter	Criteria	Proper	Improper
Batch101006	101	2.4	184	1	Size	0	11
Batch121208	209	1.7	180	1	Size	0	13
Batch151208	235	1.5	1,771	1	Size	0	16
Batch181210	295	2.3	33.4	1	Size	0	19
Batch201210	285.2	2.1	356	1	Size	0	21
C-Lay-3-2	7.4	0.6	3.5	1	Size	1	12
C-Lay-3-3	8.6	0.5	8.1	1	Size	1	12
C-Lay-4-2	14.6	0.6	46.2	1	Size	1	24
C-Lay-4-3	18.5	4.2	55.1	3	Size	3	24
C-Lay-5-2	20.9	1.8	394	1	Size	1	36
C-Lay-5-3	30.5	0.8	2,000	1	Size	1	36
C-Lay-5-4	26.5	1.9	5,521	1	Size	1	36
C-Lay-6-2	28.3	1.8	3,043	3	Size	2	36
C-Lay-6-3	32.7	1.9	3,072	2	Size	1	36
C-Lay-6-4	40.4	1.2	7,200	1	Size	1	48
C-Lay-6-5	1,045	1.4	7,200	1	Size	1	36
Dice0605	745	13.5	189	3	Size	0	4
Dice0606	1,055	9.6	139	3	Size	0	4
Dice0804	1,072	11.7	397	3	Size	0	4
Dice0805	2,183	23.5	1,282	3	Size	0	4
Dice1203	2,207	17.3	2,147	3	Size	0	4
DiceH0605	160	14.3	110	3	Size	0	5
DiceH0606	221	9.8	2,167	3	Size	0	6
DiceH0804	285	10.9	3,583	3	Size	0	6
DiceH0805	401	22.2	2,999	3	Size	0	6
DiceH1203	613	16.1	6,587	3	Size	0	8
F-lay-3	5.2	0.5	0.9	1	Size	0	6
F-lay-4	9.6	4.1	87.6	1	Size	1	9
F-lay-5	20.7	4.7	1,603	1	Size	0	6
F-lay-6	30.0	3.7	7,200	1	Size	0	6
Process-12	11.4	1.9	0.6	2	Size	2	9
Process-8	6.5	1.6	0.5	3	Size	1	6
S-Pck12	108	9.7	142	3	Iterations	0	2
S-Pck13	148	73.7	7,052	7	Size	5	4
S-Pck14	154	51.7	2,935	7	Size	5	4
S-Pck15	232	12.4	6,968	3	Iterations	0	2

Figure 3.9: Behavior of the algorithm for each instance

3.5 Conclusion

In this chapter, we have proposed an automated algorithm that improves the relaxation of GDP formulations compared to the common BM and HR reformulations. We have developed a pre-solve procedure that reduces problem formulations, generates stronger bounds, and provides a value that helps in the selection of disjunctions to which basic steps should be applied. We have presented an iterative procedure in which basic steps are applied over the same disjunction in order to improve the continuous relaxation of the problem. In this step of the algorithm, we have also proposed some heuristics on the selection of these disjunctions. We show that a hybrid reformulation can provide some of the advantages of both BM and HR, and that the selection of disjunctions reformulated through BM or HR is simple and clear for the proposed algorithm. Finally, we have applied the proposed method to improve the formulation of different numerical examples of Generalized Disjunctive Programs. These results show that the algorithm provides better formulations, in the sense that they yield strong lower bounds without an excessive increase in problem size. Furthermore, solution times for large instances are reduced by using the algorithm. Finally, even though the algorithm shows promising results, further research in the selection of key and secondary disjunctions, as well as improvements in the presolve and infeasible term detection, is needed to achieve faster performance. Furthermore, in problems with a large number of disjunctive terms the pre-solve may not be a good option. For these cases different rules for the selection of disjunctions would have to be derived.

Chapter 4

Cutting plane algorithm for convex GDP

4.1 Introduction

In this chapter, we follow a different approach to exploit the advantages of having small but weak formulations (BM), and strong but larger formulations (HR after the application of some basic steps). In order to do this, we propose a cutting plane algorithm for convex GDP problems. The algorithm iteratively derives valid inequalities (or cutting planes) for the BM reformulations. These inequalities cut-off sections of the feasible region of the continuous relaxation of the BM, but they do not cut-off any valid region of the stronger formulation. Once the cuts stop having a relevant impact in the improvement of the relaxation, the final MINLP is generated by using the BM of the GDP and including all the generated cuts. This MINLP is then solved using traditional methods. In the proposed algorithm, the cutting plane methodology is used before a branch and bound method is applied. Therefore, it can be considered as a pre-processing of the problem in the GDP space. It is important to note that, within the context of disjunctive branch and bound, these cuts could be derived at any node. The addition of this cuts would in fact yield a disjunctive branch and cut algorithm. However, the disjunctive branch and cut algorithm is out of the scope of this chapter.

This chapter is organized as follows. Section 4.2 presents the proposed algorithm, which is illustrated in detailed with an example in Section 4.3. Section 4.4 presents the statistics, results and performance of different test examples.

4.2 Cutting plane algorithm to improve GDP formulations

A method for using the strong extended reformulation, obtained through the application of basic steps, is to derive cutting planes for the BM reformulation [14,28,29]. The main idea of the cutting plane method is to solve the continuous relaxation of the BM of the convex GDP, and use the strong formulation of the HR after basic steps to derive cutting planes. The cutting planes are determined by solving a separation problem (which is an NLP), in which the feasible solution corresponds to the continuous relaxation of the HR formulation after a sequence of basic steps.

Without loss of generality, any convex GDP can be formulated as follows:

$$
\begin{aligned}
& min\ x_n \\
s.t.\quad & g(x) \leq 0 \\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ r_{ki}(x) \leq 0 \end{bmatrix} && k \in K \\
& \bigvee_{i \in D_k} Y_{ki} && k \in K \\
& \Omega(Y) = True \\
& x^{lo} \leq x \leq x^{up} \\
& x \in \mathbb{R}^n \\
& Y_{ki} \in \{True, False\} \quad k \in K, i \in D_k
\end{aligned}
\tag{4.1}
$$

where, $g(x)$ and $r_{ki}(x)$ are convex functions.

Let (F-HR) be the feasible region of the continuous relaxation of the (HR) reformulation,

and (P-HR) the projection of (F-HR) to the original space. Let (R-BM) be the continuous relaxation of the (BM) reformulation of (4.1). Let (F-BM) be the feasible region and $z^{BM} = (x^{BM}, y^{BM})$ the optimal solution of (R-BM). Also, it is possible to define the feasible region of the continuous relaxation (HR) of the GDP after application of basic steps with the following constraints:

$$g_e(x) \leq 0 \qquad E\backslash\tilde{E} \qquad (4.2a)$$

$$x = \sum_{i \in D_k} \nu^{ki} \qquad k \in K\backslash\tilde{K} \qquad (4.2b)$$

$$y_{ki} r_{ki}(\nu^{ki}/y_{ki}) \leq 0 \qquad k \in K\backslash\tilde{K}, i \in D_k \qquad (4.2c)$$

$$\sum_{i \in D_k} y_{ki} = 1 \qquad k \in K \qquad (4.2d)$$

$$Hy \geq h \qquad (4.2e)$$

$$x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} \qquad k \in K\backslash\tilde{K}, i \in D_k \qquad (4.2f)$$

$$0 \leq y_{ki} \leq 1 \qquad k \in K, i \in D_k \qquad (4.2g)$$

$$x = \sum_{\hat{i} \in D_{\hat{k}}} \nu^{\hat{k}\hat{i}} \qquad \hat{k} \in \hat{K} \qquad (4.2h)$$

$$\hat{y}_{\hat{k}\hat{i}} g_e(\nu^{\hat{k}\hat{i}}/\hat{y}_{\hat{k}\hat{i}}) \leq 0 \qquad \hat{k} \in \hat{K}, \hat{i} \in D_{\hat{k}}, e \in \hat{E}_{\hat{k}} \qquad (4.2i)$$

$$\hat{y}_{\hat{k}\hat{i}} r_{ki}(\nu^{\hat{k}\hat{i}}/\hat{y}_{\hat{k}\hat{i}}) \leq 0 \qquad \hat{k} \in \hat{K}, \hat{i} \in D_{\hat{k}}, ki \in \hat{K}I_{\hat{k}\hat{i}} \qquad (4.2j)$$

$$\sum_{\hat{i} \in D_{\hat{k}}} \hat{y}_{\hat{k}\hat{i}} = 1 \qquad \hat{k} \in \hat{K} \qquad (4.2k)$$

$$x^{lo} \hat{y}_{\hat{k}\hat{i}} \leq \nu^{\hat{k}\hat{i}} \leq x^{up} \hat{y}_{\hat{k}\hat{i}} \qquad \hat{k} \in \hat{K}, \hat{i} \in D_{\hat{k}} \qquad (4.2l)$$

$$0 \leq \hat{y}_{\hat{k}\hat{i}} \leq 1 \qquad \hat{k} \in \hat{K}, \hat{i} \in D_{\hat{k}} \qquad (4.2m)$$

$$y_{ki} = \sum_{\substack{\hat{k} \in \hat{K}, \hat{i} \in D_{\hat{k}} \\ ki \in \hat{K}I_{\hat{k}\hat{i}}}} \hat{y}_{\hat{k}\hat{i}} \qquad k \in \tilde{K}, i \in D_k \qquad (4.2n)$$

The feasible region (SEP) is defined by constraints (4.2a) - (4.2n). (4.2a) - (4.2g) is the relaxed (HR) of the global constraints and disjunctions to which no basic steps where applied ($K\backslash\tilde{K}$ and $E\backslash\tilde{E}$). (4.2h) - (4.2m) is the relaxed (HR) of the global constraints and disjunctions in which basic steps were applied. In these constraints $\hat{k} \in \hat{K}$ are the

resulting disjunctions after applying basic steps (that we denote "key disjunctions"), and $\hat{i} \in D_{\hat{k}}$ their corresponding disjunctive terms. The set $\hat{E}_{\hat{k}}$ maps the intersection of global constraints $e \in E$ with the new disjunctions $\hat{k} \in \hat{K}$. Note that a global constraint, which corresponds to an improper disjunction, can be intersected with more than one disjunction. Intersecting a global constraint with multiple disjunctions might provide further tightening of the relaxation in some cases. The set $\hat{KI}_{\hat{k}\hat{i}}$ maps the original disjunctive terms $ki, k \in K, i \in D_k$ to the resulting disjunctive terms after the application of basic steps $\hat{k} \in \hat{K}, \hat{i} \in D_{\hat{k}}$. Finally, constraint (4.2n) relates the original binary variables y_{ki} to the resulting ones $\hat{y}_{\hat{k}\hat{i}}$ as described in Theorem 1.2.2.

Let (P-SEP) be the projection of (SEP) over the original space: (P-SEP)$= Proj_x$(SEP). The following propositions allows us to derive valid cutting planes for the (BM) reformulation:

Proposition 4.2.1 *(P-SEP) $\subseteq$ (F-BM)*

Proof. For GDP (P-HR) $\subseteq$ (F-BM)[8]. Also, after applying basic steps (P-SEP) $\subseteq$ (P-HR)[12,15] ■.

Proposition 4.2.2 *(SEP) and (P-SEP) are convex regions.*

Proof. 1) The original functions $g(x)$ and $r_{ki}(x)$ are convex. (SEP) contains either the original functions or the perspective function of the original functions. The perspective function is an operation that preserves convexity, therefore, (SEP) is convex. 2) (P-SEP) is convex, since it is the projection of a convex region, and projection preserves convexity ■.

Let $z = (x, y)$. In order to derive a separating hyperplane that cuts off a point z^{BM}, consider the two following separation problems:

$$\begin{aligned} & min\ \phi(z) = ||z - z^{BM}|| \\ s.t. \quad & (z, \nu, \hat{y}) \in (SEP) \end{aligned} \tag{4.3}$$

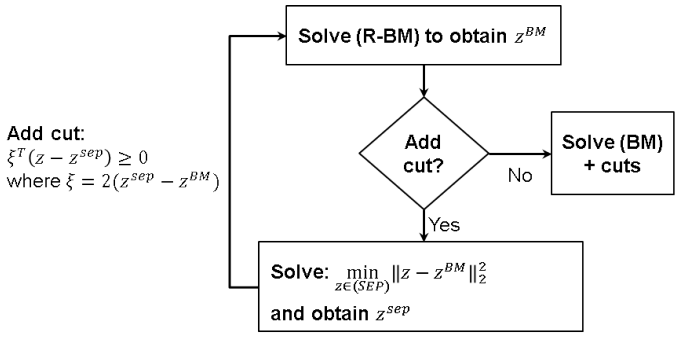

Figure 4.1: Outline of Algorithm 2

and,

$$\begin{aligned} & min\ \phi(z) = ||z - z^{BM}|| \\ s.t. \quad & z \in (P\text{-}SEP) \end{aligned} \tag{4.4}$$

(SEP) is convex, and (P-SEP) $\subseteq$ (F-BM). Therefore, following propositions hold true[13]:

Proposition 4.2.3 *Let $(z^{sep}, \nu^{sep}, \hat{y}^{sep})$ be an optimal solution of (4.3). Then z^{sep} is an optimal solution of (4.4).*

Proposition 4.2.4 *Let z^{BM} be the optimal solution of the continuous relaxation of the (BM) reformulation of (4.1), and z^{sep} an optimal solution of (4.3). If $z^{BM} \notin (P\text{-}SEP)$, then $\exists\ \xi$ such that $\xi^T(z - z^{sep}) \geq 0$ is a valid linear inequality in z^{sep} that cuts off z^{BM}, and such ξ is a subgradient of $\phi(x)$ at z^{sep}.*

Proposition 4.2.5 *Let (SEP) $\subset S$, where S is a convex set. If $\phi : S \to \mathbb{R}$ is differentiable over its entire domain, then the collection of subgradients of ϕ at z^{sep} is the singleton set $\partial\phi \equiv \{\xi^{sep} | \xi^{sep} = \nabla\phi(z^{sep})\}$.*

Proposition 4.2.6 *Let (SEP) $\subset S$, where S is a convex set. If $\phi : S \to \mathbb{R}$ is defined as $\phi(z) = ||z - z^{BM}||_2^2$, then the collection of subgradients of ϕ at z^{sep} is the singleton set $\partial\phi \equiv \{\xi^{sep} | \xi^{sep} = 2(z^{sep} - z^{BM})\}$.*

With these propositions, it is possible to derive cuts for (BM) using the (HR) reformulation after basic steps. The outline of the algorithm is shown in Figure 4.1. Figure 4.2 illustrates the algorithm in a simple example with three disjunctions, each one with two terms. There

are several decisions and heuristics in the algorithm that have to be considered:

a) Number of cuts: A predetermined number of cuts can be established. The optimal value of the objective function in (4.3) can also be used as an indicator of the performance of the cuts.

b) The number of new resulting disjunctions, or "key disjunctions" has to be decided at each iteration.

c) The number of basic steps to apply in each "key disjunction" at each iteration.

d) Heuristics to select which disjunctions to intersect in each "key disjunction" at each iteration.

e) Selection of which global constraints to intersect with each disjunction. Includes generation of "redundant" constraints (i.e. intersecting a global constraint with more than one disjunction) at each iteration.

f) Selection of the norm in (4.3). In particular, the norm-2 squared is convenient for nonlinear convex GDP, but norm-1 or the infinity-norm might be computationally more convenient for linear GDP, since (4.3) then becomes linear.

In particular for b) - e), only a few heuristics have been developed to select intersection of disjunctions[12,26,15,30]. Important improvements in the algorithm could be achieved by using better heuristics.

It is important to note that in the proposed algorithm, the derived cutting planes are stronger than the ones proposed by Vecchietti et al[28]. These authors use the (HR) formulation to derive cuts for the (BM). In this chapter, the cuts are generated using the (HR) after the application of basic steps, so the separation problem (SEP) has a tighter feasible region.

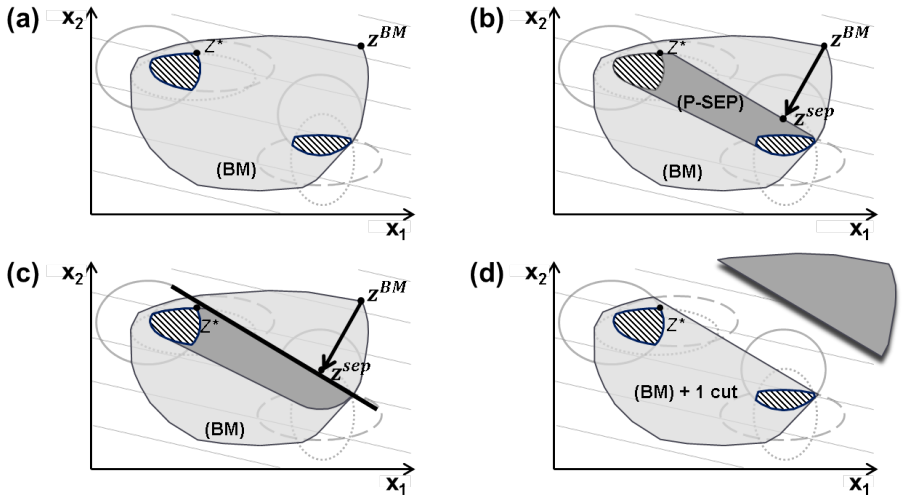

Figure 4.2: (a) Solution of the relaxation of the (BM) formulation. (b) Solution of (4.3). (c) Cutting plane $\xi(z - z^{sep}) \geq 0$. (d) Addition of cutting plane to (BM) formulation

4.3 Illustration of algorithm

To illustrate the algorithm, consider the simple convex GDP analytical example (4.5):

$$
\begin{aligned}
& min\ l \\
s.t.\quad & l \geq x_1 \\
& l \geq x_2 \\
& l \geq x_3 \\
& l \geq x_4 \\
& \begin{bmatrix} Y_{11} \\ x_1^2/50 - x_2 + 2 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ -x_1 + x_2^2/80 + 4 \leq 0 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ x_1^2/60 - x_3 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ -x_1 + x_3^2/60 + 5 \leq 0 \end{bmatrix} \\
& \begin{bmatrix} Y_{31} \\ x_1^2/60 - x_4 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{32} \\ -x_1 + x_4^2/70 + 6 \leq 0 \end{bmatrix} \\
& \begin{bmatrix} Y_{41} \\ x_2^2/60 - x_3 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{42} \\ -x_2 + x_3^2/90 + 4 \leq 0 \end{bmatrix} \\
& \begin{bmatrix} Y_{51} \\ x_2^2/70 - x_4 + 9 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{52} \\ -x_2 + x_4^2/50 + 7 \leq 0 \end{bmatrix} \\
& \begin{bmatrix} Y_{61} \\ x_3^2/90 - x_4 + 6 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{62} \\ -x_3 + x_4^2/80 + 3 \leq 0 \end{bmatrix} \\
& Y_{i1} \veebar Y_{i2} \qquad i = 1, ..., 6 \\
& 3 \leq x_1 \leq 100; 0 \leq x_2 \leq 100; 3 \leq x_3 \leq 100; 0 \leq x_4 \leq 100 \\
& Y_{i1}, Y_{i2} \in \{True, False\} \qquad i = 1, ..., 6 \\
& x_1, x_2, x_3, x_4 \in \mathbb{R}
\end{aligned}
\tag{4.5}
$$

The optimal solution of the continuous relaxation of the (BM) of this problem is **3**, and of the (HR) is **3.94**. The optimal solution of this problem is **7**. Note that if the (BM) of this

problem is solved with SBB from GAMS[21] it takes 18 nodes to find and prove the optimal solution. Consider the feasible region described in (4.6):

$$
\begin{gathered}
\begin{bmatrix} Y_{11} \\ l \geq x_1 \\ x_1^2/50 - x_2 + 2 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ l \geq x_1 \\ -x_1 + x_2^2/80 + 4 \leq 0 \end{bmatrix} \\
\begin{bmatrix} Y_{21} \\ x_1^2/60 - x_3 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ -x_1 + x_3^2/60 + 5 \leq 0 \end{bmatrix} \\
\begin{bmatrix} Y_{31} \\ x_1^2/60 - x_4 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{32} \\ -x_1 + x_4^2/70 + 6 \leq 0 \end{bmatrix} \\
\begin{bmatrix} Y_{41} \\ x_2^2/60 - x_3 \leq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{42} \\ -x_2 + x_3^2/90 + 4 \leq 0 \end{bmatrix} \\
\begin{bmatrix} \hat{Y}_{11} \\ l \geq x_2 \\ l \geq x_3 \\ l \geq x_4 \\ x_2^2/70 - x_4 + 9 \leq 0 \\ x_3^2/90 - x_4 + 6 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{12} \\ l \geq x_2 \\ l \geq x_3 \\ l \geq x_4 \\ x_2^2/70 - x_4 + 9 \leq 0 \\ -x_3 + x_4^2/80 + 3 \leq 0 \end{bmatrix} \\
\vee \begin{bmatrix} \hat{Y}_{13} \\ l \geq x_2 \\ l \geq x_3 \\ l \geq x_4 \\ -x_2 + x_4^2/50 + 7 \leq 0 \\ x_3^2/90 - x_4 + 6 \leq 0 \end{bmatrix} \vee \begin{bmatrix} \hat{Y}_{14} \\ l \geq x_2 \\ l \geq x_3 \\ l \geq x_4 \\ -x_2 + x_4^2/50 + 7 \leq 0 \\ -x_3 + x_4^2/80 + 3 \leq 0 \end{bmatrix} \\
Y_{i1} \veebar Y_{i2} i = 1, ..., 4 \\
\hat{Y}_{12} \veebar \hat{Y}_{12} \veebar \hat{Y}_{13} \veebar \hat{Y}_{14} \\
3 \leq x_1 \leq 100; 0 \leq x_2 \leq 100; 3 \leq x_3 \leq 100; 0 \leq x_4 \leq 100 \\
Y_{i1}, Y_{i2} \in \{True, False\} i = 1, ..., 4 \\
\hat{Y}_{12}, \hat{Y}_{12}, \hat{Y}_{13}, \hat{Y}_{14} \in \{True, False\} \\
x_1, x_2, x_3, x_4 \in \mathbb{R}
\end{gathered}
\tag{4.6}
$$

Note that (4.6) represents the feasible region of (4.5) after the following basic steps: the first global constraint is intersected with the first disjunction, a basic step is performed with disjunctions 5 and 6; the remaining global constraints are intersected with the disjunction that resulted from the basic step between disjunction 5 and 6.

Also, the constraints that relate the original variables y to the new variables $\hat{y}$, described in Theorem 1.2.2, are presented in (4.7).

$$\begin{aligned} y_{51} &= \hat{y}_{11} + \hat{y}_{12} \\ y_{52} &= \hat{y}_{13} + \hat{y}_{14} \\ y_{61} &= \hat{y}_{11} + \hat{y}_{13} \\ y_{62} &= \hat{y}_{12} + \hat{y}_{14} \end{aligned} \tag{4.7}$$

Let $z = [l, x_1, x_2, x_3, x_3, y_{11}, y_{12}, y_{21}, y_{22}, y_{31}, y_{32}, y_{41}, y_{42}, y_{51}, y_{52}, y_{61}, y_{62}]^T$

Step 1.

The relaxation of the (BM) reformulation of (4.5) is solved. The following solution is obtained:

$z^{BM} = [3, 3, 0, 3, 0, 0.001, 0.999, 0.002, 0.998, 0.003, 0.997, 0.004, 0.996, 0.007, 0.993, 0, 1]^T$

Step 2.

The separation problem is solved by minimizing $\Phi(z) = ||z - z^{BM}||_2^2$:

$\Phi(z) = (l-3)^2 + (x_1-3)^2 + (x_2-0)^2 + (x_3-3)^2 + (x_4-0)^2 + (y_{11}-0.001)^2 + (y_{21} - 0.999)^2 + (y_{21} - 0.002)^2 + (y_{22} - 0.998)^2 + (y_{31} - 0.003)^2 + (y_{32} - 0.997)^2 + (y_{41} - 0.004)^2 + (y_{42} - 0.996)^2 + (y_{51} - 0.007)^2 + (y_{52} - 0.993)^2 + (y_{61} - 0)^2 + (y_{62} - 1)^2$

Subject to the continuous relaxation of the (HR) of (4.6) and to (4.7). The solution of this separation problem is: $z^{SEP} = [7.5, 3, 5.2, 3, 2.3, 1, 0, 1, 0, 1, 0, 1, 0, 0.3, 0.7, 0.3, 0.7]^T$.

Cut generation.

Since $\Phi(z) = ||z - z^{BM}||_2^2$, then $\xi^{sep} = 2(z^{sep} - z^{BM})\}$:

$\xi^{sep} = [9, 0, 10.4, 0, 4.7, 2, -2, 2, -2, 2, -2, 2, -2, 0.5, -0.5, 0.5, -0.5]$.

The following cut is then added to (BM):

$$9(l - 7.5) + 0(x_1 - 3) + 10.4(x_2 - 5.2) + 0(x_3 - 3) + 4.7(x_4 - 2.3) + 2(y_{11} - 1) - 2y_{21} + 2(y_{21} - 1) - 2y_{22} + 2(y_{31} - 1) - 2y_{32} + 2(y_{41} - 1) - 2y_{42} + 0.5(y_{51} - 0.3) - 0.5(y_{52} - 0.7) + 0.5(y_{61} - 0.3) - 0.5(y_{62} - 0.7) \geq 0$$

The continuous relaxation of the (BM) with this additional cut is **6.01**.

Iteration.

This procedure can be repeated with the solution of the (BM) relaxation after adding the cut. With a second cut the continuous relaxation becomes **6.79**, and with a third iteration it becomes **6.96**. Solving the MINLP by doing a (BM) reformulation and adding the three cuts just described, takes SBB **5** nodes (in contrast to solving the (BM) reformulation without the cuts, which requires **18** nodes).

4.4 Results

In this section we present the computational results of applying the algorithm described in Section 4.2 to different problems. The GDP formulation of these problems is presented in Appendix B . The algorithm applies basic steps in a "key disjunction" following the heuristics presented in chapter 3, as long as the "key disjunction" contains less than 10 disjunctive terms (before the basic step). Four different strategies where tested for the number of "key disjunctions". In strategy $K0$ no proper basic steps where applied. In $K1$ there is only one "key disjunction". In $K5$ five "key disjunctions" were generated. In KK there are as many key disjunctions as the instance allows. Two strategies where tested for the basic steps with the global constraints. Strategy $I1$, where all global constraints are intersected once with a disjunction (as long as they share variables in common, "key disjunctions" where preferred over regular ones). In strategy $I2$ all global constraints are intersected with all of the disjunctions with which they share variables in common. For this chapter, the selection of disjunctions and basic steps does not change after each iteration,

Table 4.1: Number of constraints and variables for the test problems.

			(BM)		(HR)	
Instance	Solution	Binary	Cont. vars.	# constraints	Cont. vars.	# constraints
Clay42	8,469	32	22	93	134	349
Clay43	9,746	36	22	109	142	381
Clay44	7,923	40	22	125	150	413
Clay45	8,781	0	66	141	202	445
Clay52	11,472	50	32	138	212	548
Clay53	20,799	55	32	158	222	569
Clay54	10,876	60	32	178	232	608
Clay55	9,223	65	32	198	242	668
Flay04	20	24	20	45	212	381
Flay05a	68	40	24	68	349	633
Flay05b	63	40	24	68	349	633
Flay05c	57.5	40	24	68	349	633
Prc1-21	17.2	21	21	56	84	137
Prc1-31	12.2	41	36	102	159	255
Prc1-36	12.1	46	36	112	174	280
Prc1-48	12.1	61	45	149	228	371
Prc2-21	17	42	27	125	169	341
Prc2-31	12.3	82	46	235	336	675
Prc2-36	12.1	92	46	260	376	760

but in general it is possible to do so. Finally, the algorithm was tested by generating 1, 2, 3, 5, 7 and 10 cuts.

Nineteen nonlinear convex instances were solved for the problems presented in the Appendix. The instances were generated by defining problem size and structure, and randomly generating the parameters of the problems. The ϵ has a value of 10^{-4}, and the Big M parameter was determined using by estimating a basic feasible solution of the problem. The problem size and solution of this instances is presented in Table 4.1. The instances are solved using SBB. The algorithm and models were implemented in GAMS 24.2[21] and solved in an Intel(R) Core(TM) i7 CPU 2.93 GHz and 4 GB of RAM.

This section first presents three plots on the general performance of the algorithm. It then provides in-depth tables with the statistics and computational results of the different problems and strategies.

Figure 4.3 shows the percentage of problems solved vs. time for the (BM), (HR) and

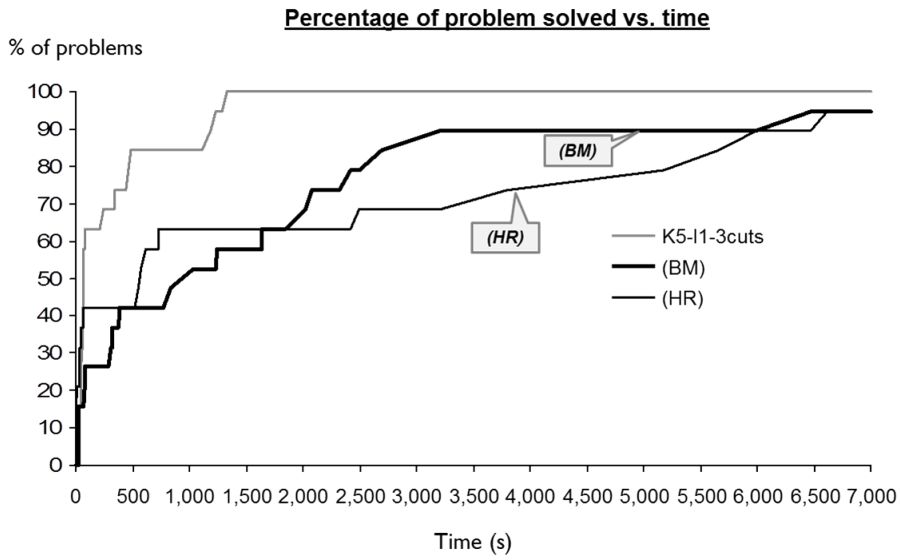

Figure 4.3: Percentage of problems solved vs. time

the algorithm, using the strategy $K5 - I1$, and using 3 cuts. The time for the algorithm includes the time to generate the cuts and time to solve the MINLP. The figure shows that the algorithm, using strategy $K5 - I1$ and 3 cuts, performs considerably better than the direct (BM) and (HR) reformulations.

Figure 4.4 shows the relaxation gap for the (BM), (HR) and the algorithm for the different strategies and different numbers of cuts. It is important to note that after one single cut, the formulation presents a stronger relaxation than the (HR). This is an important result, considering that the new formulation has the same number of variables than the (BM), and just an additional constraint. It can also be seen that after the first cut, there is a small improvement in the relaxation. In terms of strategies, intersecting global constraints with every disjunction with which they share variables provides the best continuous relaxation. Note that this strategy involves generating redundant constraints. With this strategy $I2$, and using the proposed heuristics, the number and strategy for proper basic steps has small impact in the continuous relaxation(strategies $K0 - I2$,$K1 - I2$,$K5 - I2$,$KK - I2$). However, when the global constraints are intersected only once, the use of proper basic steps helps to improve the continuous relaxation (e.g. strategy $K0 - I1$ has a weaker

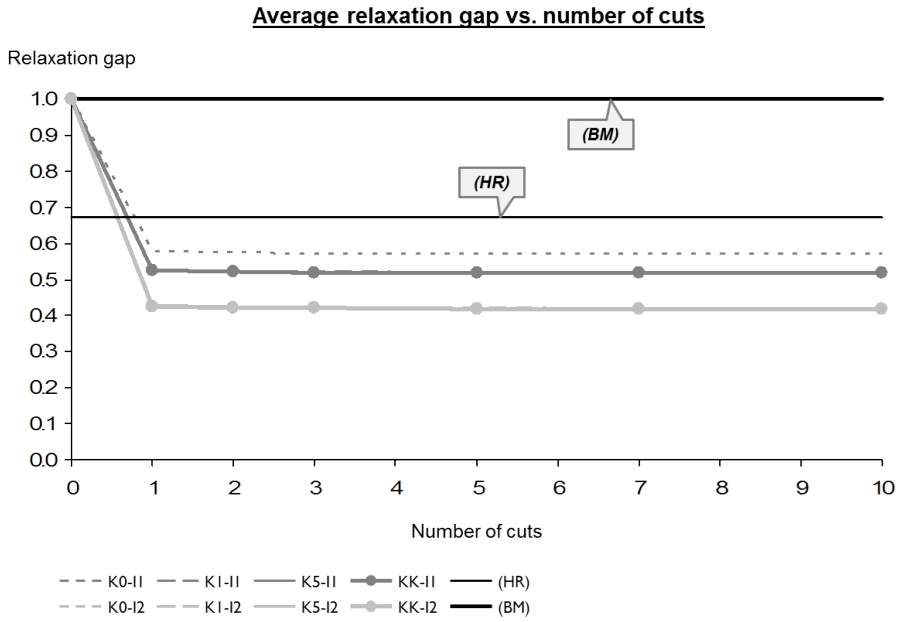

Figure 4.4: Average relaxation gap vs. number of cuts for different strategies in the algorithm

continuous relaxation than strategy $K1-I1$). It is important to note that different heuristics in the application of basic steps will impact this behavior.

Figure 4.5 shows the accumulated solution time to solve all instances, using different strategies, with different number of cuts. The maximum time allowed was 7,200 seconds. The figure shows that the solution time decreases with the first 3 cuts or so, but it starts increasing after that. This behaviour is expected, since the first cuts reduce the solution time of the MINLP considerably. However, as the number of cuts increases, the solution time of the MINLP does not improve much, and the time to generate the cuts becomes relatively more expensive. It is interesting to note that, even though the relaxation after the first cut does not change much, generating around 3 cuts seems to be the best strategy for the tested problems. As expected, in the strategies that generate large problem sizes ($K5-I2$ and $KK-I2$) the time to generate cuts is expensive, so the solution times after 3 cuts increases considerably.

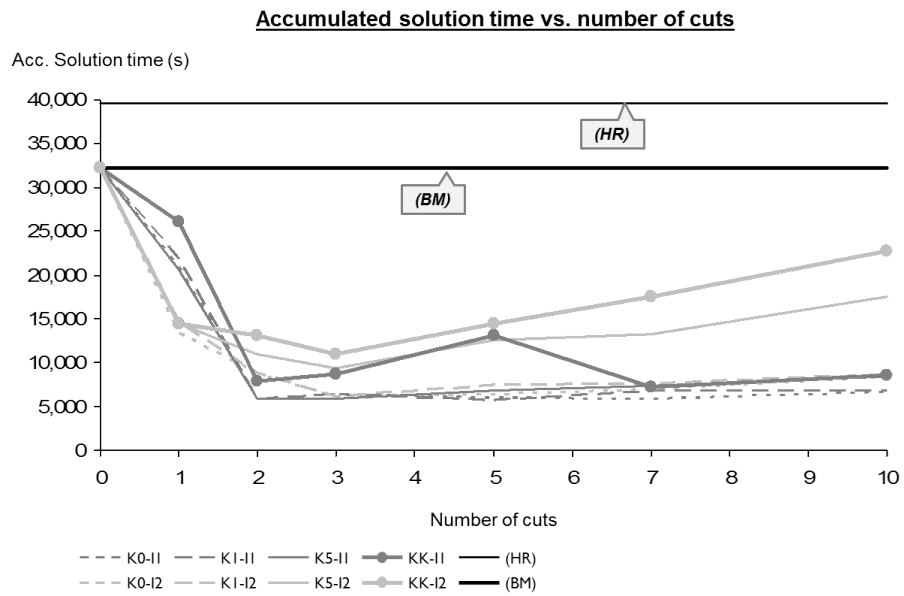

Figure 4.5: Accumulated solution time vs. number of cuts for different strategies in the algorithm

Table 4.2 summarizes the performance of the (BM), (HR), and of the algorithm for strategy $K5 - I1$ with three cuts. It is easy to see that the relaxation of the MINLP after adding the cutting planes is stronger than the (HR) in most cases. The algorithm has the strongest continuous relaxation in 13 of the 19 instances, while the (HR) is the strongest in 2. Note that in the 2 instances in which (HR) is stronger, the value relaxation of the problem after the cutting planes is very close to that of the (HR). On the C-lay problems, the algorithm provides a value of the continuous relaxation much stronger than either the (BM) or the (HR). In terms of solution times, the algorithm is the fastest in 14 of the 19 instances, the (HR) in 4, and the (BM) in 1. Except in Proc-1-21, in all other instances in which the algorithm is not the fastest it is the second fastest. It is also important to notice that the algorithm solves all of the problems in less than $1,325$ seconds, while the (BM) and (HR) cannot solve all of the problems within the two-hour limit.

Table 4.3 presents the separation problem size for the different strategies, compared to the (BM) reformulation. It is easy to see that, in general, strategy $I2$ generates a much larger

Table 4.2: Performance of the algorithm for strategy $K5 - I1$ and 3 cuts.

		Continuous relaxation			Solution time (s)				
							Algorithm		
Instance	Solution	(BM)	(HR)	Algo.	(BM)	(HR)	cut-gen	MINLP	Total
Clay42	8,469	0	0	2,877	24	34	8.3	9.6	**18**
Clay43	9,746	0	0	2,774	74	31	16.8	7.2	**24**
Clay44	7,923	0	0	2,788	312	548	25.6	36.7	**62**
Clay45	8,781	0	0	3,384	318	727	35.6	39.9	**76**
Clay52	11,472	0	0	5,187	383	574	20.8	48.3	**69**
Clay53	20,799	0	0	4,826	2,696	2,498	27.4	27	**54**
Clay54	10,876	0	0	4,417	2,075	5,175	34.2	306	**340**
Clay55	9,223	0	0	3,680	6,477	$>$ 7,200	59.5	429	**488**
Flay04	20	11.3	11.9	15.2	**27**	48	22.6	21.8	44
Flay05a	68	39.8	39.8	51.7	2,019	6,617	70.1	1,255	**1,325**
Flay05b	63	35.3	35.3	45.3	1,247	5,660	53.7	1,138	**1,192**
Flay05c	57.5	31.2	31.6	42.5	1,642	6,000	73.4	1,156	**1,229**
Prc1-21	17.2	0	15.7	15.6	24	**4.7**	20.9	20.3	41
Prc1-31	12.2	0	12.2	12.2	840	**1**	31.6	1.3	33
Prc1-36	12.1	0	12.1	12.1	2,414	**5**	62.2	1.1	63
Prc1-48	12.1	0	12.1	12	$>$ 7,200	**10**	44.9	13.4	58
Prc2-21	17	0	0.3	0.3	76	71	3.9	48.3	**52**
Prc2-31	12.3	0	0.2	0.2	1,030	616	5.4	240.4	**246**
Prc2-36	12.1	0	0.1	0.2	2,770	3,791	5.2	369.1	**374**

Table 4.3: Ratios of problem size of (SEP) compared to (BM).

	Ratio of number of constraints								Ratio of number of variables							
	Strategy $I1$				Strategy $I2$				Strategy $I1$				Strategy $I2$			
Instance	$K0$	$K1$	$K5$	KK	$K0$	$K1$	$K5$	KK	$K0$	$K1$	$K5$	KK	$K0$	$K1$	$K5$	KK
Clay42	6.1	12	18	18	18	23	34	34	12	23	33	33	29	37	55	55
Clay43	5.5	11	17	17	16	21	34	34	13	23	35	35	32	40	64	64
Clay44	5.1	9.9	16	16	16	19	36	36	14	24	39	39	35	43	76	76
Clay45	4.8	9	17	17	15	18	39	39	4.7	8.2	14	14	12	15	31	31
Clay52	6.7	13	18	19	25	30	46	49	14	24	32	34	40	49	73	78
Clay53	6	12	16	18	23	28	44	51	14	25	35	38	43	52	83	95
Clay54	5.6	11	15	17	22	26	44	53	15	25	37	42	46	55	92	110
Clay55	5.4	9.9	15	17	21	25	44	55	16	26	39	46	49	58	101	125
Flay04	12	21	33	33	17	26	43	43	15	25	40	40	18	27	46	46
Flay05a	12	18	33	33	18	24	48	48	20	28	52	52	25	32	63	63
Flay05b	12	18	33	33	18	24	48	48	20	28	52	52	25	32	63	63
Flay05c	12	18	33	33	18	24	48	48	20	28	52	52	25	32	63	63
Prc1-21	5.4	10	13	13	12	15	19	19	11	20	25	25	22	29	37	37
Prc1-31	5.7	9.6	17	17	20	23	39	39	12	22	35	35	42	49	88	88
Prc1-36	5.6	9.1	19	19	20	22	48	48	13	23	43	43	47	54	119	119
Prc1-48	5.6	9	13	18	25	27	36	51	14	23	33	42	62	69	94	135
Prc2-21	4.3	4.6	10	11	13	13	19	19	12	14	27	29	35	35	53	55
Prc2-31	3.8	4.1	6.5	10	20	20	23	28	13	14	21	30	66	66	81	103
Prc2-36	3.8	4.1	6.2	9.4	21	21	23	27	14	15	22	31	74	74	89	112

problems than strategy $I1$. The difference in size lies in the generation of the redundant global constraints that are intersected with every single disjunction. In most cases, $K1$ is twice the size of $K0$, while $k5$ is thrice its size. It is important to note that in some instances strategy $K5$ and KK provide the same separation problem. This happens when strategy $K5$ applies all of the possible basic steps.

4.5 Conclusions

In this chapter, we have proposed a cutting plane algorithm that improves the relaxation of the (BM) reformulation of convex GDP formulation. The cutting planes for the (BM) are derived through a separation problem. The separation problem minimizes the distance between the optimal solution of the continuous relaxation of the (BM), and a point that lies within a tighter continuous region. The tighter continuous region is still valid for

the original GDP. We have proposed the use of basic steps in order to obtain the tighter region of the separation problem. This region is obtained by performing basic steps on the original GDP, and then applying the (HR) reformulation. The continuous relaxation of this region is as tight, and generally much tighter, that the continuous relaxation of the (BM). We have presented the results of applying this algorithm to several test problems, using different suggested strategies. The algorithm improves the relaxation of the (BM) in all cases. Also, the algorithm solves the test problems faster than the (HR) and (BM) direct reformulations in most cases.

Chapter 5

Lagrangean relaxation of the HR of linear GDP problems and its use in the disjunctive branch and bound

5.1 Introduction

Lagrangean relaxation of an optimization program is a widely-used and powerful tool for solving constrained optimization problems. The review work by Guignard[31] discusses how Lagrangean relaxation can be used in different solution methods and applications. Fisher[32] provides a theoretical background for Lagrangean relaxation of mixed-integer linear programs (MILP). The general idea in the Lagrangean relaxation is to "dualize" some of the constraints in the optimization problem (i.e. remove some constraints from the feasible region of the problem, and penalize the violation of such constraints in the objective function). This approach is particularly useful in problems with complicating constraints. Some of these problems appear in planning[33], scheduling[34], facility location[35], and stochastic programming problems[36]. In this type of problem, a Lagrangean relaxation is simpler to solve than the original problems.

A particular method that uses Lagrangean relaxation to solve MILPs is the Lagrangean relaxation based branch and bound[37]. This method follows the same general idea as the LP based branch and bound, but it solves a Lagrangean relaxation at every node instead of an LP relaxation. This method can be useful in problems in which, by dualizing the complicating constraints, the Lagrangean relaxation is simpler to solve than the LP relaxation. One of the main difficulties in automating this strategy, or any other method that uses Lagrangean relaxation, is identifying the complicating constraints, which can be non trivial and problem specific. Typically, the modeller needs to identify the problem structure and select the constraints to dualize, or needs to modify the model to allow such a structure[31].

In this chapter, we first present a Lagrangean relaxation of the HR for linear GDP problems. The proposed Lagrangean relaxation is an MILP, and it has three important characteristics. First, the solution to the continuous relaxation of the proposed Lagrangean relaxation always yields 0-1 values for the binary variables of the HR. Second, it is easier to solve than the original problem (i.e. the HR). Furthermore, it is easier to solve than the continuous relaxation of the HR. Third, this relaxation can be applied to any linear GDP. This means that there is no need to specify which are the complicating constraints in different problems, so automating a method that uses this Lagrangean relaxation can be achieved. We use the proposed Lagrangean relaxation to improve the performance of the disjunctive branch and bound algorithm. In particular, we evaluate the Lagrangean relaxation at every node and use its solution as heuristic for finding feasible solutions to the problem. The continuous relaxation of the Lagrangean relaxation always provides 0-1 values to the binary variables, so the value of the 0-1 variables is fixed and a small LP is solved in search of feasible solutions.

This chapter is organized as follows. Section 5.2 presents a brief background on Lagrangean relaxation of MILPs and on linear GDP problems. Section 5.3 presents the proposed Lagrangean relaxation of the HR, including the formulation and main properties. The proposed Lagrangean relaxation is then incorporated into a disjunctive branch and bound, which is presented in Section 5.4. Section 5.5 demonstrates the performance of the proposed disjunctive branch and bound in an illustrative example. The performance of the disjunctive branch and bound with the Lagrangean relaxation is evaluated against

other versions of the disjunctive branch and bound with several instances. The results of these instances are presented in Section 5.6.

5.2 Background

5.2.1 Linear generalized disjunctive programming

This chapter is concerned with linear GDP problems. The general linear GDP formulation can be represented as follows:

$$
\begin{aligned}
& & & min\ c^T x & & \\
& s.t. & & Gx \leq g & & \\
& & & \bigvee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ A^{ki}x \leq a^{ki} \end{bmatrix} & & k \in K \\
& & & \bigvee_{i \in D_k} Y_{ki} & & k \in K \\
& & & \Omega(Y) = True & & \\
& & & x^{lo} \leq x \leq x^{up} & & \\
& & & x \in \mathbb{R}^n & & \\
& & & Y_{ki} \in \{True, False\} & & k \in K, i \in D_k
\end{aligned}
\tag{LGDP}
$$

The (HR) formulation of the linear GDP is as follows:

$$
\begin{aligned}
& min\ c^T x \\
s.t.\quad & Gx \leq g \\
& x = \sum_{i \in D_k} \nu^{ki} && k \in K \\
& A^{ki}\nu^{ki} \leq a^{ki} y_{ki} && k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ki} = 1 && k \in K \\
& Hy \geq h \\
& x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} && k \in K, i \in D_k \\
& x \in \mathbb{R}^n \\
& \nu^{ki} \in \mathbb{R}^n && k \in K, i \in D_k \\
& y_{ki} \in \{0,1\} && k \in K, i \in D_k
\end{aligned}
\tag{HR}
$$

Note that this reformulation for linear GDP problems is equivalent to the convex hull representation in disjunctive programming[38].

5.2.2 Lagrangean relaxation of mixed-integer linear programs

In this section we present a brief review of the Lagrangean relaxation of mixed-integer linear programs. In this chapter, we consider the complicating constraints to be equality constraints. We refer the reader to the work by Guignard[31] for a comprehensive review and for proofs of the Theorems and relations presented in this section. Throughout the chapter, for any given optimization problem (Q) we let $v(Q)$ denote its optimal value and $F(Q)$ its feasible region.

Without loss of generality, consider the following general mixed-integer linear program:

$$\begin{aligned} & min\ c^T x \\ & Ax = b \\ & Cx \leq d \\ & x \in X \end{aligned} \tag{P}$$

where X contains the integrality and sign restrictions on x (e.g. $X = \mathbb{R}_+^{n-q} \times \{0,1\}^q$). Suppose that $Ax = b$ are the complicating constraints (i.e. the problem becomes much simpler to solve without them). Let λ be a vector of weights, namely the Lagrange multipliers.

The Lagrangean relaxation of (P) is:

$$\begin{aligned} & min\ c^T x + \lambda(Ax - b) \\ & Cx \leq d \\ & x \in X \end{aligned} \tag{$LR1_\lambda$}$$

In ($LR1_\lambda$), the complicating constraints ($Ax = b$) have been "dualized" (i.e. the slacks of the complicating constraints have been added to the objective function, and the complicating constraints dropped from the formulation). Note that if these constraints are inequalities ($Ax \leq b$), then the corresponding Lagrange multipliers are non-negative.

It is easy to see that ($LR1_\lambda$) is a relaxation of (P), since $F(\text{P}) \subseteq F(LR1_\lambda)$. Therefore, $v(LR1_\lambda) \leq v(\text{P})$ in general. When $x \in F(\text{P})$ then $v(LR1_\lambda) = v(\text{P})$ (when the complicating constraints are equalities).

Property 5.2.1

1. If $x(\lambda)$ is an optimal solution of ($LR1_\lambda$) for some λ, then $c^T x(\lambda) + \lambda(Ax - b) \leq v$(P).

2. If in addition $x(\lambda)$ is feasible for (P), then $x(\lambda)$ is an optimal solution of (P), and $c^T x(\lambda) = v$(P)

Theorem 5.2.1 states that Lagrangean relaxation always provides a lower bound for the

MILP problem. The best possible lower bound that the Lagrangean relaxation provides can be obtained with the following optimization problem:

$$\max_{\lambda} v(LR1_{\lambda}) \tag{LD}$$

Problem (LD) is called the Lagrangean dual of (P) with respect to the complicating constraints $Ax = b$.

Let (RP) be the continuous relaxation of (P). In general, $v(RP) \leq v(\text{LD})$. In the particular case in which the Lagrangean dual has the integrality property (i.e. the extreme points of $\{x|Cx \leq d\}$ are in X), $v(RP) = v(\text{LD})$.

5.3 Lagrangean relaxation of the hull-reformulation

In this section we present a Lagrangean relaxation of the HR of any linear GDP. This relaxation is easier to solve than the continuous relaxation of the HR. The continuous relaxation of the proposed Lagrangean relaxation can be proved to always yield 0-1 values to the binary variables of the reformulation. We first present the Lagrangean relaxation for the case in which GDP does not involve logic relations, and we then discuss its extension to the case in which it does. We note that these properties can be extended to nonlinear convex GDP. This is achieved by using the theory for convex GDP by Ruiz and Grossmann[15], which extends part of the rich theory of disjunctive programming[38] to GDP problems with convex constraints.

5.3.1 Lagrangean relaxation of the HR without logic relations

For given Lagrange multipliers (λ_{kj}), the following Lagrangean relaxation can be applied to the hull-reformulation of any linear GDP:

$$
\begin{aligned}
& min\ c^T x + \sum_{k\in K}\sum_{j\in J_k} \lambda_{kj}(x_j - \sum_{i\in D_k} \nu^{ki}) \\
s.t.\quad & Gx \leq g \\
& A^{ki}\nu^{ki} \leq a^{ki} y_{ki} && k \in K, i \in D_k \\
& \sum_{i\in D_k} y_{ki} = 1 && k \in K \\
& x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} && k \in K, i \in D_k \\
& x \in \mathbb{R}^n \\
& y_{ki} \in \{0,1\} && k \in K, i \in D_k
\end{aligned}
\tag{LHR_λ}
$$

where J_k is the set of variables that appear in disjunction k.

Property 5.3.1 *The Lagrangean relaxation (LHR_λ) can be applied to any linear GDP.*

Property 5.3.1 is trivial, since the decomposition is applied to the MILP reformulation of the general form of linear GDP.

Note that in problem (LHR_λ) the variables x_j and ν^{ki} do not appear together in any constraint. Furthermore, variables ν^{ki} and $\nu^{k'i'}$ for $k' \neq k; i \in D_k; i' \in D_{k'}$ do not appear together in any constraint either. Therefore, (LHR_λ) can be decomposed into $|K| + 1$ simpler problems.

The first problem, which involves only the continuous variables and global constraints, is as follows:

$$
\begin{aligned}
& min\ c^T x + \sum_{k\in K}\sum_{j\in J_k} \lambda_{kj} x_j \\
s.t.\quad & Gx \leq g \\
& x \in \mathbb{R}^n
\end{aligned}
\tag{LHR_0}
$$

The remaining $k \in K$ problems, each one containing the ν^{ki} variables corresponding to

disjunction k, are as follows:

$$
\begin{aligned}
& min \; -\sum_{j\in J_k}\lambda_{kj}(\sum_{i\in D_k}\nu^{ki}) & & \\
s.t. \quad & A^{ki}\nu^{ki} \le a^{ki}y_{ki} & & k\in K, i\in D_k \\
& \sum_{i\in D_k} y_{ki} = 1 & & k\in K \\
& x^{lo}y_{ki} \le \nu^{ki} \le x^{up}y_{ki} & & k\in K, i\in D_k \\
& y_{ki}\in\{0,1\} & & k\in K, i\in D_k
\end{aligned}
\qquad (LHR_k)
$$

Property 5.3.2 *The optimal solution of (LHR_λ) can be obtained by the summation of the optimal values of $|K|+1$ problems: $v(LHR_\lambda) = v(LHR_0) + \sum_{k\in K} v(LHR_k)$.*

Property 5.3.2 indicates that it is possible to solve $|K|+1$ smaller MILPs to solve (LHR_λ). Furthermore, (LHR_λ) and each of these subproblems can be solved as an LP, as is shown in Property 5.3.3.

Property 5.3.3 *Let $(\hat{x},\hat{y},\hat{\nu})$ be a vertex of the continuous relaxation of (LHR_λ). Then, for every $k\in K$ there exists an $i\in D_k$ such that $\hat{y}_{ki}=1$, and $\hat{y}_{ki'}=0, \forall i'\in D_k, i'\neq i$.*

Proof. The proof is presented by analyzing the decomposed problem. Problem (LHR_0) does not involve any binary variable or disaggregated as decision variable. In problems (LHR_k), only the binary variables y_{ki} and the disaggregated variables ν_{ki}, that correspond to the disjunction ($k\in D_k$) are optimization variables. From Corollary 2.1.2 of Balas[38], for any vertex $(\bar{\nu},\bar{y})$ of the feasible region of the continuous relaxation of (LHR_k) there exists an $i\in D_k$ such that $\bar{y}_{ki}=1$, and $\bar{y}_{ki'}=0, \forall i'\in D_k, i'\neq i$. Therefore, for every $k\in K$ the vertices of their corresponding problem have $\{0,1\}$ values for y_{ki}. Since there are no constraints linking the variables of the feasible region of each of the $|K|+1$ problems, all of the vertices of the feasible region of the continuous relaxation of (LHR_λ) have $\{0,1\}$ values for y_{ki}. ■

Note that Property 5.3.3 indicates that every solution of (LHR_λ) yields 0-1 values for the binary variables. One of the main advantages of this property, together with 5.3.2, is that (LHR_λ) can be solved by solving $|K|+1$ small LPs. The solution of (LHR_λ) can be

used as a heuristic for finding "good" feasible solutions for (LGDP). On the downside, this property indicates that the best possible bound that can be obtained from (LHR_λ) is the same as the optimal value of the continuous relaxation of (HR)[31].

5.3.2 Including logic relations in (LHR_λ)

In cases in which there are logic relations between the Boolean variables of the GDP, (LHR_λ) needs to be modified to preserve Properties 5.3.3 and 5.3.2. This can be achieved by either introducing additional variables or including the propositional logic in the solution method[39]. Using the propositional logic in the solution method enables a large set of tools that combine logic-based methods with optimization. The work by Hooker[27] has established important theories and results in this field. However, the scope of this chapter is to establish the properties of a general Lagrangean relaxation of (HR), as well as the basis for using this relaxation as a primal heuristic. For this reason, we will present in this section a modified version of (LHR_λ) by introducing additional variables, and leave the integration of (LHR_λ) with logic-based methods for future work.

Consider the following problem, that is equivalent to (LGDP):

$$
\begin{aligned}
& min\ c^T x \\
s.t.\quad & Gx \leq g \\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ \bar{y}_{ki} = 1 \\ A^{ki} x \leq a^{ki} \end{bmatrix} && k \in K \\
& \veebar_{i \in D_k} Y_{ki} && k \in K \\
& \sum_{i \in D_k} \bar{y}_{ki} = 1 && k \in K \\
& H\bar{y} \geq h \\
& x^{lo} \leq x \leq x^{up} \\
& 0 \leq \bar{y}_{ki} \leq 1 && k \in k, i \in D_k \\
& x \in \mathbb{R}^n \\
& \bar{y}_{ki} \in \mathbb{R} && k \in K, i \in D_k \\
& Y_{ki} \in \{True, False\} && k \in K, i \in D_k
\end{aligned}
\tag{5.1}
$$

Note that in (5.1), a continuous variable ($0 \leq \bar{y}_{ki} \leq 1$) is introduced. The linear constraints that represent the logic relations are expressed for $\bar{y}$. It is easy to see that (5.1) is equivalent to (LGDP), in the sense that $Y_{ki} = True$ implies $\bar{y}_{ki} = 1$, and $Y_{ki} = False$ implies $\bar{y}_{ki} = 0$. Note that (5.1) also has the structure required: there are no logic relations between Boolean variables, and the global constraints and disjunctive terms involve only continuous variables. It is possible to perform the (HR) reformulation of (5.1). After few

algebraic substitutions, the HR of (5.1) is as follows:

$$
\begin{aligned}
& min\ c^T x \\
s.t.\quad & Gx \leq g \\
& y = \bar{y} \\
& x = \sum_{i \in D_k} \nu^{ki} && k \in K \\
& A^{ki}\nu^{ki} \leq a^{ki} y_{ki} && k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ki} = 1 && k \in K \\
& H\bar{y} \geq h \\
& x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} && k \in K, i \in D_k \\
& x \in \mathbb{R}^n \\
& \bar{y}_{ki} \in \mathbb{R} && k \in K, i \in D_k \\
& y_{ki} \in \{0,1\} && k \in K, i \in D_k
\end{aligned}
\tag{5.2}
$$

For Lagrange multipliers λ^1, λ^2, the following Lagrangean relaxation of (5.2) is obtained:

$$
\begin{aligned}
& min\ c^T x + \sum_{k \in K} \left(\sum_{j \in J_k} \lambda^1_{kj} (x_j - \sum_{i \in D_k} \nu^{ki}) + \sum_{i \in D_k} \lambda^2_{ki} (y_{ki} - \bar{y}_{ki}) \right) \\
s.t.\quad & Gx \leq g \\
& A^{ki}\nu^{ki} \leq a^{ki} y_{ki} && k \in K, i \in D_k \\
& \sum_{i \in D_k} y_{ki} = 1 && k \in K \\
& H\bar{y} \geq h \\
& x^{lo} y_{ki} \leq \nu^{ki} \leq x^{up} y_{ki} && k \in K, i \in D_k \\
& x \in \mathbb{R}^n \\
& \bar{y}_{ki} \in \mathbb{R} && k \in K, i \in D_k \\
& y_{ki} \in \{0,1\} && k \in K, i \in D_k
\end{aligned}
\tag{5.3}
$$

which decomposes into $|K| + 1$ subproblems (where $H\bar{y} \geq h$ is a global constraint and

appears in (LHR_0)).

In some cases, the solution of (LHR_λ) including the logic constraints for the original variables might still yield 0-1 values to the binary variables (e.g. it can still be solved as an LP). However, it might not be possible to decompose (LHR_λ) in smaller LPs (depending on the structure of the logic relations). Different methods could make use of problem (LHR_λ) to solve (LGDP). For example, it can be used as heuristic in the search of feasible solutions.

5.4 Lagrangean relaxation as a primal heuristic in the disjunctive branch and bound

The disjunctive branch and bound presented in Section 1.2.3 can be adapted to incorporate (LHR_λ) as a primal heuristic. Before presenting the algorithm, consider the LP subproblem (SP) that results when the value of the Boolean variables (or binary variables in the MILP reformulation) are fixed. Given $\hat{y}$ such that, for every $k \in K$ there is only one $i \in D_k$ for which $\hat{y}_{ki} = 1$ and $\hat{y}_{ki'} = 0, \forall i' \in D_k, i' \neq i$, the following (SP) is obtained:

$$\begin{aligned} & min\ c^T x \\ s.t. \quad & Gx \leq g \\ & A^{ki} x \leq a^{ki} \qquad \forall \hat{y}_{ki} = 1 \\ & x^{lo} \leq x \leq x^{up} \\ & x \in \mathbb{R}^n \end{aligned} \tag{SP}$$

In (SP), the constraints corresponding to active disjunctive terms $\hat{y}_{ki} = 1$ are enforced while constraints corresponding to non-active terms $\hat{y}_{ki'} = 0$ are removed.

The modified algorithm is as follows:

0. Initialize. Set $L = N_0$, $z^{up} = \infty$, $(x^*, y^*) = \emptyset$. Initialize $\lambda^0_{kj} = 0$.

1. Terminate. If $L = \emptyset$, then (x^*, y^*) is optimal and algorithm terminates.

2. Node selection. Choose a node $N_p \in L$, and delete it from L. Go either to 3a or to 3b.

3a. Bound. Solve the (R-HR) of GDP_p corresponding to N_p. If it is infeasible, go to step 1. Else, let z^p be its objective function and (x^p, y^p) its solution. Set λ^p_{kj} to be the Lagrange multipliers corresponding to the constraints $x_j = \sum_{i \in D_k} \nu^{ki};\ k \in K; j \in J_k$.

3b. Bound. Solve the (R-BM) of GDP_p corresponding to N_p. If it is infeasible, go to step 1. Else, let z^p be its objective function and (x^p, y^p) its solution.

4. Prune. If $z^p \geq z^{up}$, go to step 1.

If $y^p \in \mathbb{Z}^q$ let $z^{up} = z^p$ and $(x^*, y^*) = (x^p, y^p)$. Delete all nodes $N_r \in L$ in which $z^r \geq z^{up}$, and go to step 1. Else, go to step 5 or 6.

5. Primal heuristic (optional). Solve (LHR_λ) of GDP_p with λ^p_{kj}. Let $\hat{z}^p$ be its objective function and $\hat{y}$ the value of the integer variables. Let $z^p = max\{z^p, \hat{z}^p\}$.

Solve (SP) with fixed $\hat{y}$ and, if it is feasible, let $\bar{z}^p$ be its objective function and $(\bar{x}^p)$ its solution. If $\bar{z}^p \leq z^{up}$ let $z^{up} = \bar{z}^p$ and $(x^*, y^*) = (\bar{x}^p, \hat{y}^p)$. Delete all nodes $N_r \in L$ in which $z^r \geq z^{up}$. If $z^p = \bar{z}^p$, go to step 1. Else, fo to step 6.

6. Branch. Select a disjunction $k \in K$ such that $y_{ki} \notin \{0, 1\}$ for some $i \in D_k$. For every $i \in D_k$, construct the corresponding GDP (GDP^i_p) by setting the constraints corresponding to the disjunctive term i as global, and removing the Boolean variables and constraints corresponding to term $i^{'} \neq i; i^{'} \in D_k$. Add $|D_k|$ new nodes, corresponding to GDP^i_p, to L. For every one of the new nodes, let the corresponding Lagrange multipliers ($\lambda^{p,i}_{kj}$) be λ^p_{kj}. Go to step 1.

There are two main differences in the proposed disjunctive branch and bound with respect to the one presented in Section 1.2.3. The first in Step 3a and the second is the inclusion of the new Step 5.

The first difference is that when Step 3a is selected, the Lagrange multipliers are updated. Note that this implies that for the particular node in which the (R-HR) was solved, (LHR_λ) is in fact the Lagrangean dual of the HR of GDP_p. It is important (while not required) to perform Step 3a at the root node, so the Lagrange multipliers are initialized with the

Lagrangean dual of the HR of the original GDP. In subsequent nodes, it may or may not be useful to solve the larger HR reformulation in order to obtain better lower bounds and updated Lagrange multipliers.

The second difference is Step 5, which is optional. In step 5, (LHR_λ) is solved to obtain integer solutions. The discrete solution is fixed and a small LP is solved (which corresponds to the original GDP with fixed decisions). If the LP is feasible, it provides a valid upper bound and feasible solution. Note that if Step 3a was selected (LHR_λ) is in fact the Lagrangean dual. Otherwise, it is a Lagrangean relaxation that uses the Lagrange multipliers inherited from the parent node. Also note that if step 3b was selected, it is possible that $\hat{z}^p \geq z^p$ (e.g. the Lagrangean relaxation of the HR provides better bound than the continuous relaxation of the BM). For this reason, Step 5 sets $z^p = max\{z^p, \hat{z}^p\}$.

Same as the disjunctive branch and bound presented in Section 1.2.3, this algorithm converges in a finite number of iterations. In the worst case, evaluating every single resulting node of the search tree.

5.5 Illustrative example

In this section, we present a simple example and the application of the proposed disjunctive branch and bound to solve it.

Consider the following analytical example:

$$
\begin{aligned}
& min\ 7x_1 - 2x_2 \\
& s.t. \\
& \begin{bmatrix} Y_{11} \\ 0.9487x_1 + 0.3162x_2 \leq 11.3842 \\ -0.5145x_1 - 0.8575x_2 \leq -10.2899 \\ x_2 \leq 9 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ 0.9615x_1 - 0.2747x_2 \leq 5.7691 \\ -0.6247x_1 + 0.7809x_2 \leq 0.4685 \\ -0.7071x_1 - 0.7071x_2 \leq -4.2426 \end{bmatrix} \\
& \vee \begin{bmatrix} Y_{13} \\ 0.8944x_1 + 0.4472x_2 \leq 6.2610 \\ -0.9864x_1 + 0.1644x_2 \leq -0.3288 \\ 0.5547x_1 - 0.8321x_2 \leq -2.7735 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ 0.9806x_1 - 0.1961x_2 \leq 2.1573 \\ -0.6x_1 + 0.8x_2 \leq 4.8 \\ -0.5547x_1 - 0.8321x_2 \leq -4.9923 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ 0.9806x_1 + 0.1961x_2 \leq 7.2563 \\ -0.9487x_1 + 0.3162x_2 \leq -3.4785 \\ 0.3162x_1 - 0.9487x_2 \leq 0.3162 \end{bmatrix} \\
& \vee \begin{bmatrix} Y_{23} \\ x_1 \leq 10 \\ -0.3162x_1 + 0.9487x_2 \leq 6.3246 \\ -0.5547x_1 - 0.8321x_2 \leq -11.3714 \end{bmatrix} \\
& \begin{bmatrix} Y_{31} \\ x_1 \leq 6 \\ -0.3714x_1 - 0.9285x_2 \leq -3.1568 \\ -x_1 \leq -1 \\ x_2 \leq 6 \end{bmatrix} \vee \begin{bmatrix} Y_{32} \\ 0.7071x_1 + 0.7071x_2 \leq 10.6066 \\ -0.3162x_1 - 0.9487x_2 \leq -7.9057 \\ -0.4472x_1 + 0.8944x_2 \leq 6.7082 \end{bmatrix} \\
& \veebar_{i \in D_k} Y_{ki};\ k \in \{1,2,3\} \\
& 0 \leq x_1, x_2 \leq 10 \\
& Y_{ki} \in \{True, False\};\ k \in \{1,2,3\}, i \in D_k
\end{aligned}
\tag{5.4}
$$

The feasible region of problem (5.4) is shown in Figure 5.1. Figure 5.1a shows the feasible region of each disjunction, projected onto the space of the continuous variables. Figure

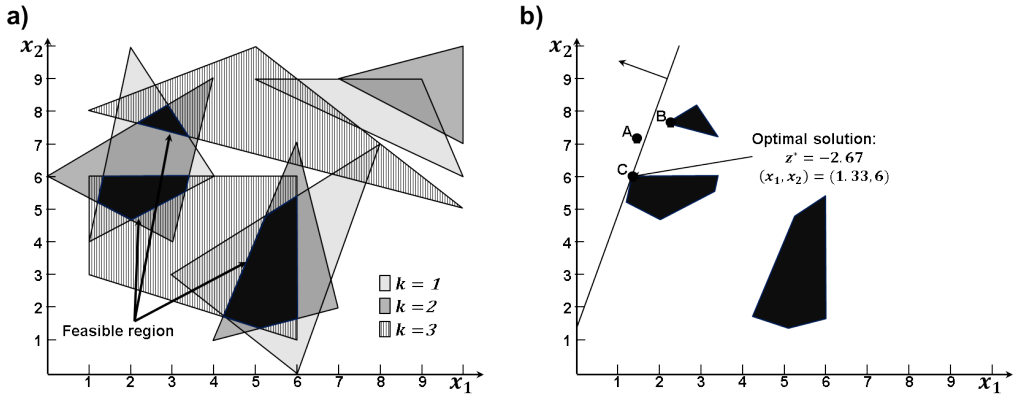

Figure 5.1: Illustration of the feasible region of: a) the feasible region projected onto x_1 and x_2, and b) the optimal solution.

5.1b shows the optimal solution to problem (5.4) in the projection onto the continuous variables.

The application of the proposed disjunctive branch and bound to (5.4) is as follows:

0. Initialize. Set $L = N_0$, $z^{up} = \infty$, $(x^*, y^*) = \emptyset$. Initialize $\lambda^0_{kj} = 0$.

1. Terminate. $L = N_0$, go to Step 2.

2. Node selection. Choose node N_0 and delete it from L. Go to 3a.

3a. Bound. The solution of the (R-HR) of GDP_0 (which corresponds to N_0) yields the following values:

Continuous variables and objective function $(x^0_1, x^0_2, z^0) = (1.5, 7.1, -3.6)$ (point A in figure 5.1b).

Binary variables $y^0_{13} = y^0_{21} = 1$, $y^0_{11} = y^0_{12} = y^0_{22} = y^0_{23} = 0$, and $y^0_{31} = 0.43, y^0_{32} = 0.57$

Lagrange multipliers $\lambda^0_{kj} = (-6.286, 1.048, -0.714, 0.952, 0, 0)$ (i.e. Lagrange multipliers corresponding to the constraints $x_j = \sum_{i \in D_k} \nu^{ki};\ k \in K; j \in J_k$ in (R-HR)).

4. Prune. Since $z^0 < z^{up}$ and $y^0 \notin \mathbb{Z}^q$, go to step 5.

5. Primal heuristic (optional). The solution of (LHR_λ) of GDP_0 with λ^0_{kj} yields: $\hat{z}^0 =$

-3.6, $\hat{y}^0_{13} = \hat{y}^0_{21} = \hat{y}^0_{32} = 1$.

The solution (SP) with fixed $\hat{y}^0$ is feasible and it yields the following values:

Continuous variables and objective function $(\bar{x}^0_1, \bar{x}^0_2, \bar{z}^0) = (2.15, 7.62, -0.154)$ (point B in Figure 5.1b).

Note that this solution indicates that even at the root node, it is possible to obtain a feasible solution. In this case, we update the best known solution: $z^{up} = -0.154$, $(x^*, y^*) = (\bar{x}^0, \hat{y}^0)$

6. Branch. Select $k = 3$ for branching. Since $|D_3| = 2$, two new nodes are created: $LDGP^1_0, LDGP^2_0$.

$LDGP_0^1$ is as follows:

$$
\begin{aligned}
& min\ 7x_1 - 2x_2 \\
s.t.\quad & x_1 \leq 6 \\
& -0.3714x_1 - 0.9285x_2 \leq -3.1568 \\
& -x_1 \leq -1 \\
& x_2 \leq 6 \\
& \begin{bmatrix} Y_{11} \\ 0.9487x_1 + 0.3162x_2 \leq 11.3842 \\ -0.5145x_1 - 0.8575x_2 \leq -10.2899 \\ x_2 \leq 9 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ 0.9615x_1 - 0.2747x_2 \leq 5.7691 \\ -0.6247x_1 + 0.7809x_2 \leq 0.4685 \\ -0.7071x_1 - 0.7071x_2 \leq -4.2426 \end{bmatrix} \\
& \vee \begin{bmatrix} Y_{13} \\ 0.8944x_1 + 0.4472x_2 \leq 6.2610 \\ -0.9864x_1 + 0.1644x_2 \leq -0.3288 \\ 0.5547x_1 - 0.8321x_2 \leq -2.7735 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ 0.9806x_1 - 0.1961x_2 \leq 2.1573 \\ -0.6x_1 + 0.8x_2 \leq 4.8 \\ -0.5547x_1 - 0.8321x_2 \leq -4.9923 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ 0.9806x_1 + 0.1961x_2 \leq 7.2563 \\ -0.9487x_1 + 0.3162x_2 \leq -3.4785 \\ 0.3162x_1 - 0.9487x_2 \leq 0.3162 \end{bmatrix} \\
& \vee \begin{bmatrix} Y_{23} \\ x_1 \leq 10 \\ -0.3162x_1 + 0.9487x_2 \leq 6.3246 \\ -0.5547x_1 - 0.8321x_2 \leq -11.3714 \end{bmatrix} \\
& \underset{i \in D_k}{\veebar} Y_{ki};\ k \in \{1,2,3\} \\
& 0 \leq x_1, x_2 \leq 10 \\
& Y_{ki} \in \{True, False\};\ k \in \{1,2,3\}, i \in D_k
\end{aligned}
\tag{5.5}
$$

Note that in (5.5), the constraints corresponding to $Y_{31} = True$ are included as global constraints, while constraints corresponding to $Y_{32} = True$ are removed from the formu-

lation. $LDGP_0^2$ can be constructed in the same manner, but including as global constraints the ones corresponding to $Y_{32} = True$ while removing the constraints corresponding to $Y_{31} = True$. For simplicity, $p = 1$ is assigned to $LDGP_0^1$, and $p = 2$ to $LDGP_0^2$. Two new nodes are added L: $L = \{N_1, N_2\}$ (corresponding to $p = 1, 2$). The Lagrange multipliers of the new nodes are initialized using λ_{kj}^0 from the parent node: $\lambda_{kj}^1 = \lambda_{kj}^2 = (-6.286, 1.048, -0.714, 0.952, 0, 0)$.

In the second iteration, the following results are obtained for the node N_1 using step 3a:

(R-HR): $(z^1, x_1^1, x_2^1, y_{11}^1, y_{12}^1, y_{13}^1, y_{21}^1, y_{22}^1, y_{23}^1) = (-2.67, 1.33, 6, 0, 0, 1, 0.852, 0.025, 0.123)$

(LHR_λ): $(\hat{z}^1, \hat{y}_{11}^1, \hat{y}_{12}^1, \hat{y}_{13}^1, \hat{y}_{21}^1, \hat{y}_{22}^1, \hat{y}_{23}^1) = (-2.67, 1.33, 6.0, 0, 0, 1, 1, 0, 0)$

(SP): $(\bar{z}^1, \bar{x}_1^1, \bar{x}_2^1) = (-2.67, 1.33, 6)$ (point C in figure 5.1b).

Note that the solution obtained by (R-HR) is optimal in the continuous variables. However, it did not yield 0-1 values to $(y_{21}^1, y_{22}^1, y_{23}^1)$. After solving ($LHR_\lambda$), the best known solution is updated: $z^{up} = -2.67$, $(x^*, y^*) = (\bar{x}^0, \hat{y}^0)$. The node can be pruned since $z^1 = z^{up}$.

Node N_2 yields an integer solution with $z^2 = -0.154$ and it can be pruned. After evaluating N_1, N_2, all the nodes are pruned ($L = \emptyset$) and the algorithm terminates.

Figure 5.2 shows the disjunctive branch and bound tree for different versions of the algorithm for problem (5.4). Figure 5.2a shows the tree for the proposed algorithm, as described earlier (using step 3a in each of the nodes). Figure 5.2b presents the tree of the HR disjunctive branch and bound (i.e. the algorithm presented in Section 1.2.3 using Step 3a at every node). Finally, Figure 5.2b presents the tree of the BM disjunctive branch and bound (i.e. using Step 3b at every node).

It is easy to see from Figure 5.2 that the proposed algorithm requires fewer number of nodes than the other two algorithms (3 in the proposed algorithm, 6 in the HR, and 12 in the BM). Nevertheless, it requires the evaluation of 3 LPs in N_0 and in N_1, while the BM and HR disjunctive branch and bound require the evaluation of one LP at every node. In this simple example, the solution of every LP takes a fraction of a second. However, in larger problems the difference in solution time among (R-HR), (LHR_λ), (R-BM), and (SP) can be very significant. For example, in instance 11 of Section 5.6.1 the solution

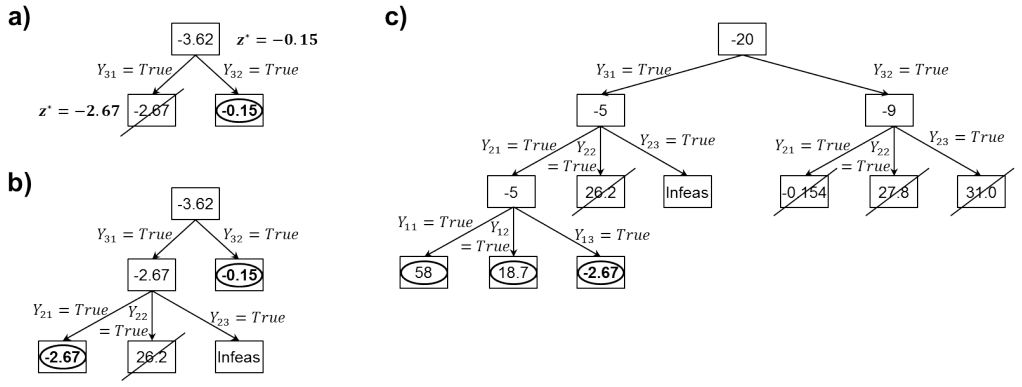

Figure 5.2: Disjunctive branch and bound tree for: a) the proposed algorithm, b) HR, and c) BM.

times (using CPLEX 12.6.1[40]) of (R-HR), (LHR_λ), (R-BM), and (SP) at the root node are 21.5s, 0.7s, 0.2s, 0.1s, respectively. Note that the solution time of (LHR_λ) is about 30 times faster than the solution of (R-HR) ((LHR_λ) is the Lagrangean dual of (R-HR), so they provide the same lower bound). Also, (LHR_λ) was evaluated as a single LP and a single core, so the difference in solution time comes from CPLEX exploiting the structure of (LHR_λ). If (LHR_λ) is solved by solving the smaller LPs in parallel (Property 5.3.2), the solution time can be further reduced.

5.6 Results

In this section, we present the performance of the proposed disjunctive branch and bound against the simple BM and HR disjunctive branch and bound. We also compare against a fourth disjunctive branch and bound with a "random heuristic". This algorithm is exactly the same as the proposed algorithm, but the primal heuristic is random (i.e. instead of solving (LHR_λ) and fixing $\hat{y}$ at its solution, $\hat{y}$ is fixed randomly). This comparison is important to show that (LHR_λ) provides a good heuristic for finding feasible solutions, and it is not only the additional work at each node what drives the improvement in the disjunctive branch and bound. The M-parameters of the Big-M reformulation were ob-

tained by using the variable bounds (i.e. for a constraint of the type $ax \leq b$ the parameter $M = \sum_{j:a_j \geq 0} x^{up} - \sum_{j:a_j < 0} x^{lo} - b$).

The results were obtained using an Intel(R) Core(TM) i7 CPU 2.93 GHz and 4 GB of RAM. The algorithm was implemented in GAMS 24.4.5[41] using CPLEX 12.6.1[40]. The proposed algorithm uses step 3b at every node, except for the root node (i.e. it solves (R-HR) at the root node, initializes the Lagrange multipliers, then solves (R-BM) at every other node without modifying the multipliers). (LHR_λ) is solved as a single LP by CPLEX, but computational experience shows that CPLEX can exploit the decomposable structure of (LHR_λ) (as described in Section 5.5). The algorithms use a breadth first branching strategy, and at every node selects the disjunction with fewest disjunctive terms that yielded non integer variables for branching.

The solution times presented in this section refer to the solve time (i.e. it includes the time to generate the model at every node, but ignores the time to create each node and to decide on branching and pruning). While in most cases the wall time is very close to the solve time, in some instances it is not. The reason for comparing the solve time (vs. wall time) is to consider only the time spent in solving the problem, and ignore inefficiencies in the code (that could be reduced in a lower level programming language or improvements in the branch and bound code).

Note that this implementation of the disjunctive branch and bound algorithm is a prototype, and its purpose is to show the improvement of the basic algorithm when using (LHR_λ) as a primal heuristic. The algorithm is implemented in GAMS, and it has to generate an MILP model at every node of the search tree. Also, it does not include presolve, heuristics or cuts, so it is much slower than CPLEX. Future work includes improvements in the implementation of the algorithm.

The algorithms were tested with 100 instances of 3 problems (i.e. a total of 300 instances). The solution time performance curve over all instances is presented in Figure 5.3. Figure 5.3 shows the percentage of instance solved vs. time. The figure shows four algorithms: “HR” refers to the HR disjunctive branch and bound, “BM” to the BM one, “ALG” to the proposed modified algorithm, and “RAN” to the algorithm with a random heuristic.

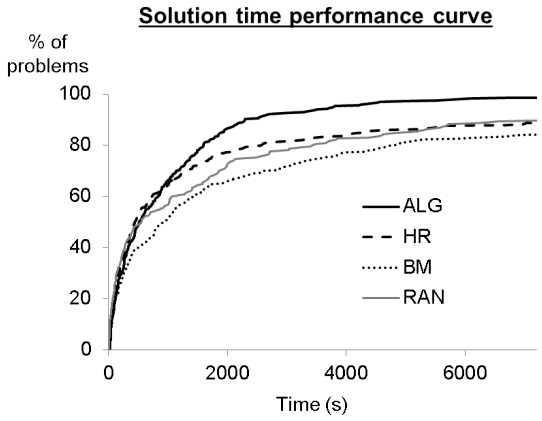

Figure 5.3: Solution time performance curve for the 300 tested instances.

It can be observed from the figure that the proposed algorithm is considerably better than the rest. The HR branch and bound performs second, but the random heuristic algorithm performs similarly in larger instances. The worst performer is the BM disjunctive branch and bound. Out of the 300 instances, the BM, HR, ALG, and RAN solve 253, 266, 296, and 269 instances, respectively.

While Figure 5.3 presents the performance over all instances, the performance varies in the different examples. In the remaining of this section, we present more details and results for each example. We first present random instances for the unstructured linear GDP problems. We then present results of two particular GDP problems: the strip packing and the contracts problem. The strip packing problem does not include logic constraints, while the contracts problem does. 100 instances are tested for each of these problems.

5.6.1 Unstructured GDP problems

For this example, 100 random instance are generated for the unstructured GDP problems without logic relations or global constraints (note that any GDP with global constraints can always be reformulated as a GDP without global constraints by using improper basic steps[10]). The coefficients of a^{ki} range between -1.00 and 1.00. The coefficient range

of c is $[-10.0, 10.0]$; of x^{lo} is $[-100, -10]$; and of x^{up} is $]10, 100]$. In order to avoid very high density in the matrices A^{ki}, and to give the problem structure (e.g. to relate more some variables to some disjunctions), the following formula was used to calculate $A^{ki}; k \in K, i \in D_k$ (where $e \in E_{ki}$ are the constraints in ki and $j \in J_k$ are the variables in $k \in K$):

$A^{ki}_{ej} = A0^{ki}_{ej} \alpha^{ki}_{ej} \beta^{ki}_{ej}$

where,

$A0^{ki}_{ej}$ is a random parameter between -1.00 and 1.00.

α^{ki}_{ej} is the probability of a variable appearing in an equation. (in all tested instances $\alpha^{ki}_{ej} = 0.5$).

$\beta^{ki}_{ej} = round(rand\text{-}between(0, 1 - |(j - k * |J|/|K|)/|J||))$.

To illustrate the idea behind the generation of A^{ki}, consider a problem with 40 variables, 10 disjunctions, 20 disjunctive terms per disjunction, and a maximum of 30 constraints per disjunctive term. In disjunction $k = 1$, the probability of variable $j = 1$ appearing in any constraint of any disjunctive term is $\alpha^{1,i}_{e,1}\beta^{1,i}_{e,1}$. Since $\beta^{1,i}_{e,1} = round(rand\text{-}between(0, 0.925))$, the probability is $\alpha^{1,i}_{e,1}\beta^{1,i}_{e,1} = (0.5)(0.46) = 23\%$. Note that, since there are up to 30 constraints in each of the 20 disjunctive terms of disjunction $k = 1$, there is a very high probability that this variable appears in a constraint of $k = 1$. For $k = 1$ and $j = 20$, $\beta^{1,i}_{e,20} = round(rand\text{-}between(0, 0.6))$ so the probability of variable $j = 20$ appearing in any constraint of any disjunctive term of $k = 1$ is 8%. The probability of variable $j > 25$ appearing in any constraint of any disjunctive term of $k = 1$ is 0%. The probability of variable $j = 20$ appearing in any constraint of any disjunctive term of $k = 5$ is 25%.

Table 5.1 presents the statistics for the instances that were generated randomly. All infeasible instances were removed from the test set, and all infeasible disjunctive terms were removed from the feasible instances. The number of constraints in a disjunctive term ($|E_{ki}|$) is random between $E_{max}/5$ and E_{max}. The problem identifier indicates the number of disjunctions $|K|$, disjunctive terms D_k, and number of continuous variables in the GDP.

Table 5.1: Average statistics for the different random problems

Problem	# instances	E_{max}	0-1 vars.	Variables		Constraints	
				BM	HR	BM	HR
k5-i40-v40	10	50	106	147	3,511	3,064	9,948
k5-i40-v50	10	50	113	164	4,650	3,344	12,515
k6-i25-v40	10	50	82	123	2,723	2,361	7,750
k6-i25-v50	10	50	86	137	3,523	2,523	9,533
k7-i15-v40	10	40	61	102	2,032	1,442	5,524
k7-i15-v50	10	40	63	114	2,567	1,490	6,674
k8-i5-v40	10	30	28	69	942	470	2,466
k8-i5-v50	10	30	29	80	1,202	500	3,061
k10-i3-v50	10	20	25	76	1,053	304	2,648
k10-i3-v80	10	20	25	106	1,655	304	4,022

The table shows the average number of binary variables, as well as the average problem size for the BM and HR. It can be observed from Table 5.1 that in some of these instances the difference in problem size between the BM and HR is considerable, particularly in the number of variables.

Figure 5.4 shows the performance of the different algorithms for the 100 random instances. Figure 5.4a shows that the solution time performance of the proposed and the random algorithms is much better than the traditional BM and HR disjunctive branch and bound for this example. The figure shows that the proposed and random algorithms solve 98 of the instances, while the BM and HR versions can only solve 81 and 72, respectively. Note that in this example, the BM disjunctive branch and bound performs better than the HR. This is because the number of nodes required in the BM and HR disjunctive branch and bound is very similar as can be observed in Figure 5.4b. Figure 5.4b shows the required number of nodes to solve the 72 instances that all algorithms solved. The figure shows that the proposed algorithm requires the fewest number of nodes. The BM and HR, however, require very similar number of nodes. Considering that the HR generates a much larger MILP (as shown in Table 5.1), it is expected that the solution time of the BM disjunctive branch and bound is faster than that of the HR.

Figure 5.5 shows the number of nodes required to find the optimal solution and the first

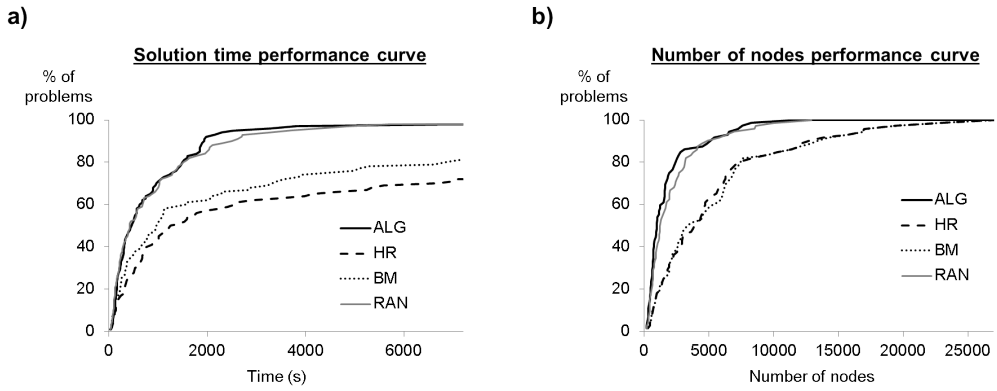

Figure 5.4: Performance curves for the different algorithm for: a) solution time, and b) number of nodes for the 72 instances that all algorithms solve.

feasible solution for the 72 instances that all algorithms solve. While the figure does not show the time to prove optimality, finding a feasible and the optimal solution is an important consideration in practice. It can be observed from the figure that the proposed algorithm is better at finding the optimal solution (Figure 5.5a) and the first feasible solution (Figure 5.5b). The random heuristic algorithm is very good in this example, requiring a similar number of nodes to find the first and optimal solution as the proposed method. However, the BM and HR versions require many more nodes to find a feasible solution and the optimal solution. The figure shows that there is small difference in the HR and BM disjunctive branch and bound with respect to the number of nodes required to find first and optimal solutions. The small difference seems to indicate that the HR requires fewer nodes to find the optimal solution, while the BM requires fewer to find the first feasible solution.

Note that the proposed algorithm performs better than the other versions of the disjunctive branch and bound for unstructured GDP instances. In the remaining of this section, we present results for two particular GDP problems: strip packing and contracts.

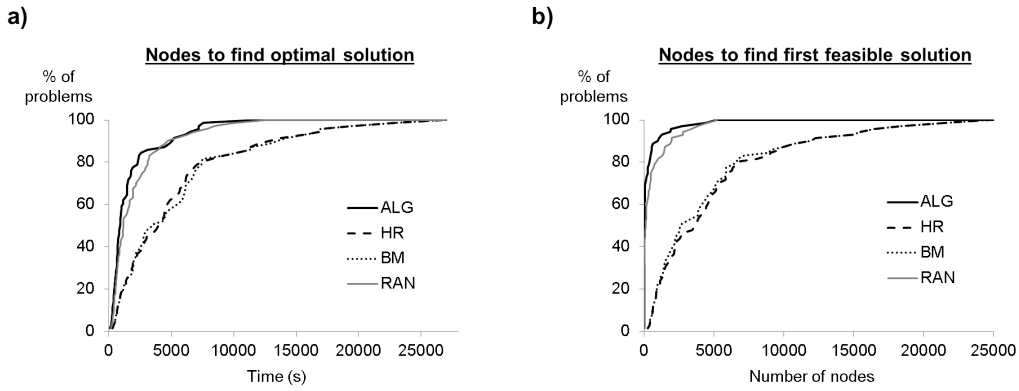

Figure 5.5: Number of nodes required to find: a) the optimal solution, and b) the first feasible solution for the 72 instances that all algorithms solve.

5.6.2 Strip packing problem

Given set of N rectangles, the strip packing problem consists on placing them on a strip while minimizing the its length. The rectangles cannot overlap or be rotated. The height and length of each rectangle is known ($H_i, L_i; i \in N$), and the strip has width W[29]. Figure A.1 illustrates the strip packing problem.

The GDP formulation of this problem is as follows[29,42]:

$$
\begin{aligned}
& & & min\ lt & & \\
& s.t. & & lt \geq x_i + L_i & & i \in N \\
& & & \begin{bmatrix} Z^1_{ij} \\ x_i + L_i \leq x_j \end{bmatrix} \vee \begin{bmatrix} Z^1_{ji} \\ x_j + L_j \leq x_i \end{bmatrix} & & \\
& & & \vee \begin{bmatrix} Z^2_{ij} \\ y_i - H_i \geq y_j \end{bmatrix} \vee \begin{bmatrix} Z^2_{ji} \\ y_j - H_j \geq y_i \end{bmatrix} & & i,j \in N, i < j \\
& & & Z^1_{ij} \veebar Z^1_{ji} \veebar Z^2_{ij} \veebar Z^2_{ji} & & i,j \in N, i < j \\
& & & 0 \leq x_i \leq UB - L_i & & i \in N \\
& & & H_i \leq y_i \leq W & & i \in N \\
& & & Z^1_{ij}, Z^2_{ij} \in \{True, False\} & & i,j \in N, i \neq j
\end{aligned}
\tag{5.6}
$$

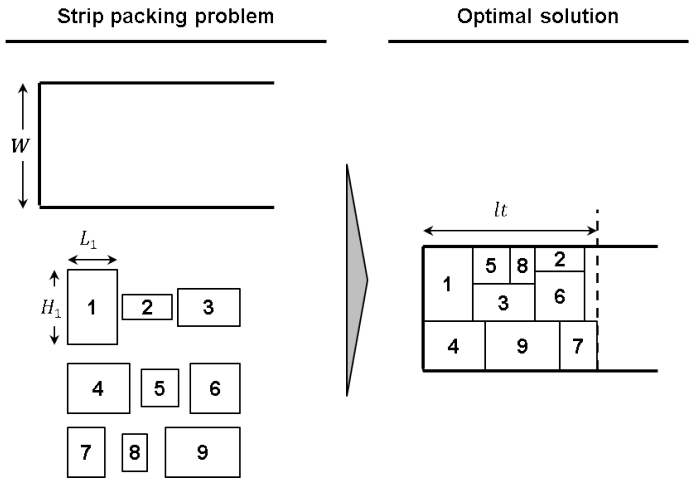

Figure 5.6: Illustration of the strip packing problem.

In (5.6), the objective is to minimize the length lt. The coordinates of the upper-left corner of rectangle i are represented with the variables (x_i, y_i). The global constraints ($lt \geq x_i + L_i$) enforce length of the strip corresponds to the largest $x_i + L_i$ (i.e. the coordinate of the top-left corner plus the length of the rectangle). There is a disjunction for every pair of rectangles $i, j \in N, i < j$. Each of the terms of the disjunction represents the relative position of rectangle i with respect to rectangle j. The first term, corresponding to $Z^1_{ij} = True$, represents rectangle i to the left of rectangle j. $Z^1_{ji} = True$ represents rectangle i to the right of rectangle j. $Z^2_{ij} = True$ represents rectangle i on top of rectangle j. Finally, $Z^2_{ji} = True$ represents rectangle i below rectangle j. The parameter UB is an upper bound for the strip (e.g. $UB = \sum_i L_i$).

The different algorithms were tested on 100 random instances of the strip packing problem. The range of values of the random parameters is as follows: $N = 4, 5; W =$ 5-7; $L_i =$ 1-10; $H_i =$ 2-5.

Figure 5.7a shows that the HR disjunctive branch and bound performs better than the other three for this problem. The BM, random heuristic, and the proposed branch and bound perform similarly in terms of solution time. It is interesting to note that the proposed algorithm performs the third in terms of solution time. However, Figure 5.7b shows that the

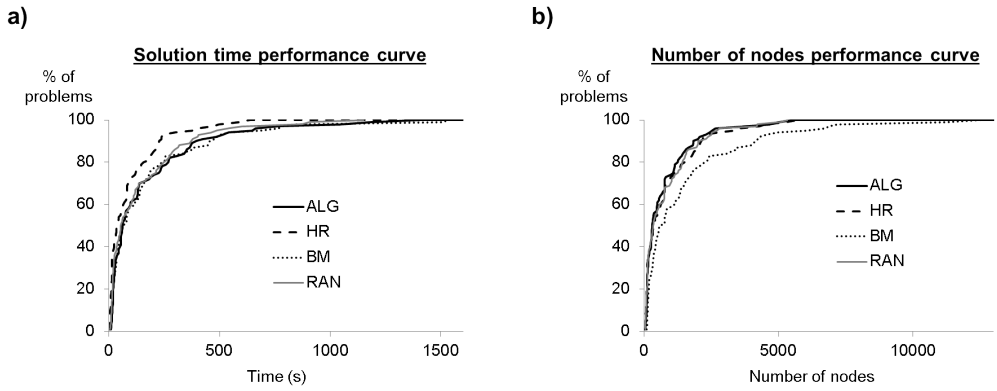

Figure 5.7: Performance curves for the different algorithm for: a) solution time, and b) number of nodes.

proposed algorithm requires the fewest number of nodes. Because some of the nodes require the evaluation of more than one LP, the fewer number of nodes is not reflected in the solution time for this problem. As expected, the random heuristic and the HR disjunctive branch and bound require fewer nodes than the BM disjunctive branch and bound. The random heuristic algorithm requires similar number of nodes as the HR. However, it requires the evaluation of two LPs in many of the nodes, so the performance in terms of the solution time is worse (as shown in Figure 5.7a). The heuristic that uses the Lagrangean relaxation requires fewer nodes than the random heuristic, which shows that the former is a better heuristic than randomly fixing variables. However, the improvement is small in this example, and hence the time spent in solving the Lagrangean relaxation at every node increases the total solution time. Note that for this problem the performance curves of all algorithms are not very different.

Figure 5.8 shows the performance curve of the number of nodes required to find the optimal solution and to find a feasible solution. Figure 5.8a shows that the number of nodes required to find an optimal solution is smaller for the proposed algorithm than for the others. Figure 5.8b shows that the first feasible solution can be almost always be found at the root node in the proposed and random algorithms. In particular, the proposed algorithm finds the first feasible solution at the root node in 94 instances; and in 4 instances it finds it at the first node. The improvement in finding optimal and feasible solutions can be better

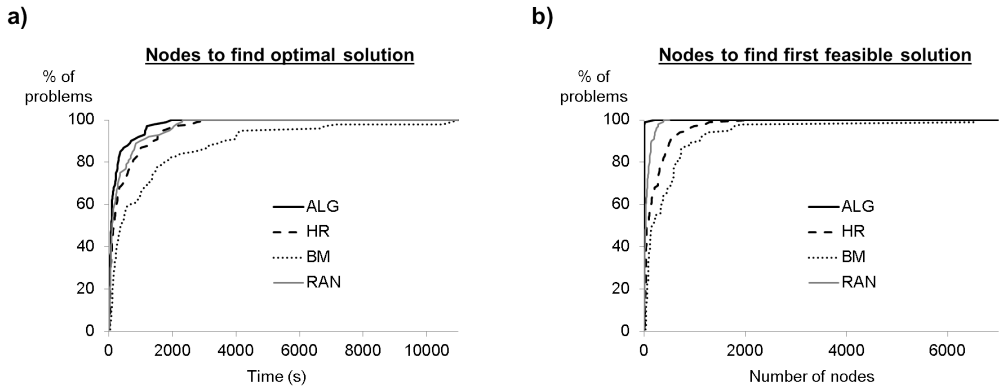

Figure 5.8: Number of nodes required to find: a) the optimal solution, and b) the first feasible solution.

Table 5.2: Average of performance metrics for the different algorithms

Average of metric	BM	HR	LAG	RAN
Solution time (s)	172.6	**93.0**	167.4	143.9
Total number of nodes	1,546	859	**763**	843
Number of nodes to find optimal	1,258	447	**229**	364
Number of nodes to find feasible	493	214	**2**	66.5

appreciated in Table 5.2. Table 5.2 shows an average of the metrics presented in Figures 5.7 and 5.8. Note that both algorithms with primal heuristic find the optimal and feasible solutions faster than the BM and HR disjunctive branch and bound. Out of the two algorithms with primal heuristic, the one that uses the Lagrangean relaxation finds the optimal and first feasible solution in fewer nodes.

5.6.3 Contracts problem

Given feedstock requirements of a raw material at every time period $D_t; t \in T$, the problems consists on finding the best purchasing contracts to minimize costs. The inventory of the feedstock i_t allows to carry material from a time period to the next ones, but there is a cost associated with inventory α_t^I. In general, there are three types of contracts. The "standard contract" which allows buying any amount of material at a given price γ_t. The

"bulk contract" which provides a discount β_t^B of the material, but there is a minimum purchase requirement $F_t^{B,lo}$. The last type of contract requires purchasing materials for the following $q \geq 1$ time periods, all of which include the same discount β_{tq}^L over the same price γ_t and the same minimum purchase requirement $F_{tq}^{L,lo}$.

The GDP of the contracts problem is as follows:

$$
\begin{aligned}
& min \ \sum_{t \in T} (\alpha_t^S s_t + c_t) \\
s.t. \quad & f_t \geq D_t && t \in T \\
& s_t = s_{t-1} + x_t - f_t && t \in T \\
& \begin{bmatrix} Y_t^S \\ c_t = \gamma_t x_t \end{bmatrix} \vee \begin{bmatrix} Y_t^B \\ c_t = (1 - \beta_t^B)\gamma_t x_t \\ x_t \geq F_t^{B,lo} \end{bmatrix} \vee \\
& \vee \begin{bmatrix} Y_t^0 \\ 0 \leq c_t \end{bmatrix} \underset{q=1,\ldots,|T|-t}{\vee} \begin{bmatrix} Y_{tq}^L & \\ c_{t'} = (1 - \beta_{tq}^L)\gamma_t x_{t'} & t' = t, ..., t+q \\ x_{t'} \geq F_{tq}^{L,lo} & t' = t, ..., t+q \end{bmatrix} && t \in T \\
& Y_t^S \veebar Y_t^B \veebar Y_t^0 \underset{q=1,\ldots,|T|-t}{\veebar} Y_{tq}^L && t \in T \\
& Y_t^0 \Leftrightarrow \underset{\substack{t'<t \\ q \geq t-t'}}{\vee} Y_{t'q}^L && t \in T \\
& Y_1^0 = False \\
& 0 \leq x_t \leq x^{up} && t \in T \\
& 0 \leq c_t \leq c^{up} && t \in T \\
& 0 \leq s_t \leq s^{up} && t \in T \\
& Y_t^S, Y_t^B, Y_t^0 \in \{True, False\} && t \in T \\
& Y_{tq}^L \in \{True, False\} \qquad 1 \leq q \leq |T| - t; && t \in T
\end{aligned}
\tag{5.7}
$$

The global constraints in (5.7) enforce the demand satisfaction ($f_t \geq D_t$) and the material balance at the inventory ($s_t = s_{t-1} + x_t - f_t$). At each time period $t \in T$ there is a

disjunction. The term that corresponds to the Boolean variable Y_t^S represents the "standard contract", where any amount x_t can be purchased at price γ_t. Y_t^B represents the "bulk contract", where $x_t \geq F_t^{B,lo}$ can be purchased with a discount in the price β_t^B. Y_t^0 is active when a long term contract from a previous time period is selected, and such a contract involves time period t ($Y_t^0 \Leftrightarrow \vee_{\substack{t'<t \\ q \geq t-t'}} Y_{t'q}^L$). For example, if $Y_{1,3}^L$ is selected (e.g. in the time period $t = 1$, a contract with a length of $q = 3$ time periods was selected), then $Y_2^0 = Y_3^0 = Y_4^0 = True$. The term associated with Y_t^0 does not constrain the variables, since the corresponding cost and purchase bound are determined at the time period in which the long term contract is active ($c_{t'} = (1 - \beta_{tq}^L)\gamma_t x_{t'}; x_{t'} \geq F_{tq}^{L,lo}; t' = t, ..., t+q$). For the first time period, Y_1^0 cannot be selected. When performing the BM or HR reformulation, the logic constraint ($Y_t^0 \Leftrightarrow \vee_{\substack{t'<t \\ q \geq t-t'}} Y_{t'q}^L$) can be reformulated as ($y_t^0 = \sum_{\substack{t'<t \\ q \geq t-t'}} y_{t'q}^L$).

Note that (5.7) involves logic constraints. Therefore, in order to preserve Properties 5.3.2 and 5.3.3, it would be necessary to use (5.3). However, the continuous relaxation of the HR and BM reformulations of (5.7) (as well as (LHR_λ) with the logic constraints reformulated for the original binary variables) can be solved in a fraction of a second. Therefore, Property 5.3.2 (which established that (LHR_λ) can be solved by solving small LPs) does not have an important impact in the solution time of the problem. Furthermore, out of the over 2.2 million nodes solved for the instances of this example, there was not a case in which the continuous relaxation of (LHR_λ) with the logic constraints reformulated for the original variables gave a non-integer solution. While this is no proof that Property 5.3.3 holds, it indicates that (LHR_λ) with the logic constraints reformulated for the original variables can be used as a primal heuristic for the tested instances.

The different algorithms were tested with 100 random instances of the contracts problem. The range of values of the random parameters is as follows: $|T| = 9\text{-}11; D_t = 50\text{-}100; \alpha_t^I = 5\text{-}20; \gamma_t = 10\text{-}30; \beta_t^B = 0.050\text{-}0.500; \beta_{tq}^L = 0.010\text{-}0.999; F_t^{B,lo} = 50\text{-}100; F_{tq}^{L,lo} = 50\text{-}100$. The algorithm uses ($LHR_\lambda$) with the logic constraints reformulated for the original variables as the primal heuristic.

Figure 5.9 presents the performance plots of the algorithms for solving the contracts problem instances. Figure 5.9a shows the performance of the solution times. It can be observed

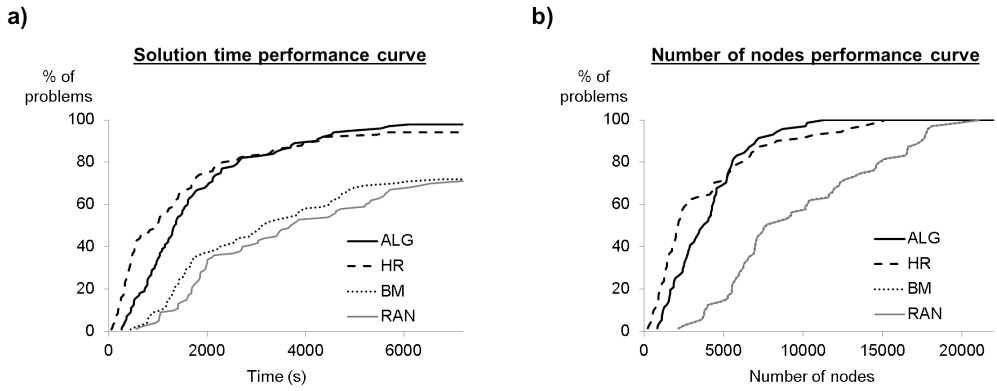

Figure 5.9: Performance curves for the different algorithm for: a) solution time, and b) number of nodes for the 71 instances that all algorithms solve.

that the HR disjunctive branch and bound performs the best for the smaller problems. However, the proposed algorithms outperforms the HR after about 4,000 seconds. Of the 100 instances, the proposed algorithm solves 98, the HR 94, the BM 72, and the random heuristic 71. It is clear from Figure 5.9a that the BM and heuristic algorithms perform much worse than the other two for this problem. Figure 5.9b shows the number of nodes required to solve the 71 instances that all algorithms solve. Similar to the solution time, the HR and the proposed algorithm are the best performers. Their performance is close, the HR being better at first and the proposed algorithm taking over in the more difficult problems. In this example, the heuristic algorithms performs exactly the same as the BM branch and bound, in terms of number of nodes. This indicates that the random heuristic is poor, since it requires additional work and it does not reduce the number of nodes required for the BM disjunctive branch and bound.

Figure 5.10 presents the number of nodes required to find the optimal solution and the first solution. Figure 5.10a shows that the proposed algorithm is faster in finding the optimal solution. Furthermore, Figure 5.10b shows that the algorithm finds a feasible solution in every instance at the root node. Again, these two figures show that the heuristic algorithms performs exactly the same as the BM branch and bound in terms of number of nodes. For this example, the BM and random heuristic branch and bound perform much worse than the HR and the proposed algorithm.

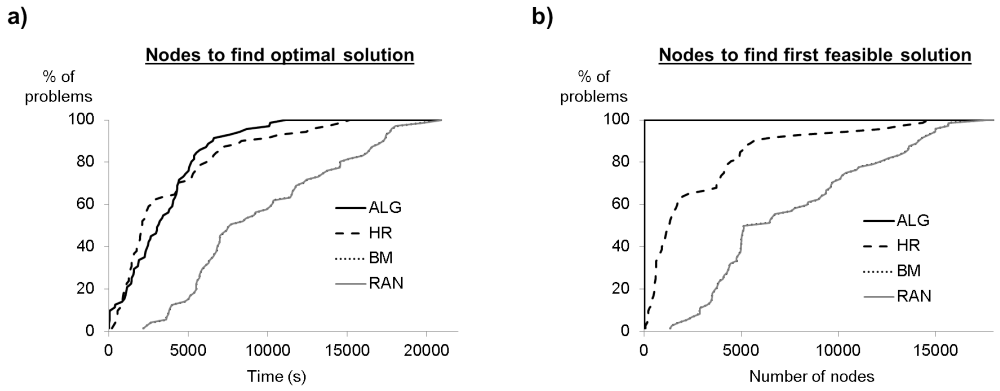

Figure 5.10: Number of nodes required to find: a) the optimal solution, and b) the first feasible solution for the 71 instances that all algorithms solve.

5.7 Conclusions

In this chapter, we have presented a Lagrangean relaxation of the hull-reformulation of linear GDP problems. This Lagrangean relaxation can be applied to any linear GDP. We proved two important properties of this relaxation. The first one is that any vertex of the continuous relaxation of the Lagrangean relaxation yields 0-1 values for the binary variables of the hull reformulation. The second one is that it can be solved by solving several small LPs (potentially in parallel). This relaxation was incorporated into a disjunctive branch and bound as a primal heuristic, and the proposed algorithm was tested with 300 instances on 3 problems: unstructured GDPs, strip packing and contracts.

For every problem, the number of nodes required by the proposed algorithm is smaller than the number of nodes required by the HR, BM, and "random heuristic" disjunctive branch and bound. Furthermore, the number of nodes to find a feasible and optimal solution to problems is drastically improved when using the Lagrangean relaxation as a primal heuristic. Over all the instances, the solution time performance of the proposed algorithm is better than alternative disjunctive branch and bound methods. Over all the instances, the BM, HR, ALG, and RAN solve 253, 266, 296, and 269, respectively.

In terms of solution time, the proposed algorithm and the heuristic one are the best for

the unstructured GDP instances. Not only is their solution time performance curve better, but they solve considerably more instances than the rest within the time limit (98% for the proposed and heuristic algorithms vs. 81% for the BM, and 72% HR). For the two examples of structured GDP problems, the performance of the solution time is different for each of the problems. For the strip packing problem, the HR disjunctive branch and bound performs the best, while de proposed algorithm performs third (and very close to the last performer: the BM disjunctive branch and bound). For the contracts problem, the proposed algorithm performs the best, solving 98% of the instances (while the BM, HR, and random heuristic algorithm solve 72%, 94%, and 71%). The performance curve is similar to that of the HR disjunctive branch and bound, but it is able to solve more instances.

The proposed Lagrangean relaxation can be extended to nonlinear convex GDP problems. Future work will address these problems through the Lagrangean relaxation described in this chapter, as well as improvements in the implementation.

Chapter 6

Cutting planes for improved global logic-based outer-approximation of nonconvex GDP problems

6.1 Introduction

This chapter is motivated by the synthesis of process networks, which is an area of active research in Process Systems Engineering (PSE). The objective is to synthesize the optimal process flowsheet contained in a process superstructure[43], which contains alternative units (with their corresponding models) and interconnections. The synthesis process networks can be modeled as a mixed-integer nonlinear problem (MINLP)[44]. In the MINLP approach, the selection of units and interconnections are modeled using binary variables. The process unknowns (flow, concentration, temperature, etc.) are modeled using continuous variables. Alternative superstructure representations of processes have also been proposed[45,46,47,48].

The synthesis of process networks yields MINLP models that can be highly nonconvex. In particular, the nonconvexities arise in two forms. The first form involves the modelling

of each individual process unit. This models can range from linear input/output equations to large differential algebraic models. The second form of nonconvexities arise in the modeling of flow and properties that result when mixing streams, in the simplest case as bilinear terms. It is important to note that when the units and interconnections are fixed in a superstructure, the resulting problem is a nonlinear program (NLP). Not only is this NLP continuous, it does not include nonconvexities related to the units or interconnections that were not selected. Because of these two reasons, it is common that the resulting NLP after fixing the discrete decisions is much simpler to solve than the full problem, provided that the constraints of the non-selected units and interconnections are removed from the NLP model. This property is not general for MINLPs, but very common in the synthesis process networks. General methods for solving MINLPs to global optimality do not take advantage of this particular property.

The most common deterministic method for solving nonconvex MINLP problems is the spatial branch and bound algorithm[49]. This algorithm is used by several general purpose MINLP global solvers, such as: αBB[50], ANTIGONE[51], BARON[52,53], Couenne[54], LINDO[55], and SCIP[24]. We refer the reader to the work by Bussieck and Vigerske[56] for details of the different MINLP solvers. In addition to the spatial branch and bound, Kesavan et al.[57] present an outer-approximation method for global optimization. This method builds on the traditional outer-approximation method for convex optimization[58,59].

Other methods have been developed to exploit the simplification of problems when discrete decisions are fixed, typically in the form of logic-based Benders decomposition[60] (and Generalized Benders decomposition[61,62]). In particular for nonconvex MINLP, Li et al.[63] present a nonconvex generalized Benders decomposition method for stochastic MINLPs.

Synthesis of process networks is one of the areas where GDP has been most successful. Raman and Grossmann[64] propose a GDP model for the synthesis of process networks. The logic-based outer-approximation is of particular interest in the synthesis of process networks. This method exploits the fact that the NLP that is generated when fixing the discrete alternatives is much simpler to solve than the original problem. This method iteratively solves a linear GDP approximation of the original GDP (master problem) and

an NLP in which the discrete decisions are fixed (subproblem). The original logic-based outer-approximation is valid only for convex GDP problems. However, a valid logic-based outer-approximation for the global optimization of nonconvex GDP problems is presented by Bergamini et al.[65].

It is important to note that all of the global methods discussed so far require a convex MINLP/MILP relaxation of the original problem. This relaxation is obtained by reformulating the problem in univariate and some specific multivariate functions[66] and then overestimating the feasible region of each constraint with convex inequalities[67,68,69].

In this chapter, we improve the global logic-based outer-approximation with two enhancements: one for finding feasible solutions faster, and a new strategy for improving the linear GDP approximation using cutting planes. The first enhancement is the partition of the algorithm into two phases. The first phase allows the evaluation of many discrete alternatives for a short period of time, but does not guarantee termination in a finite number of iterations. The second phase is the rigorous global logic-based outer-approximation that terminates in a finite number of iterations. In order to diversify the search in the first phase, a penalty term in the objective function is also included. The second enhancement, and main contribution of this chapter, is a cutting plane method for improving the linear approximation of the nonconvex GDP. This method derives cuts based on the complete feasible region of the processing units, not based on individual constraints.

This chapter is organized as follows. Section 6.2 provides an overview of the basic form of the global logic-based outer-approximation. Section 6.3 presents the two enhancements to the global logic-based outer-approximation. First, the partition of the algorithm into two phases. Second, the new method for deriving cutting planes that improve the lower bound of the master problem. A simple illustrative example is also presented in this section. The algorithm is tested with several instances of the layout-optimization of screening systems in recovered paper production, several instances of a simplified generic superstructure that involves reactors and separation units, and a more realistic test case for the design of a distillation column for the separation of benzene and toluene with ideal equilibrium. The examples and results are presented in Section 6.4.

6.2 Global logic-based outer-approximation

In this chapter, we represent the general GDP formulation as follows:

$$
\begin{aligned}
& & & min\ c^T x & & \\
& s.t. & & g(x) \leq 0 & & \\
& & & \bigvee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ r_{ki}(x) \leq 0 \end{bmatrix} & & k \in K \\
& & & \bigvee_{i \in D_k} Y_{ki} & & k \in K \\
& & & \Omega(Y) = True & & \\
& & & x^{lo} \leq x \leq x^{up} & & \\
& & & x \in \mathbb{R}^n & & \\
& & & Y_{ki} \in \{True, False\} & & k \in K, i \in D_k
\end{aligned}
\tag{GDP2}
$$

In (GDP2), the objective function is linear in the continuous variables $x \in \mathbb{R}^n$. Note that this is a general representation of any GDP. If the objective function is nonlinear $f(x)$, a new variable x_{n+1} is introduced and the objective function is $min\ x_{n+1}$ with $x_{n+1} \geq f(x)$ as a constraint.

The idea behind the basic global logic-based outer-approximation (GLBOA) is similar to the convex logic-based outer-approximation. GLBOA iteratively solves a master problem and a subproblem. The master problem is a linear GDP relaxation of the nonconvex GDP, and the subproblem is an NLP in which the discrete decisions are fixed. The main difference in the algorithms is the method for obtaining the master problem and the method for solving the NLP subproblem (which needs to be solved to ϵ-global optimality).

The NLP subproblem of GLBOA, for a given alternative Y^P in (GDP2) is as follows:

$$\begin{aligned} & min\ c^T x \\ s.t. \quad & g(x) \leq 0 \\ & r_{ki}(x) \leq 0 \qquad \forall Y^P_{ki} = True \\ & x^{lo} \leq x \leq x^{up} \\ & x \in \mathbb{R}^n \end{aligned} \tag{SP}$$

(SP) is an NLP in which the constraints $r_{ki}(x) \leq 0$ that correspond to $Y^P_{ki} = True$ are enforced, while the constraints that correspond to $Y^P_{ki} = False$ are ignored. Note that (SP) can be nonconvex and it needs to be solved to global ϵ-global optimality for the GLBOA method to be valid. If (SP) is solved with a global optimization method, the method will provide and upper bound (Z^*) and lower bound (Z^P) for the objective function (i.e. Z^* corresponds to the feasible solution; hence, if (SP) is solved to ϵ-global optimality, then $(Z^* - Z^P)/Z^* \leq \epsilon$).

For a given Y^P, let $P \in FS$ if (SP) is feasible and let Z^P be a lower bound for the objective function in (SP). Let $P \in IS$ if (SP) is infeasible. Let y^P be the corresponding binary representation of fixed alternative Y^P. Consider the following integer "no-good-cuts"[70,71] for a set of alternatives $Y^p; p = 1, ..., P$ in which the subproblem was evaluated:

$$Z \geq (Z^p - LB)\left(1 - \sum_{y^p_{ki}=0}(y_{ki}) - \sum_{y^p_{ki}=1}(1 - y_{ki})\right) + LB \qquad p \in FS \tag{6.1a}$$

$$\sum_{y^p_{ki}=0}(y_{ki}) + \sum_{y^p_{ki}=1}(1 - y_{ki}) \geq 0 \qquad p \in IS \tag{6.1b}$$

where Z is the objective function and LB is a global lower bound of the objective function. (6.1a) indicates that $Z \geq Z^p$ if $y = y^p$ and $p \in FS$. It indicates $Z \geq LB$ for any other alternative. (6.1b) indicates that alternative y is infeasible for $y = y^p$ and $p \in IS$.

The master problem (linear GDP) can be formulated using (6.1):

$$
\begin{aligned}
& min\ Z && && \\
s.t.\quad & Z \geq c^T x && && \\
& A\hat{x} \leq a && && \\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ y_{ki} = 1 \\ B_{ki}\hat{x} \leq b_{ki} \end{bmatrix} && && k \in K \\
& \bigvee_{i \in D_k} Y_{ki} && && k \in K \\
& \sum_{i \in D_k} y_{ki} = 1 && && k \in K \\
& \Omega(Y) = True && && \\
& Z \geq (Z^p - LB)\left(1 - \sum_{y^p_{ki}=0}(y_{ki}) - \sum_{y^p_{ki}=1}(1 - y_{ki})\right) + LB && && p \in FS \\
& \sum_{y^p_{ki}=0}(y_{ki}) + \sum_{y^p_{ki}=1}(1 - y_{ki}) \geq 0 && && p \in IS \\
& \hat{x}^{lo} \leq \hat{x} \leq \hat{x}^{up} && && \\
& \hat{x} \in \mathbb{R}^{n+s} && && \\
& 0 \leq y_{ki} \leq 1 && && k \in K, i \in D_k \\
& Y_{ki} \in \{True, False\} && && k \in K, i \in D_k
\end{aligned}
\tag{MP}
$$

The master problem (MP) is a linear GDP relaxation of the original GDP. The variables y are included inside the disjunctive terms of (MP) only to simplify the representation of the no-good-cuts and the description of the algorithm. These variables are not required in the GDP representation, and in the MILP reformulation of the GDP this variables are the same as y presented in (HR) and (BM). Some linear relaxations require the use of additional variables (e.g. when separating large constraints into univariate terms). For this reason, (MP) optimizes $\hat{x} = (x, x_{aux})$, which involves the original variables $x \in \mathbb{R}^n$

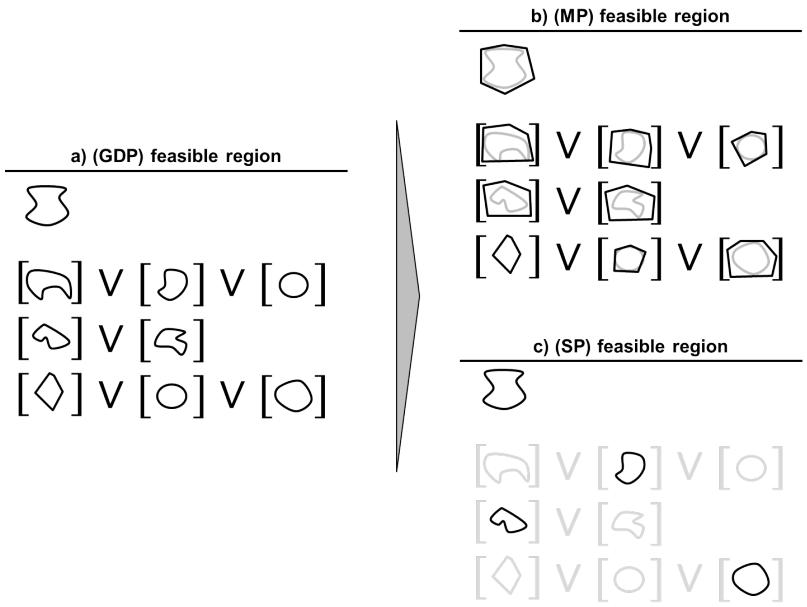

Figure 6.1: Illustration of the feasible region of a) the original (GDP), b) the linear GDP relaxation (MP) and c) the subproblem (SP).

and possibly auxiliary variables $x_{aux} \in \mathbb{R}^s$. $A\hat{x} \leq a$ is a linear relaxation of $g(x) \leq 0$, and $B_{ki}(\hat{x}) \leq b_{ki}$ is a linear relaxation of $r_{ki}(x) \leq 0$. There are different methods for obtaining the linear relaxations of the nonlinear constraints. Two of these methods include dropping the constraints that include nonlinear terms and using polyhedral envelopes for the nonlinear constraints[67,68]. The former yields a weaker master problem than the latter (i.e. it provides a worse lower bound). Stronger approximations can be achieved by using piecewise linear relaxations[6] and nonlinear convex approximations[68]. These two types of relaxations will not be addressed in this thesis.

Figure 6.1 illustrates the feasible region of the original of (GDP), the master problem (MP) and the subproblem (SP). Figure 6.1.a) illustrates the feasible region of a GDP. Note that the disjunctive terms involve linear, convex and nonconvex feasible regions. Figure 6.1.b) illustrates a linear GDP relaxation of the original GDP. Finally, the intersection of the feasible regions presented in Figure 6.1.c) represents the resulting NLP subproblem for $Y_{12} = Y_{21} = Y_{33} = True$.

The basic global logic-based outer-approximation is as follows:

0. Initialize. Let $LS = IS = \emptyset$. Set $\epsilon_1 \geq \epsilon_2 > 0$. Set $P = 1$ and $UB = \infty$.

1. Solve master problem. Solve (MP). Let $(Z^*, \hat{x}^*, Y^*, y^*)$ be the optimal solution of (MP). Set $y^P = y^*$, $Y^P = Y^*$ and $LB = Z^*$.

2. Solve subproblem. Solve (SP), with fixed Y^P, to ϵ_2-global optimality.

If (SP) is feasible, let (Z^*, x^*) be the optimal solution of (SP). Let Z^P be a lower bound for the objective function, provided by the NLP global solution method, and set $P \in FS$. If $Z^* < UB$, let $UB = Z^*$ and $(\bar{Z}, \bar{x}, \bar{Y}) = (Z^*, x^*, Y^P)$.

If (SP) is infeasible, set $P \in IS$.

3. Terminate. If $(UB - LB)/UB \leq \epsilon_1$, terminate with optimal solution $(\bar{Z}, \bar{x}, \bar{Y})$. Else, set $P = P + 1$ and go to step 1.

Theorem 6.2.1 *The basic global logic-based outer-approximation terminates in a finite number of iterations.*

Proof. The cuts (6.1) are included in the master problem. This cuts enforce that: a) for $Y^p, p \in IS$ the master problem will be infeasible; b) for $Y^p, p \in FS$ the optimal solution of the master problem will be $Z^* \geq Z^p$. This means that if all the Y^p solutions that are feasible for (MP) are evaluated, then $LB \geq \min_{p \in FS} Z^p$. (SP) is solved to ϵ_2-global optimality, so $(UB - \min_{p \in FS} Z^p)/UB \leq \epsilon_2$. Since $\epsilon_1 \geq \epsilon_2$, then $(UB - LB)/UB \leq \epsilon_1$ ■.

6.3 Improved global logic-based outer-approximation

The basic global logic-based outer-approximation terminates in a finite number of iterations, but the convergence can be slow. In this section we present two enhancements: one is to find feasible solutions faster, and another one is to improve the lower bound provided by the master problem.

6.3.1 Two-phase algorithm to improve Upper Bound

The basic global logic-based outer-approximation requires the solution of the NLP subproblem to global optimality. Therefore, it is possible that the algorithm takes a very long time even in the first iteration. In order to evaluate many iterations, it is possible to modify the algorithm and consider a relatively short time limit for the solution of the subproblem ($\tau_{sub-limit}$). The step 2 of the algorithm can be modified as follows:

2. Solve subproblem. Solve (SP), with fixed Y^P and time limit $\tau_{sub-limit}$, using a global optimization method.

If (SP) is proven infeasible, set $P \in IS$.

If at least one feasible solution for (SP) is found, let (Z^*, x^*) be the best feasible solution found for (MP). Let Z^P be a lower bound for the objective function, provided by the NLP global solution method, and set $P \in FS$. If $Z^* < UB$, let $UB = Z^*$ and $(\bar{Z}, \bar{x}, \bar{Y}) = (Z^*, x^*, Y^P)$.

If no feasible solution is found, but (SP) is not proven infeasible, let Z^P be a lower bound for the objective function provided by the NLP global solution method. Set $P \in FS$.

This modification of the algorithm allows to evaluate several different alternatives in a short period of time, assuming that the lower bound of the subproblems is still higher than the lower bound of the master problem. The downside of this modified algorithm as that it does not necessarily terminate in a finite number of iterations. In order to ensure that the algorithm terminates, the global logic-based outer-approximation can be divided into two phases. The first phase is the modified version with a time limit for solving the subproblem. The second phase is the basic global logic-based outer-approximation described in the previous section.

Note that this enhancement is useful when the solution to global optimality of the subproblem is the most time consuming step. It considers that the solution of the master problem (linear GDP) is "easy" in comparison. Considering that the master problem can be solved

as an MILP, and that MILP solvers have become considerably efficient, this is usually true. Furthermore, in the synthesis of process networks the nonlinear terms associated to the operation can be highly nonconvex. Therefore, solving an NLP with few units can be difficult to be handled by NLP global solvers. On the other hand, a linear GDP with a few hundred alternative units (e.g. a few hundred binary variables in the MILP reformulation) can typically be easily handled by MILP solvers.

An additional enhancement to the two-phase algorithm is to diversify the search of feasible solutions in the first phase. The reason for this is that the no-good-cuts (6.1) avoid the evaluation of the exact same alternative, but the cuts have no effect if there is just one difference in an alternative (vs. the alternatives previously evaluated). In order to search for more diverse solutions, the objective function in the first phase can be modified as follows:

$$min\ Z - W \sum_{p=1,\dots,P-1} \left(\sum_{y^p_{ki}=0} (y_{ki}) + \sum_{y^p_{ki}=1} (1 - y_{ki}) \right) \tag{6.2}$$

where W is a positive weighting parameter.

The modification of the objective function in (6.2) promotes the search for solutions that are "very different" from previously evaluated alternatives. This penalty term is only included in the first phase. Note that the algorithm has to be slightly modified in step 1 to ensure a valid lower bound, so the lower bound becomes $LB = Z^* - W \sum_{p=1,\dots,P} \left(\sum_{y^p_{ki}=0} (y^*_{ki}) + \sum_{y^p_{ki}=1} (1 - y^*_{ki}) \right)$ instead of $LB = Z^*$.

6.3.2 Cutting planes to improve Lower Bound

The no-good-cuts (6.1) ensure the termination of the algorithm in a finite number of iterations. However, these cuts are useful only in avoiding the evaluation of previous solutions, and they normally do not have much impact in improving the lower bound. For this reason, we propose a new method to derive valid cuts that help to improve the lower bound.

The main objective of the proposed method is to derive linear inequalities that help to

represent better the individual feasible region of the selected disjunctive terms. In the synthesis of process networks these linear inequalities will try to improve the linear representation of the nonconvex operating region of each unit. These linear inequalities will be obtained through a separation problem, extending the approach of Stubbs and Mehrotra[14] to nonconvex feasible regions. The cuts will be based on the strongest possible convex envelope (i.e. the convex hull) of the complete feasible region of the disjunctive term.

For clarity purposes, we first present the case in which the cut by Stubbs and Mehrotra[14] can be directly applied. Consider that, for a disjunctive term that was selected by the master problem (Y^P), a nonlinear convex relaxation is available ($\hat{r}_{ki}(\hat{x}) \leq 0$). Consider that this convex relaxation is stronger than the linear relaxation, so it is possible to establish the following relations on the feasible region of the selected term:

$(r_{ki}(x) \leq 0) \subseteq (\hat{r}_{ki}(\hat{x}) \leq 0) \subseteq (B_{ki}\hat{x} \leq b_{ki})$, where $\hat{x}^{lo} \leq \hat{x} \leq \hat{x}^{up}$, $\hat{x} = (x, x_{aux})$, $\hat{x}^{lo} = (x^{lo}, x^{lo}_{aux})$, and $\hat{x}^{up} = (x^{up}, x^{up}_{aux})$. Figure 6.2.a) illustrates these three feasible regions, projected into the original space x.

Let x^* be the solution of the master problem at iteration P. Let $Y^P_{ki} = True$ be a selected disjunctive term at iteration P, and assume that a nonlinear convex relaxation, that is stronger than the linear relaxation, is available for that term ($\hat{r}_{ki}(\hat{x}) \leq 0$). It is then possible to obtain cuts in the original space x, that separate a point in x^* from the feasible region $\hat{r}_{ki}(\hat{x}) \leq 0$, using the following separation problem[14]:

$$\begin{aligned} & min \ ||x - x^*||_2^2 \\ s.t. \quad & \hat{r}_{ki}(\hat{x}) \leq 0 \\ & \hat{x}^{lo} \leq \hat{x} \leq \hat{x}^{up} \\ & \hat{x} \in \mathbb{R}^{n+s} \end{aligned} \tag{6.3}$$

where $\hat{x} = (x, x_{aux})$.

Note that (6.3) is a convex NLP. Let $\tilde{x}^P_{ki}$ be the value of x at the optimal solution of (6.3).

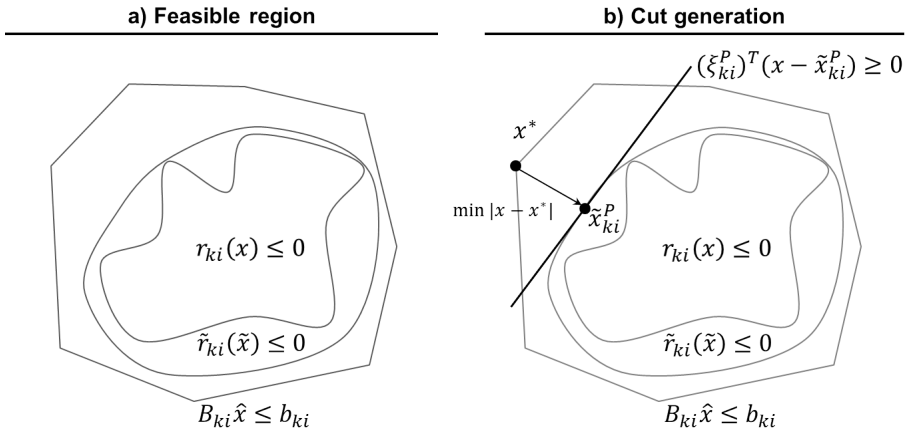

Figure 6.2: Illustration of the feasible region and cuts generated for $r_{ki}(x) \leq 0$, $\hat{r}_{ki}(\hat{x}) \leq 0$, and $B_{ki}\hat{x} \leq b_{ki}$; projected into the original space.

Then the following inequality is a valid cut for $\hat{r}_{ki}(\hat{x}) \leq 0$.

$$\left(\xi^P_{ki}\right)^T (x - \tilde{x}^P_{ki}) \geq 0 \tag{6.4}$$

where $\xi^P_{ki} = 2(\tilde{x}^P_{ki} - x^*)$.

Inequality (6.4) lies in the original space of the variables x, and it is valid for any convex region $\hat{r}_{ki}(\hat{x}) \leq 0$[14]. Because $(r_{ki}(x) \leq 0) \subseteq (\hat{r}_{ki}(\hat{x}) \leq 0)$, it is also valid for $r_{ki}(x) \leq 0$. Note that the objective function in (6.3) evaluates the distance using the square of the Euclidean norm, which provides a good cut and an analytical expression for the subgradient ξ^P_{ki}. Any other norm also provides a valid cut, but the subgradient ξ^P_{ki} has to be adjusted accordingly. The separation problem (6.3), and the cut generated from it (6.4) are illustrated in Figure 6.2.b) (projected into the original space x).

In order to obtain the cut, it is necessary to solve (6.3), which requires the knowledge of a convex relaxation of the problem. However, we present an alternative separation problem that not only allows to obtain a cut without $\hat{r}_{ki}(\hat{x}) \leq 0$, but it also derives the cut using the strongest possible convex envelope of $r_{ki}(x) \leq 0$ (i.e. its convex hull). The new separation

problem is as follows:

$$
\begin{aligned}
& & & min\ ||x - x^*||_2^2 & \\
& s.t. & & x = \sum_{j=1,...,\hat{n}} \lambda_j \hat{x}_j & \\
& & & r_{ki}(\hat{x}_j) \leq 0 & j = 1, ..., \hat{n} \\
& & & x^{lo} \leq \hat{x}_j \leq x^{up} & j = 1, ..., \hat{n} \\
& & & \sum_{j=1,...,\hat{n}} \lambda_j = 1 & \\
& & & 0 \leq \lambda_j \leq 1/\hat{n} & j = 1, ..., \hat{n}
\end{aligned}
\tag{6.5}
$$

In (6.5), $\hat{n}$ is the number of variables present in the disjunctive term. It is easy to see that x is the convex combination of $\hat{n}$ points that satisfy the constraints $r_{ki}(\hat{x}_j) \leq 0$.

Theorem 6.3.1 *The projection of the feasible region of (6.5) into the original space of x is the convex hull of the feasible region described by $r_{ki}(x) \leq 0$ and $x^{lo} \leq x \leq x^{up}$.*

The proof of Theorem 6.3.1 is trivial. The feasible region of (6.5) describes x as a convex combination of $\hat{n}$ points, each of which satisfy $r_{ki}(x) \leq 0$ and $x^{lo} \leq x \leq x^{up}$. The feasible region of (6.5), projected into the original space, is illustrated in Figure 6.3.a).

Note that in general, (6.5) requires $\hat{n}$ copies of the nonconvex constraints, as well as $\hat{n}$ additional bilinear terms. However, it is possible to use only a subset of the variables in some cases. Furthermore, in some cases it is possible to use an explicit description of the convex hull through generating sets. For such cases, (6.5) can be reformulated as a convex NLP. The theory behind such descriptions is out of the scope of this thesis. We refer the reader to the work by Khajavirad and Sahinidis[72] for details. Also note that the constraint $\lambda_1 \geq \lambda_2 \geq ...$ can be included in (6.5) to eliminate symmetric solutions.

Since the projection of the feasible region of (6.5) into the original space is convex, the inequality (6.4) is also a valid cut if $\tilde{x}_{ki}^P$ is set as the value of x at the optimal solution of (6.5). The cut generated using separation problem (6.5) is illustrated in Figure 6.3.b).

Separation problem (6.5) provides a tool for generating linear cuts that separate a point x^*

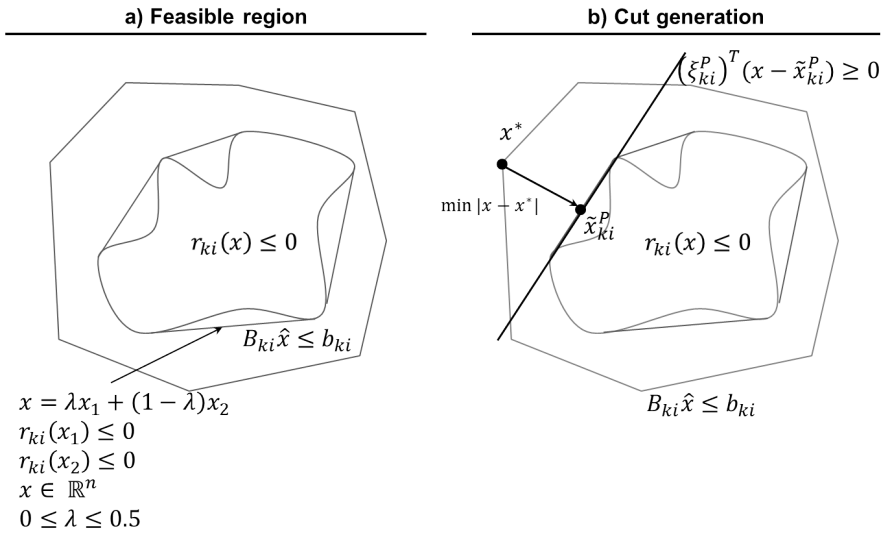

Figure 6.3: Illustration of the feasible region and cuts generated for $r_{ki}(x) \leq 0$, $B_{ki}\hat{x} \leq b_{ki}$, and the feasible region of (6.5); projected into the original space.

from the convex hull of the feasible region of a selected disjunctive term ($Y_{ki} = True$). The main downside of (6.5) is that it is nonconvex since it involves $r_{ki}(\hat{x}_j)$ that can be nonconvex, and the bilinear terms in $x = \sum_{j=1,\dots,\hat{n}} \lambda_j \hat{x}_j$. Furthermore, in order to obtain a valid cut it is necessary to solve (6.5) to global optimality. Even though (6.5) can be difficult to solve, it is important to consider that: a) (6.5) is solved for an individual disjunctive term, which normally involves a small fraction of the total number of constraints and variables of the original problem; and b) the cut is not necessary for the convergence algorithm, it only helps to provide better lower bounds (so it is possible to try to obtain the cuts only for a short period of time).

In order to improve the global logic-based outer-approximation using (6.5), the master

problem of the basic global logic-based outer-approximation is modified as follows:

$$
\begin{aligned}
& min\ Z \\
s.t.\quad & Z \geq c^T x \\
& A\hat{x} \leq a \\
& \bigvee_{i \in D_k} \begin{bmatrix} Y_{ki} \\ y_{ki} = 1 \\ B_{ki}\hat{x} \leq b_{ki} \\ \left(\xi_{ki}^P\right)^T \left(x - \tilde{x}_{ki}^P\right) \geq 0\ \forall p \in CC_{ki} \end{bmatrix} && k \in K \\
& \veebar_{i \in D_k} Y_{ki} && k \in K \\
& \sum_{i \in D_k} y_{ki} = 1 && k \in K \\
& \Omega(Y) = True \\
& Z \geq (Z^p - LB)\left(1 - \sum_{y_{ki}^p = 1}(y_{ki}) - \sum_{y_{ki}^p = 0}(1 - y_{ki})\right) + LB && p \in FS \\
& \sum_{y_{ki}^p = 0}(y_{ki}) + \sum_{y_{ki}^p = 1}(1 - y_{ki}) \geq 0 && p \in IS \\
& \hat{x}^{lo} \leq \hat{x} \leq \hat{x}^{up} \\
& \hat{x} \in \mathbb{R}^{n+s} \\
& 0 \leq y_{ki} \leq 1 && k \in K, i \in D_k \\
& Y_{ki} \in \{True, False\} && k \in K, i \in D_k
\end{aligned}
\tag{MP2}
$$

where $\left(\xi_{ki}^P\right)^T \left(x - \tilde{x}_{ki}^P\right) \geq 0$ are the cuts that were generated for that disjunctive term using (6.5). The feasible region of (MP2) is illustrated in Figure 6.4.

Considering that (6.5) may be difficult to solve but that the cuts are not necessary for the convergence of the algorithm, the basic global logic-based outer-approximation can be modified as follows:

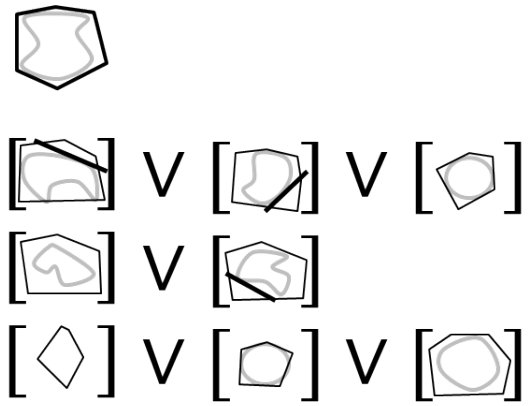

Figure 6.4: Illustration of the feasible region of (MP2).

0. Initialize. Let $LS = IS = CC_{ki} = \emptyset$. Set $\epsilon_1 \geq \epsilon_2 > 0$. Set $P = 1$ and $UB = \infty$. Set $\tau_{sep-limit}$.

1. Solve master problem. Solve (MP2). Let $(Z^*, \hat{x}^*, Y^*, y^*)$ be the optimal solution of (MP) where $\hat{x}^* = (x^*, x^*_{aux})$. Set $y^P = y^*$, $Y^P = Y^*$ and $LB = Z^*$.

2. Find cuts. For every $Y^P_{ki} = True$ that involves nonconvex terms, solve (6.5) with time limit $\tau_{sep-limit}$.

If (6.5) solves to proven global optimality and $|x - x^*|^2 > 0$, let $\tilde{x}^P_{ki}$ be the value of x at the optimal solution of (6.3), and $\xi^P_{ki} = 2(\tilde{x}^P_{ki} - x^*)$. Set $P \in CC_{ki}$.

3. Solve subproblem. Solve (SP), with fixed Y^P, to ϵ_2-global optimality.

If (SP) is feasible, let (Z^*, x^*) be the optimal solution of (SP). Let Z^P be a lower bound for the objective function, provided by the NLP global solution method, and set $P \in FS$. If $Z^* < UB$, let $UB = Z^*$ and $(\bar{Z}, \bar{x}, \bar{Y}) = (Z^*, x^*, Y^P)$.

If (SP) is infeasible, set $P \in IS$.

4. Terminate. If $(UB - LB)/UB \leq \epsilon_1$, terminate with optimal solution $(\bar{Z}, \bar{x}, \bar{Y})$. Else, set $P = P + 1$ and go to step 1.

Note that both phases in the two-phase algorithm can include the cuts in the same manner.

Also note that the cuts could be included in the NLP subproblem to help the global solvers find the optimal solutions faster. From computational experiments we observed that the cuts do not help in reducing the solution time of the subproblem. Finally, note that no cutting planes are included in the global constraints in the described algorithm. If needed, the global constraints (or a subset of the global constraints) can be considered as disjunction with a single term, and the described algorithm would generate the corresponding valid cutting planes.

6.3.3 Illustrative example

We illustrate the algorithm with the following simple analytical example:

$$
\begin{aligned}
& min\ 5 + 0.2x_1 - x_2 \\
s.t.\ & \\
& \begin{bmatrix} Y_{11} \\ x_2 \leq 0.4exp(x_1/2) \\ x_2 \leq 0.5(x_1 - 2.5)^2 + 0.3 \\ x_2 \leq 6.5/(x_1/0.3 + 2) + 1 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ x_2 \leq 0.3exp(x_1/1.8) \\ x_2 \leq 0.7(x_1/1.2 - 2.1)^2 + 0.3 \\ x_2 \leq 6.5/(x_1/0.8 + 1.1) \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ x_2 \leq 0.9exp(x_1/2.1) \\ x_2 \leq 1.3(x_1/1.5 - 1.8)^2 + 0.3 \\ x_2 \leq 6.5/(x_1/0.8 + 1.1) \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ x_2 \leq 0.4exp(x_1/1.5) \\ x_2 \leq 1.2(x_1/ - 2.5)^2 + 0.3 \\ x_2 \leq 6/(x_1/0.6 + 1) + 0.5 \end{bmatrix} \\
& Y_{11} \veebar Y_{12} \\
& Y_{21} \veebar Y_{22} \\
& 0 \leq x_1 \leq 5 \\
& 0 \leq x_2 \leq 3 \\
& Y_{11}, Y_{12}, Y_{21}, Y_{22} \in \{True, False\}
\end{aligned}
\tag{6.6}
$$

The feasible region of the example is presented in Figure 6.5. The figure shows: (a) the

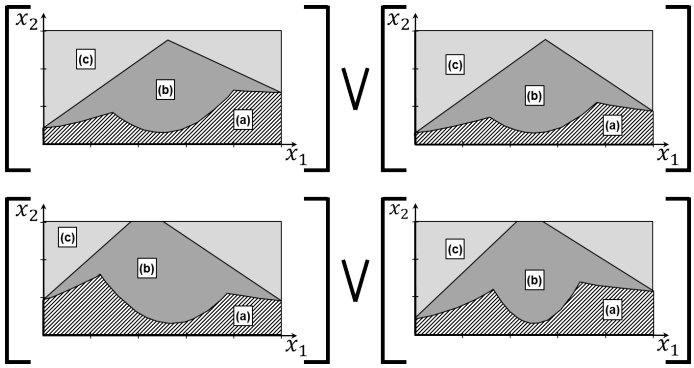

Figure 6.5: Illustration of the feasible region of example (6.6): (a) original GDP; (b) linear relaxation using polyhedral envelopes; (c) linear relaxation dropping nonlinear constraints.

feasible region of the original GDP; (b) the feasible region of the linear relaxation using polyhedral envelopes; and (c) the feasible region of the linear relaxation if the nonlinear constraints are dropped (i.e. only the variable bounds are considered).

The optimal objective value is 4.46 with $Y_{11} = Y_{22} = True$ and $(x_1, x_2) = (1.47, 0.83)$.

The two-phase algorithm is a basic modification of the algorithm. Furthermore, for a very small problem such as (6.6), limiting the solution time of the NLP subproblem makes no difference. For this reason, we illustrate the algorithm and derivation of cutting planes with a one-phase algorithm.

0. Initialize. Let $LS = IS = CC_{ki} = \emptyset$. $\epsilon_1 = 0.1; \epsilon_2 = 0.005$. Set $P = 1$ and $UB = \infty$. Set $\tau_{sep-limit}$.

1. Solve master problem. For the master problem, we consider the most basic type of linear relaxation (i.e. dropping all the constraints that involve nonlinear terms). Since the bound of the variables are still part of the problem, consider the following master problem:

$$
\begin{aligned}
& min\ Z \\
& s.t.\ Z \geq 5 + 0.2x_1 - x_2 \\
& \begin{bmatrix} Y_{11} \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 3 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 3 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 3 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 3 \end{bmatrix} \\
& Y_{11} \veebar Y_{12} \\
& Y_{21} \veebar Y_{22} \\
& 0 \leq x_1 \leq 5 \\
& 0 \leq x_2 \leq 3 \\
& Y_{11}, Y_{12}, Y_{21}, Y_{22} \in \{True, False\}
\end{aligned}
\tag{6.7}
$$

The optimal solution of (6.7) is $(Z^*, x_1^*, x_2^*) = (2, 0, 3)$ with $y_{11}^1 = y_{22}^1 = 1; y_{12}^1 = y_{21}^1 = 0$. $Y_{11}^1 = Y_{22}^1 = True$ and $LB = 2$.

2. Find cuts. For $Y_{11}^P = True$ the following separation problem is obtained:

$$
\begin{aligned}
& min\ (x_1 - 0)^2 + (x_2 - 3)^2 \\
s.t.& \\
& x_1 = \lambda x_{11} + (1-\lambda) x_{12} \\
& x_2 = \lambda x_{21} + (1-\lambda) x_{22} \\
& x_{21} \leq 0.4 exp(x_{11}/2) \\
& x_{21} \leq 0.5(x_{11} - 2.5)^2 + 0.3 \\
& x_{21} \leq 6.5/(x_{11}/0.3 + 2) + 1 \\
& x_{22} \leq 0.4 exp(x_{12}/2) \\
& x_{22} \leq 0.5(x_{12} - 2.5)^2 + 0.3 \\
& x_{22} \leq 6.5/(x_{12}/0.3 + 2) + 1 \\
& 0 \leq x_{11}, x_{12} \leq 5; \quad 0 \leq x_{21}, x_{22} \leq 3; \quad 0 \leq \lambda \leq 1
\end{aligned}
\tag{6.8}
$$

The global optimal solution of (6.8) is $(\tilde{x_1}^1_{11}, \tilde{x_2}^1_{11}) = (0.670, 0.587)$ with value of objective function **6.27**. With these values: $\xi^1_{11} = [1.34, -4.825]^T$. Set $CC_{11} = \{1\}$.

The following valid cut for the term corresponding to Y_{11} is obtained: $1.34(x_1 - 0.670) - 4.83(x_2 - 0.587) \geq 0$. Figure 6.6 shows the cut obtained by solving 6.8. Figure 6.6.a) shows the cut and the nonconvex region of the disjunctive term. Note that the cut is generated based on the feasible region of the disjunctive term, and not based on individual constraints. Figure 6.6.b) shows the linear relaxation before the cutting plane (only the variable bounds). Figure 6.6.c) shows the relaxation after applying the cutting plane. Note that the feasible region after including this cut is different from the feasible region of the linear relaxation using polyhedral envelopes, presented in Figure 6.5.

For $Y_{22}^P = True$ the separation problem is also solved and the following cut is obtained: $1.99(x_1 - 0.994) - 4.28(x_2 - 0.862) \geq 0$.

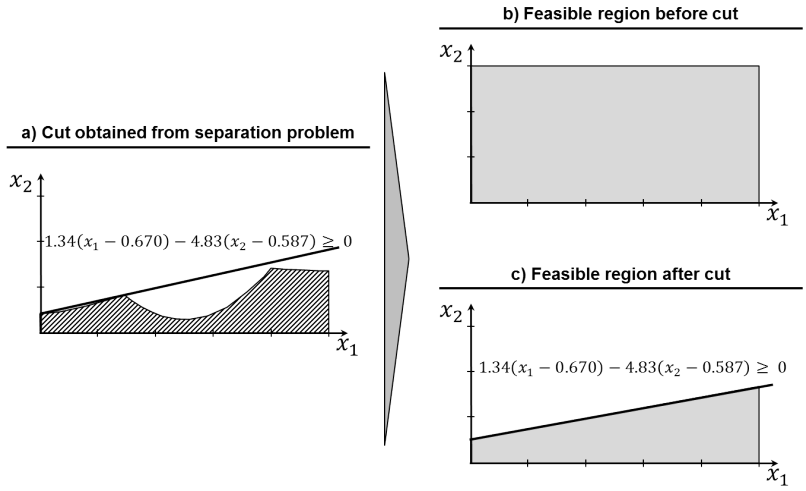

Figure 6.6: Illustration of the feasible region of the term corresponding to Y_{11}, before and after the cut obtained by solving (6.8).

3. Solve subproblem. The following NLP is solved to global optimality:

$$\begin{aligned} & min\ 5 + 0.2x_1 - x_2 \\ s.t. & \\ & x_2 \leq 0.4exp(x_1/2) \\ & x_2 \leq 0.5(x_1 - 2.5)^2 + 0.3 \\ & x_2 \leq 6.5/(x_1/0.3 + 2) + 1 \\ & x_2 \leq 0.4exp(x_1/1.5) \\ & x_2 \leq 1.2(x_1/ - 2.5)^2 + 0.3 \\ & x_2 \leq 6/(x_1/0.6 + 1) + 0.5 \quad 0 \leq x_1 \leq 5 \\ & 0 \leq x_2 \leq 3 \end{aligned} \tag{6.9}$$

(6.9) is feasible with $(Z^*, x_1^*, x_2^*) = (4.46, 1.467, 0.833)$. The lower bound provided by the global solver is $Z^P = 4.44$. Set $1 \in FS$, $UB = 4.46$ and $(\bar{Z}, \bar{x}_1, \bar{x}_2) = (4.46, 1.467, 0.833)$, with $\bar{Y}_{11} = \bar{Y}_{22} = True$.

4. Terminate?. $(UB - LB)/UB = (4.46 - 2)/4.46 = 0.55 \geq \epsilon_1$. Set $P = 2$ and go to step 1.

1. Solve master problem. The new master problem (including the variables y to simplify the presentation of the no-good-cut) is as follows:

$$
\begin{aligned}
& min\ Z \\
& s.t.\ Z \geq 5 + 0.2x_1 - x_2 \\
& \begin{bmatrix} Y_{11} \\ y_{11} = 1 \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 3 \\ 1.34(x_1 - 0.670) - 4.83(x_2 - 0.587) \geq 0 \end{bmatrix} \vee \begin{bmatrix} Y_{12} \\ y_{12} = 1 \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 3 \end{bmatrix} \\
& \begin{bmatrix} Y_{21} \\ y_{21} = 1 \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 3 \end{bmatrix} \vee \begin{bmatrix} Y_{22} \\ y_{22} = 1 \\ 0 \leq x_1 \leq 5 \\ 0 \leq x_2 \leq 3 \\ 1.99(x_1 - 0.994) - 4.28(x_2 - 0.862) \geq 0 \end{bmatrix} \\
& Z \geq (4.44 - 2)(1 - y_{12} - y_{21} - (1 - y_{11}) - (1 - y_{22})) + 2 \\
& Y_{11} \veebar Y_{12} \\
& Y_{21} \veebar Y_{22} \\
& y_{11} + y_{12} = 1 \\
& y_{21} + y_{22} = 1 \\
& 0 \leq x_1 \leq 5 \\
& 0 \leq x_2 \leq 3 \\
& Y_{11}, Y_{12}, Y_{21}, Y_{22} \in \{True, False\} \\
& 0 \leq y_{11}, y_{12}, y_{21}, y_{22} \leq 1
\end{aligned}
\tag{6.10}
$$

Table 6.1: Performance of the algorithm with and without cutting planes for the illustrative example.

	Algorithm with cutting planes		Algorithm without cutting planes	
Iteration	LB	UB	LB	UB
1	2	4.46	2	4.46
2	2	4.46	2	4.46
3	**4.21**	**4.46**	2	4.46
4	-	-	2	4.46
5	-	-	**4.44**	**4.46**

The optimal solution is $(Z^*, x_1^*, x_2^*) = (2, 0, 3)$ with $y_{12}^2 = y_{21}^2 = 1; y_{11}^2 = y_{22}^2 = 0$. $Y_{12}^2 = Y_{21}^2 = True$ and $LB = 2$.

2. Find cuts.

For $Y_{12}^P = True$ (6.5) is solved and the following cut is obtained: $1.27(x_1 - 0.635) - 5.08(x_2 - 0.459) \geq 0$.

For $Y_{21}^P = True$ (6.5) is solved and the following cut is obtained: $1.79(x_1 - 0.896) - 5.19(x_2 - 1.402) \geq 0$.

3. Solve subproblem. The subproblem is fixed for $Y_{12}^2 = Y_{21}^2 = True$. The problem is feasible with $(Z^*, x_1^*, x_2^*) = (4.59, 1.586, 0.724)$. The lower bound provided by the global solver is $Z^P = 4.57$. Set $2 \in FS$.

4. Terminate? $(UB - LB)/UB = (4.46 - 2)/4.46 = 0.55 \geq \epsilon_1$. Set $P = 3$ and go to step 1.

In the next iteration the master problem yields a lower bound of 4.21 and $Y_{12}^2 = Y_{21}^2 = True$. Since $(UB - LB)/UB = (4.46 - 4.21)/4.46 = 0.06 \leq \epsilon_1$ the algorithm terminates in the third iteration. Note that the algorithm, without the cutting planes, would require two more iterations to finish (i.e. it requires to evaluate all of the alternatives of the problem). Table 6.1 summarizes the performance of the algorithm, with and without cutting planes, for the illustrative example.

6.4 Numerical examples and results

In this section we present three examples: layout-optimization of screening systems, superstructures involving reactors and separation units, and design of a distillation column for the separation of benzene and toluene with ideal equilibrium. The first two examples are tested with 20 instances each. The parameters in these instances were created randomly. However, the structure of the problem and constraints represent the actual operation of ideal units. The last example uses thermodynamic data, equilibrium relations, etc. All of the instances were solved using GAMS 24.3.3[21], using an Intel(R) Core(TM) i7 CPU 2.93 GHz and 4 GB of RAM. For comparison, all instances were formulated as MINLP using the BM reformulation and solved with BARON 14.0.3[53]. It was not possible to accurately compare with ANTIGONE 1.1[51] and SCIP 3.1[24]. The former returned "infeasible" in instances to which known solutions existed, and with the latter the computer ran out of memory in several instances. All of the variable bounds in these problems are defined.

The algorithm stays in the first phase for 20% of the time limit (which is 7,200 seconds in all instances or if there is no improvement in the best known solution after 50 iterations. The weight parameter (W) in the first phase is calculated as follows: $W = LB/((P-1)(\sum_{k\in K} D_k))$. The idea behind this weight value is that the penalty for diversifying solutions will never be greater than the value of the lower bound. In the extreme in which every single binary variable is different from all previously evaluated solutions (which can only happen in a very specific situation), then the penalty function has exactly the same value as LB. The time limit for generating cuts in each selected term is 10 seconds in the first phase and 120 seconds in the second phase. The time limit for the subproblem in the second phase is 10 seconds for the first two examples and 60 seconds for the design of the distillation column. All of the master problems were solved by reformulating the linear GDP as MILP using the BM reformulation and using CPLEX 12.6.0.1[40]. Note that, because in these instances the master problem is much simpler than the subproblems, using BM or HR reformulation in the master problem has little impact on the performance of the algorithm. The subproblem and separation problem were solved with BARON 14.0.3.

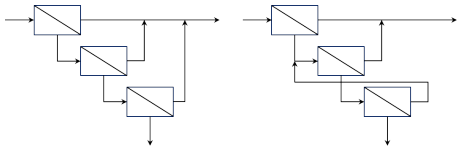

Figure 6.7: Two alternative configurations for a three stage screening system.

6.4.1 Layout-optimization of screening systems in recovered paper production

This problem is a GDP representation of the MINLP presented by Fügenschuh et al.[73]. The problem seeks to optimize the layout of multi-stage-screening systems, in order to separate the impurities (stickies) from the paper pulp. In addition to optimizing the configuration, the problem presented in this section also among alternative units with different rejection and cost coefficients. Figure 6.7 illustrate two alternative configurations for a screening system with three units. The GDP formulation of this problem is presented in C. The nonconvexities arise in the cost constraints and in the relationship of the separation efficiency and the rejection rate.

Figure 6.8 presents the performance of BARON and the algorithm. The plot shows the average relative bound (upper and lower) for 20 instances vs. time. The relative upper (lower) bound is obtained by dividing the upper (lower) bound by the best known solution to that instance. If there is no solution found for an instance, the relative upper bound was set to 5. In this figure, the linearization in master problem of the algorithm is performed by using polyhedral envelopes of the nonconvex functions. Figure 6.8 shows that, on average, the algorithm finds slightly better solutions. However, BARON is better at finding good solutions faster and provides a slightly better lower bound in average for this problem.

Figure 6.9 presents the performance of the basic logic-based outer-approximation, and how it improves with the different enhancements. In this analysis, the linear GDP in the master problem is obtained through polyhedral envelopes. Similar to Figure 6.8, the plot shows the average relative bound (upper and lower) for 20 instances vs. time. Figure 6.9.a) shows the improvements when the algorithm is divided into two phases, and it does

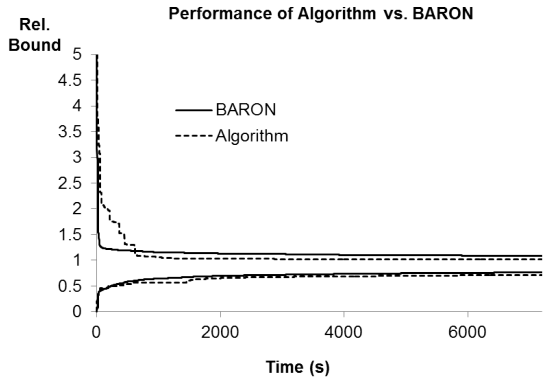

Figure 6.8: Performance of BARON and the complete algorithm, for layout-optimization of screening systems, using polyhedral envelopes for the linear relaxation in the master problem.

not include the cuts. The plot shows that dividing the algorithm into two phases helps to find feasible solutions faster. Furthermore, by including the penalty term in the objective function the algorithm finds good solutions slightly faster than without it. As expected, the two-phase algorithm and penalty function (6.2) have no significant impact in the lower bound. Figure 6.9.b) presents the performance of the algorithm with and without the cuts. The plot shows that the cuts slightly improve the lower bound. It also shows a very small improvement in the upper bound. Although the cuts are not intended to improve the upper bound, by having a better linear GDP representation in the master problem the algorithm selects better alternatives to evaluate in the subproblem.

Figure 6.10 also presents the performance of the algorithm with and without cuts. However, in this case the linear GDP in the master problem was obtained by dropping the nonlinear constraints (instead of using polyhedral envelopes). The lower bound of the problem is zero and does not change when using the algorithm without cutting planes. If the cutting planes are used in the algorithm, then the average lower bound increases considerably, up to 0.6 relative to the best known solution. It can also be observed from the figure that the upper bound improves as well. The reason for this is that by having a more accurate linear representation, the master problem selects better alternatives to evaluate in the subproblem. Note that the lower bound improves considerably at around 1,400

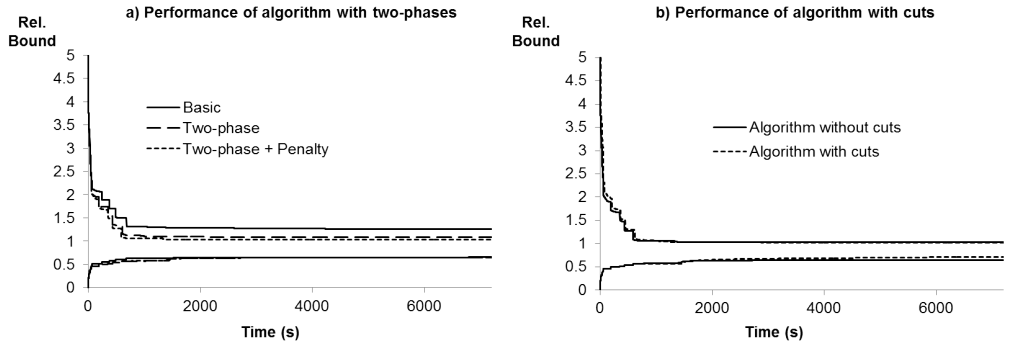

Figure 6.9: Performance of the algorithm, for layout-optimization of screening systems with, a) two-phase enhancement, and b) cutting planes.

seconds. The main reason is that at around this time the algorithm moves from the first phase to the second phase. This means that: a) the penalty function for diversification in the first phase does not allow much improvement in the lower bound; and b) the time limit for generating cuts is 10 seconds in the first phase (vs. 120 seconds in the second one) so several additional cuts are generated in the second phase.

6.4.2 Reactor-separator process superstructure

In this problem a set of reactors and separation units are given, as well as different material sources and product demands. The problem seeks to minimize the cost of satisfying that demand. Any equipment can be selected, and any interconnections between potential equipment is allowed. In this example, the product is component C and the raw materials are component A and B. Each source has a different concentration of A and B. A is the most volatile component, then B, and then C. The reactors follow second order kinetics $(A + B \rightarrow C)$, and the kinetic and cost parameters are random variables. The separation is assumed to be sharp (e.g. it separates A from B and C). It is assumed that the two outer streams of the separation units have the same total molar concentration as the inlet stream. An illustration of a process superstructure with three potential sources, two reactors and two separating units is presented in Figure 6.11.a). For illustration purposes, the figure does not show the interconnection of units from and to the separation units.

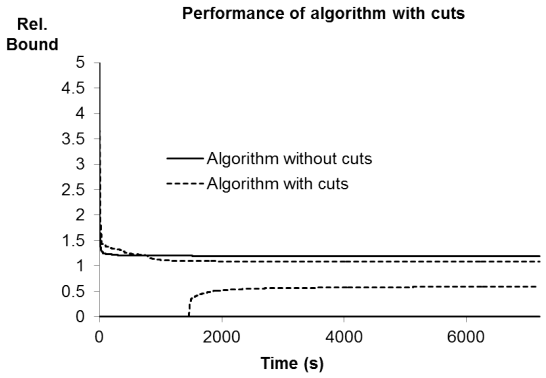

Figure 6.10: Performance of the algorithm,, for layout-optimization of screening systems, with and without cutting planes when dropping the nonlinear constraints in the master problem.

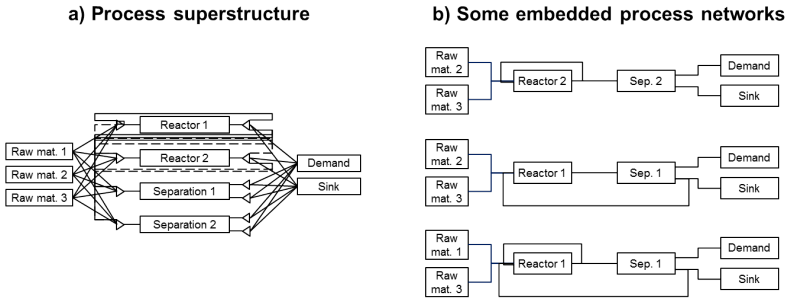

Figure 6.11: Illustration of process superstructure with two reactors and two separators.

The interconnections are in fact allowed in the general formulation. This simple example contains several process networks embedded as presented in Figure 6.11.b). The problem formulation is presented in Appendix C.

Figure 6.12 presents the performance of BARON and the algorithm for the process superstructure. 20 instances of this problem were tested. If there is no solution found for an instance it was averaged as a relative upper bound of 5. In this figure, the linearization in master problem of the algorithm is performed by using polyhedral envelopes of the nonconvex functions. Figure 6.12 shows that the proposed algorithm performs much better than BARON. In particular, the average relative lower bound is very similar for both

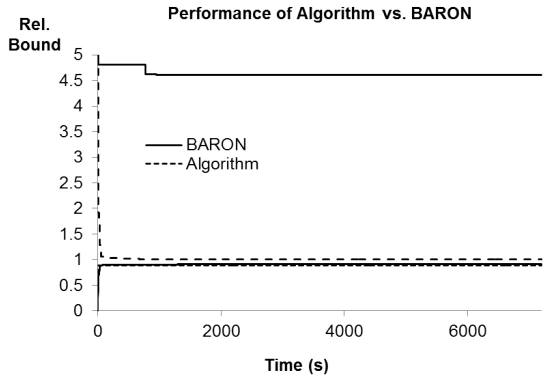

Figure 6.12: Performance of BARON and the complete algorithm, for reactor-separator process superstructure, using polyhedral envelopes for the linear relaxation in the master problem.

methods, but the upper bound (i.e. the finding of feasible solutions) is much better for the enhanced GLBOA. The reason for this is that BARON is able to find a feasible solution in only 2 of the 20 instances; therefore the average relative upper bound is almost 5. The algorithm finds feasible solutions in every problem, and very close to the lower bound in most cases.

Figure 6.13 presents the performance of the different versions of the logic-based outer-approximation for the 20 instances of this problem. In this analysis, the linear GDP in the master problem is obtained through polyhedral envelopes. Figure 6.13.a) shows the improvements when the algorithm is divided into two phases, and it does not include the cuts. In this example, the two-phase algorithm does not show much improvement when compared to the basic GLBOA. However, by including the penalty in the objective function the two-phase algorithms improves drastically. The main reason for this is that the discrete solutions provided by the master problem are not good when the subproblem is evaluated. By including the penalty function, the solutions are diversified and the algorithm is able to find much better solutions. Figure 6.13.b) shows the impact of using the cutting planes. In this example, the cutting planes do not help to improve the lower bound since the polyhedral envelopes already provide a very good linear approximation (as can be observed from the lower bounds in Figures 6.12 and 6.13).

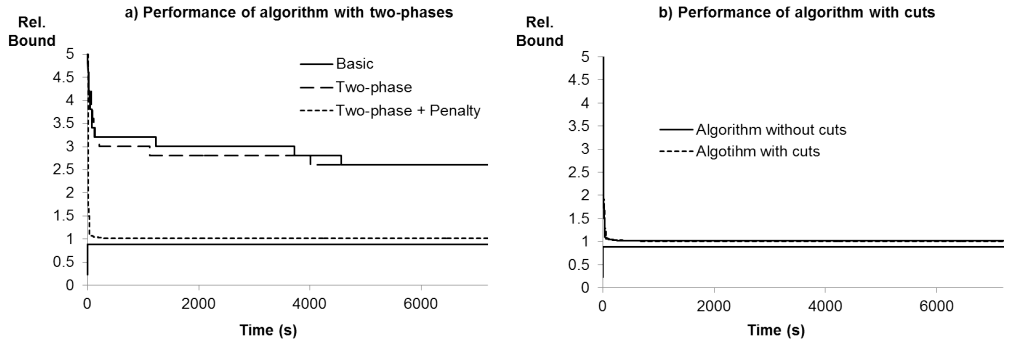

Figure 6.13: Performance of the algorithm, for reactor-separator process superstructure with, a) two-phase enhancement, and b) cutting planes.

Figure 6.14 presents the performance of the algorithm with and without cuts, dropping the nonlinear constraints to generate the master problem (instead of using polyhedral envelopes). It can be observed that the average relative lower bound improves considerably by including the cutting planes. Without the cutting planes, the average relative lower bound is about 0.6. With the cutting planes, the algorithm improves and provides a lower bound of 0.9. It can be seen in this plot that the major improvement happens at around 1,400 seconds, which is about the time limit for the first phase.

6.4.3 Design of distillation column for the separation of benzene and toluene with ideal equilibrium

This problem is presented by Yeomans and Grossmann[74]. The objective is to design a distillation column for the separation of Benzene and Toluene, assuming ideal equilibrium. The feed to the column has a composition of 100 kmol/h of benzene and 50 kmol/h of toluene. The required purity for the product is 99% benzene in the overhead and a minimum recovery of 50%. The GDP model uses a tray by tray representation of the distillation column. The separation is carried out at 1.01 bar. Figure 6.15 illustrates the idea of the GDP formulation. In the model, there are three trays that are fixed, so the MESH (Mass-Equilibrium-Summation-Heat) constraints for this plates are enforced (i.e. they are global constraints). The MESH constraints corresponding to the plates in the rectification

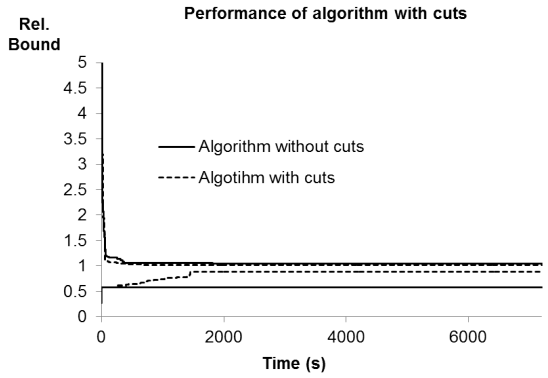

Figure 6.14: Performance of the algorithm, for reactor-separator process superstructure, with and without cutting planes when dropping the nonlinear constraints in the master problem.

and stripping sections are conditional: if a tray is installed then MESH constraints are enforced; if it is not selected then the trays are simply a bypass. For details on the MESH equations, we refer the reader to the original model[74]. The MINLP reformulation of this problem using the BM has 3,257 constraints, 1,758 variables and 64 discrete variables.

For this problem, BARON 14.0 is not able to find a feasible solution and provides a lower bound of -119. For the algorithm, the master problem was obtained by using polyhedral envelopes for the linearizations. Figure 6.16 shows the performance of the algorithm with the different enhancements. This figure shows the progress of the upper and lower bounds with time (not relative upper and lower bounds as presented in the previous figures). From the figure, it can be observed that the lower bound is quite weak in all cases, and Figure 6.16.b) shows that the cutting planes do not help to obtain better bounds. The main reason for this is that the separation problems in this example take a long time, so only 5 cuts were generated by the algorithm. These cuts did not have any effect on improving the relaxation. On the other hand, it is clear that the upper bound improves considerably with the enhancements. In particular, neither BARON nor the basic LBOA can find a feasible solution within two hours. In contrast, the two-stage algorithm finds a feasible solution of 268 after 244 seconds; a good solution (within 10% of the best known solution) of 80.6 after 2,200 seconds; and the best solution it finds is 73.9 after 5,640 seconds. The two-

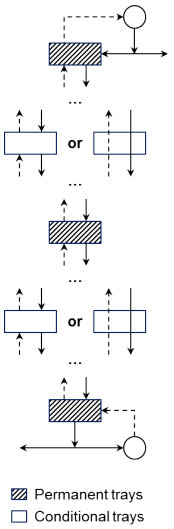

Figure 6.15: Illustration of the GDP model representation for tray by tray design of distillation columns.

stage algorithm with a penalty function for diversification finds a feasible solution of 85 after 122 seconds; a good solution of 77 after 980 seconds; and the best solution it finds is 74.3 after 5,815 seconds. With cutting planes, the algorithm finds a feasible solution of 85 after 130 seconds; a good solution of 77 after 830 seconds; and the best solution it finds is 74.0 after 5,680 seconds.

6.5 Conclusions

In this chapter, we have presented a basic global logic-based outer-approximation method for nonconvex GDP problems and improved it with two enhancements. The algorithm was tested with three examples, 20 random instances of each of the first two and one instance of the last one. The first enhancement improves finding feasible solutions by partitioning the algorithm into two phases and diversifying the search of feasible solutions in the first phase. The partition of the algorithm considerably improves finding good solutions in the layout-optimization of screening systems and in the design of a distillation column. In the former, the basic algorithm finds an average relative upper bound of 1.3 and it improves to

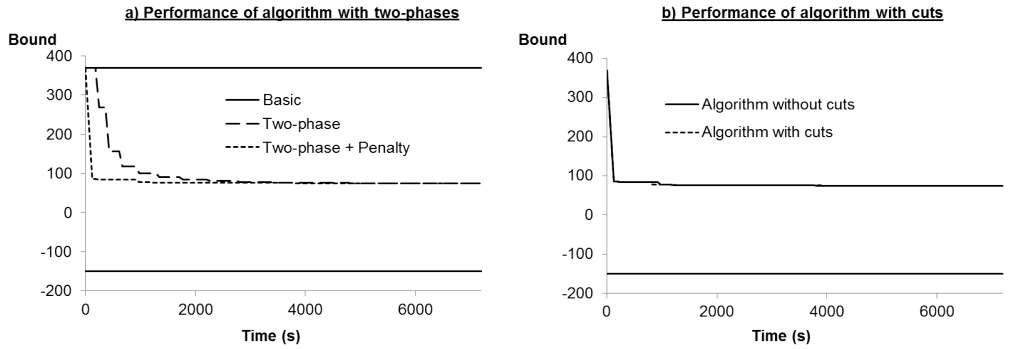

Figure 6.16: Performance of the algorithm, for reactor-separator process superstructure, with a) Two-phase enhancement and b) cutting planes.

1.1 with the two phases. In the latter, the algorithm improves from not finding a feasible solution to finding the best known solution. The search of feasible solutions is further enhanced with the use of a penalty in the objective function that diversifies search. This strategy is useful in the three problems, and the results show a speed up in the finding of good feasible solutions. Furthermore, in the first example the best found solution also improves from 1.1 average relative upper bound to 1.

The second enhancement is a cutting plane method to improve the lower bounding of the algorithm. If polyhedral envelopes are used in the master problem, considering the tight variable bounds used in the examples, this method is useful only slightly in the second example. However, when the linear relaxation is obtained by dropping the nonlinear terms the method is useful in all instances of the first two examples. This result indicates that for problems in which the linear relaxation is poor the method can derive strong cutting planes that considerably improve the linear approximation. Note that if the bounds of the variables are poor, the polyhedral envelopes will tend to be poor as well. However, the cutting planes obtained through the proposed method depend only on the feasible region of the disjunctive term. This means that, as long as the feasible region is the same, the cutting planes obtained through the method are the same regardless of the variable bounds.

Chapter 7

GLBOA for the global optimization of a source based model of the multiperiod blending problem

7.1 Introduction

Many processes in the petrochemical industry involve the blending of intermediate and final products. Large cost savings can be achieved by efficient blending schemes that satisfy the technical and regulatory specifications of products. For example, the economic and operability benefits from optimal crude-oil blend scheduling can reach multimillion dollars per year[75].

One of the first mathematical programming models to represent the scheduling of blending operations is the pooling problem[76]. The pooling problem seeks to find the optimal blend of materials available from a set of supply streams, while satisfying the demand of a set of products. The model enforces that the end products satisfy a specified minimum and maximum level for each specification. The objective is to minimize the total cost (or maximize the profit) of the operation. Several optimization models for the pooling problem have

been reported in the literature. The p-formulation[76], based on total flows and component compositions, is commonly used in chemical process industries. The q-formulation[77] uses variables based on the fraction that each input stream contributes to the total input to each pool, and does not explicitly use the pool specifications as variables. The pq-formulation[68] is obtained by including valid redundant inequalities in the q-formulation. Tawarmalani and Sahinidis[68] prove that the redundant constraints help to obtain a stronger polyhedral relaxation of the pooling problem. Lastly, Audet et al.[78] propose a hybrid formulation by combining the p and q models to avoid additional bilinear terms that arise when generalized pooling problems are modeled using the q-formulation.

The multiperiod blending problem can be regarded as an extension of the pooling problem. In addition to the pooling problem restrictions, it considers inventory and time variations of supply and demand. The multiperiod blending problem can be formulated as a mixed-integer nonlinear programming (MINLP) problem[79]. Binary variables are required to model the movements of materials in and out of the tanks and to account for fixed costs. Even in the absence of binary variables, bilinear terms (which are necessary to model the mixing of various streams) make the problem nonconvex. Due to this highly combinatorial and nonconvex nature, the blend scheduling problem is very challenging. General purpose global optimization solvers fail to solve even small instances.

Foulds et al.[80] were the first to propose a global optimization algorithm to solve a single-component pooling problem. They use McCormick envelopes[67] to relax the bilinear terms. Androulakis et al.[81] propose a convex quadratic NLP relaxation, known as αBB underestimator. However, due to its generality, the NLP relaxation is weaker than its LP counterpart. Ben-Tal et al.[77] and Adhya et al.[82] present different Lagrangean relaxation approaches for developing lower bounds for the pooling problem. These bounds are tighter than standard LP relaxations used in global optimization algorithms.

In the context of processing network problems, Quesada and Grossmann[83] apply the reformulation-linearization technique (RLT)[84], together with McCormick envelopes, to improve the relaxation of a bilinear program by creating redundant constraints. These authors combine concentration and flow based models in order to obtain a relaxed LP formulation that provides a valid and strong lower bound to the global optimum. Simi-

lar results are obtained by Tawarmalani and Sahinidis[68] for the multicomponent pooling problem. The idea of using redundant constraints to strengthen the relaxation of the original problem is also used by Karuppiah et al.[85] in the context of water networks. These constraints correspond to total mass balance of contaminants and serve as deep cuts in the McCormick relaxation.

Piecewise MILP relaxations are an alternative relaxation of MINLPs that provide stronger bounds than traditional MILP relaxations. The first references to the use of piecewise MILP relaxation are by Bergamini et al.[65] and Karuppiah et al.[85]. Following this idea, Wicaksono and Karimi[86] propose several novel formulations for piecewise MILP under and overestimators for bilinear programs. Gounaris et al.[87] present a comprehensive computational comparison study of a collection of fifteen piecewise linear relaxations over a collection of benchmark pooling problems. Misener et al.[88], building on the ideas from Vielma and Nemhauser[89], introduce a formulation for the piecewise linear relaxation of bilinear functions with a logarithmic number of binary variables. Another alternative to piecewise relaxations are discretization techniques, such as multiparametric disaggregation[90,91]. The number of additional binary variables increases linearly with each increment in the precision of the discretization.

As an alternative to branch-and-bound solution procedures, Kolodziej et al.[90] propose a heuristic as well as a rigorous two-stage MILP-NLP and MILP-MILP global optimization algorithms. Approximate and relaxed MILPs are obtained through the multiparametric disaggregation technique. Kesavan et al.[92] propose two approaches to generalize the outer approximation algorithm to separable nonconvex MINLP. Similarly, Bergamini et al.[65], based on the work from Turkay and Grossmann[18], present a deterministic algorithm based on logic-based outer approximation that can guarantee global optimality in the solution of an optimal the synthesis of process network problem. The global logic-based outer-approximation is presented in chapter 6.

Although the multiperiod blending problem arises in several applications, crude-oil blending is of great importance due to the potential increase in profit derived from optimal operation. In fact, crude-oil costs account for about 80% of the refinery turnover[93]. As a scheduling extension of the blending problem, crude-oil scheduling involves the unloading

of crude marine vessels into storage tanks, followed by the transfer of crude from storage to charging tanks and finally, to the crude-oil distillation units (CDUs)[94,95]. Lately, crude-oil scheduling models incorporate more quantity, quality, and logistics decisions related to real-life refinery operations, such as minimum run-length requirements, one-flow out of blender or sequence-dependent switchovers[96].

Several authors have proposed different algorithms relying on mixed-integer linear formulations to avoid solving the full nonconvex MINLP. These models can be seen as relaxations of the original MINLP. Mendez et al.[97] present a novel MILP-based method where a very complex MINLP formulation is replaced by a sequential MILP approximation that can deal with non-linear gasoline properties and variable recipes for different product grades. Similarly, a two-stage MILP-NLP solution procedure is employed by Jia et al.[98] and Mouret et al.[99], featuring in the first stage a relaxed MILP model without the bilinear blending constraints followed by the solution of the original MINLP after fixing all binary variables. The same two-stage algorithm is studied by Castro and Grossmann[100] together with several global optimization methods. However, instead of dropping the bilinear constraints in the two-stage algorithm, they use multiparametric disaggregation to relax the bilinear terms. Moro and Pinto[101] and Karuppiah et al.[85] tackle the problem with the augmented penalty version and a specialized version of the outer-approximation method, respectively. Reddy et al.[102] propose an MILP relaxation combined with a rolling-horizon algorithm to eliminate the composition discrepancy. Finally, Li et al.[93] use a spatial branch-and-bound global optimization algorithm, that at each node uses the MILP-NLP two-stage strategy previously mentioned, to solve the MINLP problems.

Even though, continuous-time models seem to be preferred for crude-oil scheduling, the demand-driven nature of the multiperiod blending problem makes a simple discrete-time framework a better choice for our problem. Despite the latest modeling and algorithmic advances for this class of problems, large instances are still intractable. Improvements or even new problem formulations and solution approaches must be proposed.

In this chapter, we make two primary contributions for solving multiperiod blending problems. The first is an alternative formulation of the problem, in terms of GDP, that makes use of redundant constraints. These constraints considerably improve the linear GDP re-

laxation of the nonlinear GDP. Based on the observation that one can reduce the complexity of a problem by fixing values of certain variables, a decomposition method is proposed next. The algorithm decomposes the GDP model into two levels. The first level, or master problem, is a linear GDP relaxation of the original GDP that provides rigorous upper bounds. The second level, or subproblem, is a smaller GDP in which some of the Boolean variables of the original problem are fixed. The subproblem, when a feasible solution is found, provides a feasible solution to the original GDP and a rigorous lower bound. These problems are solved successively until the gap between the upper and lower bound is closed. We illustrate the new formulation and decomposition method with several test problems. The results show that the new formulation can be solved faster than the alternatives, and that the decomposition method can solve the problems faster than state-of-the-art general purpose solvers.

Table 7.1: Nomenclature of sets and variables.

Sets	Symbols	Element
Total number of tanks	$\mathcal{N}$	n
Blending tanks	$\mathcal{B} \subset \mathcal{N}$	b
Supply tanks	$\mathcal{S} \subset \mathcal{N}$	s
Demand tanks	$\mathcal{D} \subset \mathcal{N}$	d
Specifications	$\mathcal{Q}$	q
Sources	$\mathcal{R}$	r
Time periods	$\mathcal{T}$	t
Variables	Symbols	Sets
Continuous Variables		
Flow between tanks n and n' at time t	$F_{nn't}$	$(n,n') \in \mathcal{A}, t \in \mathcal{T}$
Demand flow from tanks d at time t	FD_{dt}	$d \in \mathcal{D}$, $t \in \mathcal{T}$
Inventory in tank n at time t	I_{nt}	$n \in \mathcal{N}$, $t \in \mathcal{T}$
Specification q in tank b at time t	C_{qbt}	$q \in \mathcal{Q}$, $b \in \mathcal{B}$, $t \in \mathcal{T}$
Flow of spec. q between tanks n and n' at time t	$\bar{F}_{qnn't}$	$q \in \mathcal{Q}$, $(n,n') \in \mathcal{A}$, $t \in \mathcal{T}$
Inventory of spec. q in blending tank b at time t	$\bar{I}_{qbt}$	$q \in \mathcal{Q}$, $n \in \mathcal{N}$, $t \in \mathcal{T}$
Flow of source r between tanks n and n' at time t	$\tilde{F}_{rnn't}$	$r \in \mathcal{R}$, $(n,n') \in \mathcal{A}$, $t \in \mathcal{T}$
Inventory of source r in blending tank b at time t	$\tilde{I}_{rbt}$	$r \in \mathcal{R}$, $n \in \mathcal{N}$, $t \in \mathcal{T}$
Fraction of inventory in blending tank b sent to tank n at the end of time t	ξ_{bnt}	$(b,n) \in \mathcal{A}$, $t \in \mathcal{T}$
Boolean Variables		
Variable that indicates the existence of flow between tanks n and n' at the end of time t	$X_{nn't}$	$(n,n') \in \mathcal{A}$, $t \in \mathcal{T}$
For blending tank b at time t: $YB_{bt} = True$ if tank is charging. $YB_{bt} = False$ if tank is discharging.	YB_{bt}	$b \in \mathcal{B}$, $t \in \mathcal{T}$
Binary Variables		
0-1 variable corresponding to Y_{bt}. $yb_{bt} = 1$ indicates the tank is charging. $yb_{bt} = 0$ indicates the tank is discharging	yb_{bt}	$b \in \mathcal{B}$, $t \in \mathcal{T}$

Table 7.2: Nomenclature of parameters.

Parameters	Symbols	Sets
Initial inventory for tank n	I_n^0	$n \in \mathcal{N}$
Initial values for the specifications q in tank b	C_{qb}^0	$q \in \mathcal{Q}$, $b \in \mathcal{B}$
Incoming supply flows enter tank s at time t	F_{st}^{IN}	$s \in \mathcal{S}$, $t \in \mathcal{T}$
Specification q in supply flow to tank s	C_{qs}^{IN}	$q \in \mathcal{Q}$, $s \in \mathcal{S}$
Specification q in source r	$\hat{C}_{qr}^0$	$q \in \mathcal{Q}$, $r \in \mathcal{R}$
Bounds on demand flow from tanks d at time t	$[FD_{dt}^{\mathrm{L}}, FD_{dt}^{\mathrm{U}}]$	$d \in \mathcal{D}$, $t \in \mathcal{T}$
Bound on specification q in demand tank d	$[C_{qd}^{\mathrm{L}}, C_{qd}^{\mathrm{U}}]$	$q \in \mathcal{Q}$, $d \in \mathcal{D}$
Bounds on inventory for tank n	$[I_n^{\mathrm{L}}, I_n^{\mathrm{U}}]$	$n \in \mathcal{N}$
Bounds on flow between tank n and n'	$[F_{nn'}^{\mathrm{L}}, F_{nn'}^{\mathrm{U}}]$	$(n, n') \in \mathcal{A}$
Costs for the supply flow for tank s	β_s^T	$s \in \mathcal{S}$
Prices for demand flow for tank d	β_d^T	$d \in \mathcal{D}$
Fixed costs for flow from tank n to tank n'	$\alpha_{nn'}^N$	$(n, n') \in \mathcal{A}$
Variable costs for flow from tank n to tank n'	$\beta_{nn'}^N$	$(n, n') \in \mathcal{A}$

Table 7.3: GDP models.

Model	**Description**
($\mathbb{C}$)	Concentration of individual specifications is a variable. Bilinear terms appear when blending tank is in "charging" mode.
($\mathbb{SF}$)	Flow and inventory of specifications and split fraction are variables. Bilinear terms appear when blending tank is in "discharging" mode.
($\mathbb{SB}$)	Flow and inventory of sources and split fraction are variables. Bilinear terms appear when blending tank is in "discharging" mode.
($\mathbb{CSB}$)	Same as ($\mathbb{C}$), but including redundant constraints from ($\mathbb{SB}$).
($\mathbb{MP}$)	Master problem. Linear relaxation of ($\mathbb{CSB}$), including enumeration cuts.
($\mathbb{SP}$)	Subproblem using ($\mathbb{CSB}$) model with Y_{bt} fixed.

7.2 The Multiperiod Blending Problem

The multiperiod blending problem is defined on a network $(\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of arcs connecting these nodes. The set of nodes is partitioned into three subsets corresponding to the types of tanks: supply nodes $s \in \mathcal{S}$, blending nodes $b \in \mathcal{B}$, and demand nodes $d \in \mathcal{D}$. Directed arcs $(n, n') \in \mathcal{A}$ between nodes correspond to streams from tank n to tank n'. In general, interconnections between the supply and demand tanks, as well as between blending tanks, are allowed by the model. The streams and inventories in the system possess different specifications $q \in \mathcal{Q}$, such as concentration of chemical compounds or physical properties. The network operates over a time horizon composed of multiple time periods, $\mathcal{T} = \{0, 1, \ldots, T\}$, over which demand within concentration specifications has to be satisfied at the end of each time period. Figure 7.1 shows a schematic representation of the blending system.

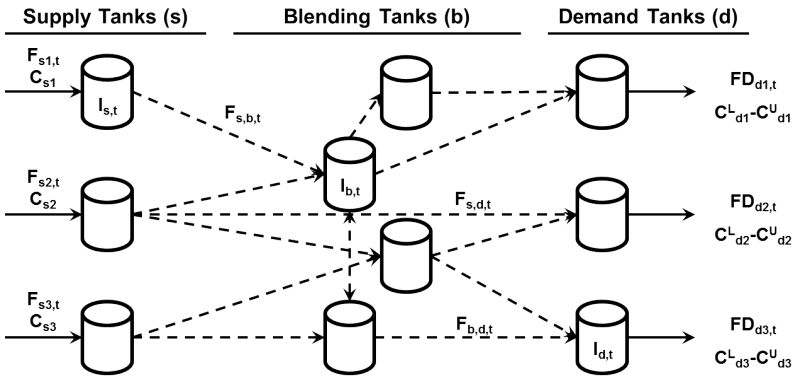

Figure 7.1: Sketch of the multiperiod blending problem

Given fixed feed compositions C_{qs}^{IN} and incoming flows F_{st}^{IN} to the supply tanks, as well as initial conditions in each tank, the problem consists of determining the flows $F_{nn't}$, FD_{dt}, inventories I_{nt} and compositions C_{qbt} in the network in each time period so as to maximize the profit (or minimize the total cost) of the blending schedule, while meeting the demand limits $[FD_{dt}^{\mathrm{L}}, FD_{dt}^{\mathrm{U}}]$ within specified limits of composition $[C_{qd}^{\mathrm{L}}, C_{qd}^{\mathrm{U}}]$.

Note that each time period $t \in \mathcal{T}$ is not independent of the others due to the coupling

created by the inventories[79]. For instance, the composition and flow of an outgoing stream from a blending tank at time t depends on the inventory in that tank at the end of the previous time period, $t-1$. As a consequence, the optimization must be performed simultaneously over all time periods.

For simplicity, the composition of the incoming flow to the supply tanks C_{qs}^{IN} and the bounds on the concentration of flows leaving the demand tanks $[C_{qd}^{\mathrm{L}}, C_{qd}^{\mathrm{U}}]$ are assumed to be constant over the time horizon. As a result, the compositions C_{qbt} in the blending tanks in each time period are the only ones that are unknown in the system (hence the subscript b instead of n in C_{qbt}). On the other hand, the supply and demand flows can vary in amount over time (hence the subscript t in F_{st}^{IN} and $[FD_{dt}^{\mathrm{L}}, FD_{dt}^{\mathrm{U}}]$).

The system operates within bounds on the inventories $[I_n^{\mathrm{L}}, I_n^{\mathrm{U}}]$, and on the flows $[F_{nn'}^{\mathrm{L}}, F_{nn'}^{\mathrm{U}}]$ between each pair of tanks $(n, n') \in \mathcal{A}$.

In order to quantify the profit of the blending process, costs for the supply flows β_s^T, prices for the demand flows β_d^T, and fixed and variable costs $[\alpha_{nn'}^N, \beta_{nn'}^N]$ for the flows within the network are taken into account.

An important assumption is that, due to operational and safety considerations, simultaneous input/output streams to blending tanks is not allowed, i.e. flow cannot enter and exit a blending tank in the same time period. Boolean variables $X_{nn't}$, which represent existence ($X_{nn't} = True$) or absence ($X_{nn't} = False$) of flow between tanks n and n', are required to model this assumption, as well as to represent fixed costs for using the pipelines in the objective function. Additional Boolean variables (YB_{bt}) are used to represent the operating mode of a blending tank ($YB_{bt} = True$ if a tank is "charging" and $YB_{bt} = False$ if a tank is "discharging"). Finally, the multiple liquid streams that enter the blending tanks are assumed to be perfectly mixed at the end of the time period.

Tables 7.1 and 7.2 contain a detailed explanation of the nomenclature used for sets, variables and data in the problem. It can be noted from these tables that parameters contain a superscript and variables do not. Table 7.3 contains the different models presented in this chapter, as well as a brief description with the main difference among them.

7.2.1 Motivating Example

In this section we present a small illustrative example to provide some insight on the complexity of multiperiod blending problems. It should be noted that the instance is significantly simple so that the solution can in fact be obtained by inspection. The instance consists of 2 supply tanks, 8 blending tanks, 2 demand tanks, 6 time periods and 1 specification. The topology of the network is shown in Figure 7.2.

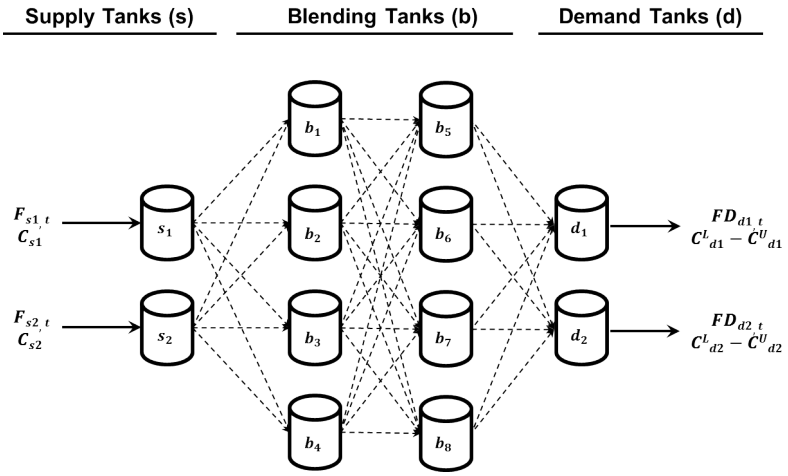

Figure 7.2: Topology of the motivating example

Tables 7.4 and 7.5 contain the parameters of the supply and demand streams. The initial inventory and concentration are zero for all blending tanks, $I_b^0 = C_{qn}^0 = 0 \quad b \in \mathcal{B},\ q \in \mathcal{Q}$. There is no inventory capacity in supply and demand tanks ($I_s^{\mathrm{U}} = I_d^{\mathrm{U}} = 0 \quad s \in \mathcal{S},\ d \in \mathcal{D}$). The maximum inventory in the blending tanks is 30 for the first row of tanks ($I_b^{\mathrm{U}} = 30 \quad b \in \{1, 2, 3, 4\}$) and 20 for the second row of tanks ($I_b^{\mathrm{U}} = 20 \quad b \in \{5, 6, 7, 8\}$). The maximum flow between tanks is 30 ($F_{nn'}^{\mathrm{U}} = 30 \quad (n, n') \in \mathcal{A}$). The fixed cost for using the pipelines of 0.1 ($\alpha_{nn'}^{N} = 0.1 \quad (n, n') \in \mathcal{A}$).

Note the rigid structure of the instance. The sum of the supply flow over the time horizon equals the demand. Since the initial inventory is zero, all the supply should be used to satisfy the demand, thus all blending tanks will be empty at the end of the time horizon.

Table 7.4: Supply tank specifications.

	C_s^{IN}	F_{st}^{IN}						
Supply tank	Qual.A	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	β_s^T
s_1	0.06	10	10	10	0	0	0	0
s_2	0.26	30	30	30	0	0	0	0

Table 7.5: Demand tank specifications.

	$[C_{qd}^L, C_{qd}^U]$	FD_{dt}^L						
Demand tank	Qual.A	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	β_d^T
d_1	$[0, 0.16]$	0	0	15	15	15	15	2
d_2	$[0, 1]$	0	0	15	15	15	15	1

Besides, the supply with low concentration of specification A should be equally mixed with flow from supply s_2 in order to satisfy the specifications of demand tank d_1. The rest of supply s_2 can be sent directly to demand tank d_2 because there is no upper limit for specification A. The uneven inventory upper bounds on the tanks and the high symmetry derived from an empty initial inventory, increases the complexity of a seemingly simple instance. The maximum profit of this problem is 177.5 and an optimal flow schedule is shown in Figure 7.3. Table 7.6 contains the dimensions of the problem in terms of number of variables, constraints and bilinear terms.

Even though it is a relative trivial instance, global optimization solvers, such as BARON 14.0[52], ANTIGONE 1.1[51] or SCIP 3.1[103], have difficulty even finding a feasible solution to this problem when using the original MINLP formulation of Kolodziej et al.[79]. In fact, after 30 minutes of computational time, none of them reported a feasible solution. As mentioned before, this example motivates the need for alternative formulations and customized techniques that can handle even larger instances.

Table 7.6: Size of the ($\mathbb{C}$) formulation (explained below) for the motivating example.

Continuous Variables	Binary variables	Constraints	Bilinear terms
584	240	1178	128

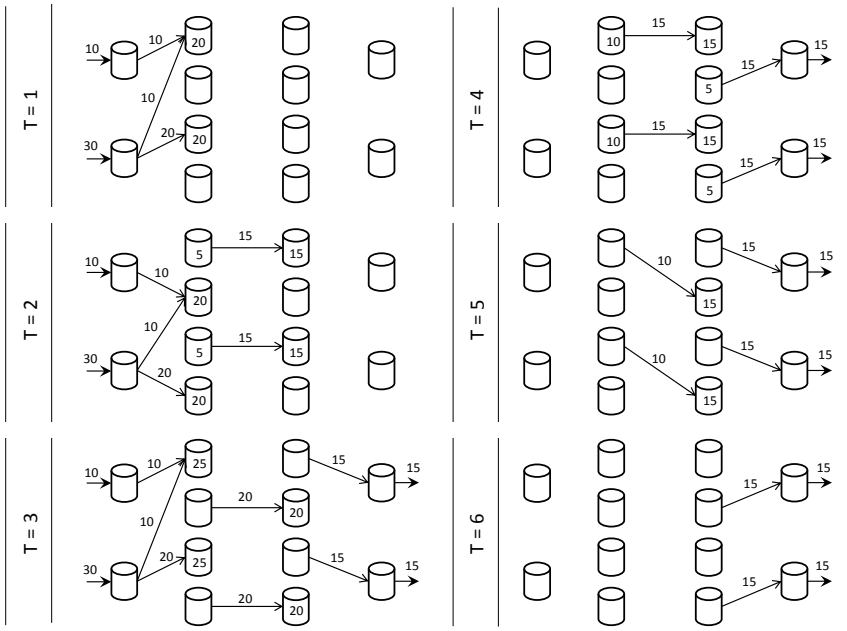

Figure 7.3: An optimal flow schedule for the motivating example

7.2.2 Generalized Disjunctive Programming (GDP) Formulations

In this section we present two alternative formulations for the multiperiod blending problem: a concentration model ($\mathbb{C}$) and a split fraction model ($\mathbb{SF}$). The concentration model ($\mathbb{C}$) includes the concentration of individual specifications as variables. As such, the bilinear terms of this formulation appear when a tank is "blending". The split fraction model ($\mathbb{SF}$) includes as variables the flow and inventory of individual specifications, and the split fraction of discharge. As such, the bilinear terms appear when a tank is "discharging". Both formulations are presented as Generalized Disjunctive Programming (GDP) models.

Kolodziej et al.[79] presented an MINLP model for the multiperiod blending problem, in terms of total flow and concentration. The GDP formulation of this concentration model ($\mathbb{C}$) is as follows:

($\mathbb{C}$):

$$\max \quad \sum_{t\in\mathcal{T}}\Big[\sum_{(n,d)\in\mathcal{A}}\beta_d^T F_{ndt} - \sum_{(s,n)\in\mathcal{A}}\beta_s^T F_{snt} - \sum_{(n,n')\in\mathcal{A}}(\alpha_{nn'}^N x_{nn't} + \beta_{nn'}^N F_{nn't})\Big] \tag{7.1}$$

s.t.

$$I_{st} = I_{st-1} + F_{st}^{\mathrm{IN}} - \sum_{(s,n)\in\mathcal{A}} F_{snt} \qquad s\in\mathcal{S},\, t\in\mathcal{T} \tag{7.2a}$$

$$I_{dt} = I_{dt-1} + \sum_{(n,d)\in\mathcal{A}} F_{ndt} - FD_{dt} \qquad d\in\mathcal{D},\, t\in\mathcal{T} \tag{7.2b}$$

$$\begin{bmatrix} X_{nbt} \\ F_{nb}^{\mathrm{L}} \le F_{nbt} \le F_{nb}^{\mathrm{U}} \end{bmatrix} \vee \begin{bmatrix} \neg X_{nbt} \\ F_{nbt} = 0 \end{bmatrix} \qquad (n,b)\in\mathcal{A},\, t\in\mathcal{T} \tag{7.3}$$

$$\begin{bmatrix} X_{sdt} \\ F_{sd}^{\mathrm{L}} \le F_{sdt} \le F_{sd}^{\mathrm{U}} \\ C_{qd}^{\mathrm{L}} \le C_{qs}^{\mathrm{IN}} \le C_{qd}^{\mathrm{U}} \quad q\in\mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{sdt} \\ F_{sdt} = 0 \end{bmatrix} \qquad (s,d)\in\mathcal{A},\, t\in\mathcal{T} \tag{7.4}$$

$$\begin{bmatrix} X_{bdt} \\ F_{bd}^{\mathrm{L}} \le F_{bdt} \le F_{bd}^{\mathrm{U}} \\ C_{qd}^{\mathrm{L}} \le C_{qbt-1} \le C_{qd}^{\mathrm{U}} \quad q\in\mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{bdt} \\ F_{bdt} = 0 \end{bmatrix} \qquad (b,d)\in\mathcal{A},\, t\in\mathcal{T} \tag{7.5}$$

$$\begin{bmatrix} YB_{bt} \\ I_{bt} = I_{bt-1} + \sum_{(n,b)\in\mathcal{A}} F_{nbt} \\ I_{bt}C_{qbt} = I_{bt-1}C_{qbt-1} + \sum_{(s,b)\in\mathcal{A}} F_{sbt}C_{qs}^{\text{IN}} \\ + \sum_{(b',b)\in\mathcal{A}} F_{b'bt}C_{qb't-1} \quad q \in \mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg YB_{bt} \\ I_{bt} = I_{bt-1} - \sum_{(b,n)\in\mathcal{A}} F_{bnt} \\ C_{qbt} = C_{qbt-1} \quad q \in \mathcal{Q} \end{bmatrix} \quad b \in \mathcal{B},\ t \in \mathcal{T} \tag{7.6}$$

$$X_{nbt} \Rightarrow YB_{bt} \qquad (n,b) \in \mathcal{A},\ t \in \mathcal{T} \tag{7.7a}$$
$$X_{bnt} \Rightarrow \neg YB_{bt} \qquad (b,n) \in \mathcal{A},\ t \in \mathcal{T} \tag{7.7b}$$

$$I_n^{\text{L}} \le I_{nt} \le I_n^{\text{U}} \qquad n \in \mathcal{N},\ t \in \mathcal{T} \tag{7.8a}$$
$$F_{nn'}^{\text{L}} \le F_{nn't} \le F_{nn'}^{\text{U}} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \tag{7.8b}$$
$$FD_{dt}^{\text{L}} \le FD_{dt} \le FD_{dt}^{\text{U}} \qquad d \in \mathcal{D},\ t \in \mathcal{T} \tag{7.8c}$$
$$C_q^{\text{L}} \le C_{qbt} \le C_q^{\text{U}} \qquad q \in \mathcal{Q},\ b \in \mathcal{B},\ t \in \mathcal{T} \tag{7.8d}$$

$$X_{nn't} \in \{True, False\} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \tag{7.9a}$$
$$YB_{bt} \in \{True, False\} \qquad b \in \mathcal{B},\ t \in \mathcal{T} \tag{7.9b}$$

In ($\mathbb{C}$), the objective function (7.1) maximizes the profit that results from delivering products to the demand tanks, minus the costs associated with supply flows as well as fixed and variable costs of transferring the liquids between tanks. Note that costs and revenues are accounted through flows leaving the supply tanks and entering the demand tanks. Equations (7.2) are total mass balances over the supply and demand tanks.

Disjunctions (7.3) to (7.5) represent the set of constraints regarding the existence of flow between nodes. If the flow between nodes exists ($X_{nn't} = True$), then upper and lower

bounds on flow and concentration are enforced. If the flow does not exist ($X_{nn't} = False$), then the flow is zero and no concentration constraints are enforced. Note that in disjunction (7.4) C_{qs}^{IN} is a parameter. However, this disjunction enforces that there can only exist flow between supply and demand when the supply specifications lie within the demand bounds ($C_{qd}^{\mathrm{L}} \leq C_{qs}^{\mathrm{IN}} \leq C_{qd}^{\mathrm{U}}$).

Disjunction (7.6) models the operation of the blending tanks. Since there cannot be simultaneous input/output streams to blending tanks, they can be either charging or discharging but not both. The total mass balance of the inventory is calculated if a tank is either charging or discharging. However, the individual specification inventory balance is only calculated when a tank is charging ($YB_{bt} = True$). When it is discharging ($YB_{bt} = False$), the required constraint specifies that there is no change in the concentration from the previous time period. Figure 7.4 illustrates the disjunction used to model the blending tanks.

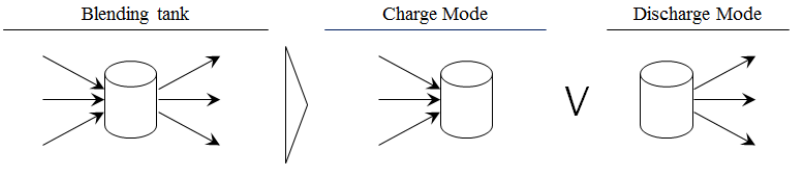

Figure 7.4: Illustration of no simultaneous input/output streams in a blending tank

Constraints (7.7) state the logic relationship between the binary variables. If there is flow coming into a blending tank ($X_{nbt} = True$), then YB_{bt} must be active ($YB_{bt} = True$) to indicate that it is in charging mode; the opposite if flow is leaving the tank. The last set of constraints (7.8) impose upper and lower bounds on the variables.

It is important to note that the original MINLP formulation of Kolodziej et al.[79] does not make use of disjunction (7.6). Instead, the mass balance of the blending tanks is described through global constraints. Introducing (7.6) in the formulation not only makes the "no simultaneous charge/discharge" condition more explicit, but it also reduces the number of bilinear terms. The reason for this reduction is that the mass balance individual specifications is defined as a global constraint in the model by Kolodziej et al. ($I_{bt}C_{qbt} = I_{bt-1}C_{qbt-1} + \sum_{(s,b)\in\mathcal{A}} F_{sbt}C_{qs}^{\mathrm{IN}} + \sum_{(b',b)\in\mathcal{A}} F_{b'bt}C_{qb't-1} - \sum_{(b,n)\in\mathcal{A}} F_{bnt}C_{qbt-1}$). As

such, bilinear terms appear in the constraint regardless if the tank is charging or discharging. Furthermore, the bilinear terms not only involve flow and concentration of blending tanks as in ($\mathbb{C}$), but also flow and concentration of the nodes connected to the blending tanks ($F_{bnt}C_{qbt-1}; (b,n) \in \mathcal{A}$). When compared with the original MINLP formulation of Kolodziej et al. for the motivating example presented before, the number of bilinear terms decreases 50%, from 248 to 128 . This formulation requires more binary variables but, due to the logic implications (7.7), it does not increase the combinatorial complexity of the problem.

Model ($\mathbb{C}$) uses total flows, inventories and concentration of specifications as variables ($F_{nn't}$, I_{nt} and C_{qbt}). In this sense, formulation ($\mathbb{C}$) is akin to the *p*-formulation of the pooling problem. An alternative formulation for the multiperiod blending problem is the split fraction model ($\mathbb{SF}$). ($\mathbb{SF}$) includes as variables the flow and inventory of individual specifications, and the split fraction of discharge ($\bar{F}_{qnn't}$, $\bar{I}_{qbt}$ and ξ_{bnt}). This type of model was first proposed by Quesada and Grossmann[83] in their work on general process networks.

The split fraction model ($\mathbb{SF}$) is as follows:

($\mathbb{SF}$):

$$\max \quad \sum_{t\in\mathcal{T}}\Big[\sum_{(n,d)\in\mathcal{A}}\beta_d^T F_{ndt} - \sum_{(s,n)\in\mathcal{A}}\beta_s^T F_{snt} - \sum_{(n,n')\in\mathcal{A}}(\alpha^N_{nn'}y_{nn't} + \beta^N_{nn'}F_{nn't})\Big] \tag{7.10}$$

s.t.

$$I_{st} = I_{st-1} + F^{\text{IN}}_{st} - \sum_{(s,n)\in\mathcal{A}} F_{snt} \qquad s\in\mathcal{S},\, t\in\mathcal{T} \tag{7.11a}$$

$$I_{dt} = I_{dt-1} + \sum_{(n,d)\in\mathcal{A}} F_{ndt} - FD_{dt} \qquad d\in\mathcal{D},\, t\in\mathcal{T} \tag{7.11b}$$

$$\begin{bmatrix} X_{nbt} \\ F^{\mathrm{L}}_{nb} \le F_{nbt} \le F^{\mathrm{U}}_{nb} \end{bmatrix} \vee \begin{bmatrix} \neg X_{nbt} \\ F_{nbt} = 0 \\ \bar{F}_{qnbt} = 0 \quad q \in \mathcal{Q} \end{bmatrix} \quad (n,b) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.12}$$

$$\begin{bmatrix} X_{sdt} \\ F^{\mathrm{L}}_{sd} \le F_{sdt} \le F^{\mathrm{U}}_{sd} \\ C^{\mathrm{L}}_{qd} \le C^{\mathrm{IN}}_{qs} \le C^{\mathrm{U}}_{qd} \quad q \in \mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{sdt} \\ F_{sdt} = 0 \\ \bar{F}_{qbdt} = 0 \quad q \in \mathcal{Q} \end{bmatrix} \quad (s,d) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.13}$$

$$\begin{bmatrix} X_{bdt} \\ F^{\mathrm{L}}_{bd} \le F_{bdt} \le F^{\mathrm{U}}_{bd} \\ F_{bdt} C^{\mathrm{L}}_{qd} \le \bar{F}_{qbdt} \le F_{bdt} C^{\mathrm{U}}_{qd} \quad q \in \mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{bdt} \\ F_{bdt} = 0 \\ \bar{F}_{qbdt} = 0 \quad q \in \mathcal{Q} \end{bmatrix} \quad (b,d) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.14}$$

$$\begin{bmatrix} YB_{bt} \\ I_{bt} = I_{bt-1} + \sum_{n \in \tilde{\mathcal{N}}_b} F_{nbt} \\ \bar{I}_{qbt} = \bar{I}_{qbt-1} + \sum_{(s,b) \in \mathcal{A}} F_{sbt} C^{\mathrm{IN}}_{qs} \\ + \sum_{(b',b) \in \mathcal{A}} \bar{F}_{qb'bt} \quad q \in \mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg YB_{bt} \\ I_{bt} = I_{bt-1} - \sum_{(b,n) \in \mathcal{A}} F_{bnt} \\ \bar{I}_{qbt} = \bar{I}_{qbt-1} \\ - \sum_{(b,n) \in \mathcal{A}} \bar{F}_{qbnt} \quad q \in \mathcal{Q} \\ F_{bnt} = \xi_{bnt} I_{bt-1} \quad (b,n) \in \mathcal{A} \\ \bar{F}_{qbnt} = \xi_{bnt} \bar{I}_{qbt-1} \quad q \in \mathcal{Q}, \\ (b,n) \in \mathcal{A} \end{bmatrix} \quad b \in \mathcal{B},\, t \in \mathcal{T} \tag{7.15}$$

$$X_{nbt} \Rightarrow YB_{bt} \qquad (n,b) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.16a}$$

$$X_{bnt} \Rightarrow \neg YB_{bt} \qquad (b,n) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.16b}$$

$$I_n^{\mathrm{L}} \leq I_{nt} \leq I_n^{\mathrm{U}} \qquad n \in \mathcal{N},\ t \in \mathcal{T} \qquad (7.17a)$$

$$F_{nn'}^{\mathrm{L}} \leq F_{nn't} \leq F_{nn'}^{\mathrm{U}} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \qquad (7.17b)$$

$$FD_{dt}^{\mathrm{L}} \leq FD_{dt} \leq FD_{dt}^{\mathrm{U}} \qquad d \in \mathcal{D},\ t \in \mathcal{T} \qquad (7.17c)$$

$$I_b^{\mathrm{L}} C_q^{\mathrm{L}} \leq \bar{I}_{qbt} \leq I_b^{\mathrm{U}} C_q^{\mathrm{U}} \qquad q \in \mathcal{Q},\ b \in \mathcal{B},\ t \in \mathcal{T} \qquad (7.17d)$$

$$F_{nn'}^{\mathrm{L}} C_q^{\mathrm{L}} \leq \bar{F}_{qnn't} \leq F_{nn'}^{\mathrm{U}} C_q^{\mathrm{U}} \qquad q \in \mathcal{Q},\ (n,n') \in \mathcal{A},\ t \in \mathcal{T} \qquad (7.17e)$$

$$0 \leq \xi_{bnt} \leq 1 \qquad (b,n) \in \mathcal{A},\ t \in \mathcal{T} \qquad (7.17f)$$

$$X_{nn't} \in \{True, False\} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \qquad (7.18a)$$

$$YB_{bt} \in \{True, False\} \qquad b \in \mathcal{B},\ t \in \mathcal{T} \qquad (7.18b)$$

The main difference between ($\mathbb{C}$) and ($\mathbb{SF}$) is that in the former the concentration of individual specifications is a variable (C_{qbt}), while in the latter the flow and inventory of individual specifications, and the split fraction of discharge are the variables ($\bar{F}_{qnn't}$, $\bar{I}_{qbt}$ and ξ_{bnt}). In ($\mathbb{SF}$), constraints (7.10) and (7.11) are the same as constraints (7.1) and (7.2) in ($\mathbb{C}$). Constraints (7.12), (7.13) and (7.14) enforce flow and concentration bounds when there exists flow between two nodes. Disjunction (7.15) models the charging and discharging constraints of blending tanks. In order to enforce the same specification concentrations in the outflows and inventory of a tank, it is necessary to introduce a new variable ξ_{bnt}. When discharging, ξ_{bnt} represents the proportion of the inventory that flows to a tank ($F_{bnt} = \xi_{bnt} I_{bt-1}$). This proportion needs to be the same for the total flow and the flow of the individual specifications ($\bar{F}_{qbnt} = \xi_{bnt} \bar{I}_{qbt-1}$). Note that in formulation ($\mathbb{SF}$) the bilinear terms appear in the formulation every time a blending tank operates in discharge mode ($YB_{bt} = False$). Constraints (7.16) state the logic relationship between the binary variables. Finally, (7.17) impose the bounds on the variables.

Note that model ($\mathbb{SF}$) is not equivalent to the q-formulation of the generalized pooling problem[77]. The proportion variables in the q-formulation denote the fraction of incoming flow to the blending tank that is contributed by input n, which implies that the sum over all n add to 1. In other words, the variables model the incoming streams to the tank and not what is being withdrawn, which does not have to sum up to one if the tank is not being

Table 7.7: Number of bilinear terms of GDP formulations. $\hat{B} = (b, b') \in \mathcal{A}$, $\hat{N}_b = (b, n) \in \mathcal{A}$.

Model	Bilinear terms	Motivating Example $\lvert\mathcal{Q}\rvert = 5, \lvert\mathcal{T}\rvert = 6$
($\mathbb{C}$)	$\lvert\mathcal{Q}\rvert \left[\lvert\mathcal{B}\rvert\lvert\mathcal{T}\rvert + \lvert\hat{\mathcal{B}}\rvert(\lvert\mathcal{T}\rvert - 1)\right]$	640
($\mathbb{SF}$)	$\lvert\hat{\mathcal{N}}_b\rvert(\mathcal{T} - 1)(1 + \lvert\mathcal{Q}\rvert)$	720

emptied completely. In addition, the variables in the q-formulation represent the fraction of raw materials that are supplied to the system, whereas the split fraction model tracks the specifications just as the concentration model. In other words, instead of fractions of raw materials, the variables of the split fraction model represent the actual amount of flow of each specification q in each and every stream. The q-formulation is discussed in more detail in Section 7.3.1. Nevertheless, Alfaki and Haugland[104] use proportions for flows transported from pools to demand tanks for the standard pooling problem and named it the TP formulation.

Table 7.7 compares the number of bilinear terms of the two GDP models. Table 7.7 also shows that, for the motivating example, ($\mathbb{C}$) has fewer bilinear terms than ($\mathbb{SF}$) (640 vs. 720). However, depending on the structure of the network, number of tanks, time periods and specifications, one formulation can have more bilinear terms than the other. Note that the number of continuous variables is larger when the system is modeled using individual flows and inventories ($\mathbb{SF}$).

The proposed formulations imply that a decomposition approach can be used to exploit the operational constraint on the blending tanks. By deciding whether the tank is in charge or discharge mode, the number of binary variables representing the connection between blending tanks and the rest of the network is reduced. Moreover, if the operating mode of the tanks is fixed at each period of time, the number of bilinear terms can be further reduced, thus yielding smaller and easier problems to solve. Before presenting the details of the decomposition algorithm, an alternative formulation is proposed.

7.3 Improving Formulation with Redundant Constraints

A crucial feature for solving a nonconvex MINLP is the tightness of the formulation when the non-convex constraints are relaxed (i.e. the MILP relaxation of a nonconvex MINLP). Note that performing the linear relaxation on the GDP and then using the (BM) reformulation yields the exact same MILP as first using the (BM) reformulation and then performing the linear relaxation (assuming the same big-M parameters are used). Therefore, the MILP relaxation of the (BM) reformulation of a GDP is the same as the (BM) reformulation of the linear GDP (LGDP) relaxation of the original GDP. A tighter LGDP relaxation of a GDP means a tighter MILP relaxation of the MINLP reformulation of the GDP. Therefore, tighter LGDP relaxations of a GDP typically translate into improved solution times in the MINLP reformulation of the GDP.

In this section we present two new models for the multiperiod blending problem: a source based model ($\mathbb{SB}$) and a hybrid model between the concentration and source based models ($\mathbb{CSB}$). We first describe the new source based model ($\mathbb{SB}$). We prove that the LGDP relaxation of the source based model ($\mathbb{SB}$) is tighter than the LGDP relaxation of the split fraction model ($\mathbb{SF}$). We present computational experiments that show that it is also tighter than the LGDP relaxation of the concentration model ($\mathbb{C}$) in all tested cases. Using the key idea behind the source based model ($\mathbb{SB}$), we then present an improvement to the concentration model ($\mathbb{C}$) using redundant constraints. The resulting model ($\mathbb{CSB}$) is a hybrid between the source based model ($\mathbb{SB}$) and the concentration model ($\mathbb{C}$). We prove that the model ($\mathbb{CSB}$) has the tightest LGDP relaxation of all the models presented in this chapter.

7.3.1 Alternative Problem Formulation

If the blending network is modeled using concentrations, as in the ($\mathbb{C}$) model, or using individual flows and inventories, as in the ($\mathbb{SF}$) model, the physical insight behind the equations is to track the specifications from supply to demand. The disadvantage of these models is that, when the non-convex constraints are dropped entirely, the composition

limits of the demand can be violated. For instance, if model ($\mathbb{C}$) is relaxed, total mass balances are the only equations that restrict flows and inventories. As a consequence, the streams entering the demand tanks can have any composition. Similarly, when the bilinear terms are dropped from model ($\mathbb{SF}$), the individual flows and inventories are allowed to take any value between the bounds. The drawback is that any configuration that satisfies the total mass balance in the tanks is a feasible solution for the relaxed problem, whereas most of them will be infeasible to the original problem.

Alternatively, there is the option of tracking the "sources" or "commodities" in the system, which is the insight behind the q-formulation of the pooling problem. This type of model has also been used in crude-oil scheduling problems. The idea of "following" the crudes along the network seems reasonable at the front-end of a refinery due to the specifications in the feed to the distillation columns[94,99].

Each supply and initial inventory in the blending tanks can be considered as a different "source". For instance, if crudes A and B are being unloaded and supplied to the system, in which tanks 1 and 3 contain an initial inventory of crude C and D respectively, the blending network has a total of four different types of crudes (or sources). Following with the crude-oil scheduling example, once the crudes are mixed and right before the mixture is discharged to the distillation columns, it is possible to calculate the relative amount of each specification in the blend since the composition of the sources is known. It is not until the final mixture of crudes is fed to the distillation columns that the composition specifications are checked.

Sources are defined as the supply and the blending tanks that have initial inventory greater than zero. The new index $r \in \mathcal{R}$ denotes the set of sources in the blending network. It is defined as $\mathcal{R} = \mathcal{S} \cup \breve{\mathcal{B}}$ where $\breve{\mathcal{B}} = \{b \in \mathcal{B} : I_b^0 > 0\}$. The variables in the model resemble the ones in the ($\mathbb{SF}$) model, but note that now $\tilde{F}_{rnn't}$ and $\tilde{I}_{rbt}$ are individual flows and inventories per source r instead of per specification q. Also, the model involves new parameters $\hat{C}_{qr}^0$ that represent the amount of specification q in source r and are defined as

follows:

$$\hat{C}^0_{qs} = C^{\text{IN}}_{qs} \qquad s \in \mathcal{S} \tag{7.19a}$$

$$\hat{C}^0_{qb} = C^0_{qb} \qquad b \in \breve{\mathcal{B}} \tag{7.19b}$$

The source based model ($\mathbb{SB}$) is similar to the split fraction model ($\mathbb{SF}$), but the sources are tracked instead of the specifications. In the ($\mathbb{SF}$) model, the fraction of specification q in a stream is defined as the amount of flow of specification q in the stream, divided by the total flow between tanks, see (7.20a). In model ($\mathbb{SB}$), the composition of a stream is determined from the compositions of each of the sources present in the stream. The sum of the amount of specification q in each source corresponds to the total amount of specification q in the stream, i.e. $\bar{F}_{qbdt} = \sum_{r \in \mathcal{R}} \tilde{F}_{rbdt} \hat{C}^0_{qr}$. If divided by the total flow, the composition can be calculated as in equation (7.20b).

$$C_{qbt} = \frac{\bar{F}_{qbdt}}{F_{bdt}} \qquad q \in \mathcal{Q},\ (b,d) \in \mathcal{A},\ t \in \mathcal{T} \tag{7.20a}$$

$$C_{qbt} = \frac{\sum_{r \in \mathcal{R}} \tilde{F}_{rbdt} \hat{C}^0_{qr}}{F_{bdt}} \qquad q \in \mathcal{Q}, (b,d) \in \mathcal{A},\ t \in \mathcal{T} \tag{7.20b}$$

The source based model ($\mathbb{SB}$), where sources $r \in \mathcal{R}$ correspond to the supply and blending tanks with initial inventory, is as follows:

($\mathbb{SB}$):

$$\max \quad \sum_{t \in \mathcal{T}} \Big[\sum_{(n,d) \in \mathcal{A}} \beta^T_d F_{ndt} - \sum_{(s,n) \in \mathcal{A}} \beta^T_s F_{snt} - \sum_{(n,n') \in \mathcal{N}} (\alpha^N_{nn'} y_{nn't} + \beta^N_{nn'} F_{nn't}) \Big] \tag{7.21}$$

s.t.

$$I_{st} = I_{st-1} + F^{\mathrm{IN}}_{st} - \sum_{(s,n)\in\mathcal{A}} F_{snt} \qquad s \in \mathcal{S},\, t \in \mathcal{T} \quad (7.22a)$$

$$I_{dt} = I_{dt-1} + \sum_{(n,d)\in\mathcal{A}} F_{ndt} - FD_{dt} \qquad d \in \mathcal{D},\, t \in \mathcal{T} \quad (7.22b)$$

$$F_{nn't} = \sum_{r\in\mathcal{R}} \tilde{F}_{rnn't} \qquad n \in \mathcal{N},\, n' \in \hat{\mathcal{N}}_n,\, t \in \mathcal{T} \quad (7.23a)$$

$$I_{bt} = \sum_{r\in\mathcal{R}} \tilde{I}_{rbt} \qquad b \in \mathcal{B},\, t \in \mathcal{T} \quad (7.23b)$$

$$\begin{bmatrix} X_{nbt} \\ F^{\mathrm{L}}_{nb} \le F_{nbt} \le F^{\mathrm{U}}_{nb} \end{bmatrix} \vee \begin{bmatrix} \neg X_{nbt} \\ F_{nbt} = 0 \end{bmatrix} \qquad (n,b) \in \mathcal{A},\, t \in \mathcal{T} \quad (7.24)$$

$$\begin{bmatrix} X_{sdt} \\ F^{\mathrm{L}}_{sd} \le F_{sdt} \le F^{\mathrm{U}}_{sd} \end{bmatrix} \vee \begin{bmatrix} \neg X_{sdt} \\ F_{sdt} = 0 \\ \tilde{F}_{rsdt} = 0 \quad r \in \mathcal{R} \end{bmatrix} \qquad (s,d) \in \mathcal{A},\, t \in \mathcal{T} \quad (7.25)$$

$$\begin{bmatrix} X_{bdt} \\ F^{\mathrm{L}}_{bd} \le F_{bdt} \le F^{\mathrm{U}}_{bd} \\ C^{\mathrm{L}}_{qd} F_{bdt} \le \sum\limits_{r\in R} \tilde{F}_{rbdt} \hat{C}^0_{qr} \le C^{U}_{qd} F_{bdt} \quad q \in Q \\ C^{\mathrm{L}}_{qd} I_{bt-1} \le \sum\limits_{r\in\mathcal{R}} \tilde{I}_{rbt-1} \hat{C}^0_{qr} \le C^{\mathrm{U}}_{qd} I_{bt-1} \quad q \in \mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{bdt} \\ F_{bdt} = 0 \end{bmatrix} \qquad (b,d) \in \mathcal{A},\, t \in \mathcal{T} \quad (7.26)$$

$$\begin{bmatrix} YB_{bt} \\ I_{bt} = I_{bt-1} + \sum\limits_{(n,b)\in\mathcal{A}} F_{nbt} \\ \tilde{I}_{rbt} = \tilde{I}_{rbt-1} \\ + \sum\limits_{(n,b)\in\mathcal{A}} \tilde{F}_{rnbt} \quad r \in \mathcal{R} \end{bmatrix} \vee \begin{bmatrix} \neg YB_{bt} \\ I_{bt} = I_{bt-1} - \sum\limits_{(b,n)\in\mathcal{A}} F_{bnt} \\ \tilde{I}_{rbt} = \tilde{I}_{rbt-1} \\ - \sum\limits_{(b,n)\in\mathcal{A}} \tilde{F}_{rbnt} \quad r \in \mathcal{R} \\ F_{bnt} = \xi_{bnt} I_{bt-1} \\ \tilde{F}_{rbnt} = \xi_{bnt}\tilde{I}_{rbt-1} \quad r \in \mathcal{R} \end{bmatrix} \quad b \in \mathcal{B}, t \in \mathcal{T} \tag{7.27}$$

$$X_{nbt} \Rightarrow YB_{bt} \qquad (n,b) \in \mathcal{A},\ t \in \mathcal{T} \tag{7.28a}$$
$$X_{bnt} \Rightarrow \neg YB_{bt} \qquad (b,n) \in \mathcal{A},\ t \in \mathcal{T} \tag{7.28b}$$

$$I_n^{\mathrm{L}} \le I_{nt} \le I_n^{\mathrm{U}} \qquad n \in \mathcal{N},\ t \in \mathcal{T} \tag{7.29a}$$
$$F_{nn'}^{\mathrm{L}} \le F_{nn't} \le F_{nn'}^{\mathrm{U}} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \tag{7.29b}$$
$$FD_{dt}^{\mathrm{L}} \le FD_{dt} \le FD_{dt}^{\mathrm{U}} \qquad d \in \mathcal{D},\ t \in \mathcal{T} \tag{7.29c}$$
$$I_b^{\mathrm{L}} \le \tilde{I}_{rbt} \le I_b^{\mathrm{U}} \qquad r \in \mathcal{R},\ b \in \mathcal{B},\ t \in \mathcal{T} \tag{7.29d}$$
$$F_{nn'}^{\mathrm{L}} \le \tilde{F}_{rnn't} \le F_{nn'}^{\mathrm{U}} \qquad r \in \mathcal{R},\ (n,n') \in \mathcal{A},\ t \in \mathcal{T} \tag{7.29e}$$
$$0 \le \xi_{bnt} \le 1 \qquad (b,n) \in \mathcal{A},\ t \in \mathcal{T} \tag{7.29f}$$

$$\tilde{F}_{rsnt}|_{r=s} = F_{snt} \qquad (s,n) \in \mathcal{A},\ t \in \mathcal{T} \tag{7.30a}$$
$$\tilde{F}_{rbnt}|_{r=b} = F_{bnt} \qquad (b,n) \in \mathcal{A},\ t = 1 \tag{7.30b}$$

$$X_{nn't} \in \{True, False\} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \tag{7.31a}$$
$$YB_{bt} \in \{True, False\} \qquad b \in \mathcal{B},\ t \in \mathcal{T} \tag{7.31b}$$

The source based model ($\mathbb{SB}$) follows the same general idea as the split fraction model ($\mathbb{SF}$). However, there are four main differences. The first one is that the individual flows

and inventories are based on sources $r \in \mathcal{R}$ instead of specifications $q \in \mathcal{Q}$. The second difference are the constraints (7.23). These constraints relate the source flows and inventories to the total flows and inventories, and they assume linear blending. Note that (7.23) is redundant for the GDP, however, it is not redundant for its LGDP relaxation. Also note that similar constraints cannot be included in the split fraction model ($\mathbb{SF}$), since the specifications can represent completely different properties (e.g. density and concentration of sulfur). The third difference is disjunction (7.26). In this disjunction, the bounds on the different specifications $q \in \mathcal{Q}$ for the demand, are transformed into restrictions for the sources $r \in \mathcal{R}$. This transformation is easily performed using the equations presented in (7.20). The last difference lies in equations (7.30). These equations link the supply and initial inventories with the corresponding individual flow per source. For instance, supply tank 1 holds source 1 and nothing else.

The LGDP relaxation of the source based model ($\mathbb{SB}$) is tighter than the LGDP relaxation of the split fraction model ($\mathbb{SF}$), as shown in the following theorem:

Theorem 7.3.1 *Let ($\mathbb{R}-\mathbb{SF}$) and ($\mathbb{R}-\mathbb{SB}$) be, respectively, an LGDP relaxation of ($\mathbb{SF}$) and ($\mathbb{SB}$) in which the nonlinear constraints are removed from the problem formulation. Then ($\mathbb{R}-\mathbb{SB}$) $\subseteq$ ($\mathbb{R}-\mathbb{SF}$).*

Proof. Let $(I_{nt}, F_{nn't}, FD_{dt}, \tilde{I}_{rbt}, \tilde{F}_{rnn't}, X_{nn't}, YB_{bt})$ be a feasible point in ($\mathbb{R}-\mathbb{SB}$). Let $\bar{I}_{qbt} = \sum_{r\in\mathcal{R}} \tilde{I}_{rbt}\hat{C}^0_{qr}$ and $\bar{F}_{qnn't} = \sum_{r\in\mathcal{R}} \tilde{F}_{rnn't}\hat{C}^0_{qr}$.

If $X_{nbt} = False$, then $\tilde{F}_{rnbt} = 0 \quad r \in \mathcal{R}$. Then, for every $q \in \mathcal{Q}$ it is possible to multiply both sides of the equation by $\hat{C}^0_{qr}$:

$$\hat{C}^0_{qr}\tilde{F}_{rnbt} = 0 \quad r \in \mathcal{R}, q \in \mathcal{Q} \tag{7.32}$$

By summing over all sources:

$$\sum_{r\in\mathcal{R}} \hat{C}^0_{qr}\tilde{F}_{rnbt} = 0 \quad q \in \mathcal{Q} \tag{7.33}$$

$$\bar{F}_{qnbt} = 0 \quad q \in \mathcal{Q} \tag{7.34}$$

The same scheme can be used when $X_{sdt} = False$, $X_{bdt} = False$ to obtain $\bar{F}_{qndt} = 0 \quad q \in \mathcal{Q}$. For the source inventory balance constraint (associated with the Boolean variable YB_{bt}) the same two steps can be applied.

If $YB_{bt} = True$, then $\tilde{I}_{rbt} = \tilde{I}_{rbt-1} + \sum_{(n,b)\in\mathcal{A}} \tilde{F}_{rnbt} \;\; r \in \mathcal{R}$, which implies:

$$\bar{I}_{qbt} = \bar{I}_{qbt-1} + \sum_{(n,b)\in\mathcal{A}} \bar{F}_{qnbt} \;\; q \in \mathcal{Q} \tag{7.35}$$

where $\bar{F}_{qsbt} = F_{sbt} C^{IN}_{qs} \;\; (s,b) \in \mathcal{A}$

If $YB_{bt} = False$, then $\tilde{I}_{rbt} = \tilde{I}_{rbt-1} - \sum_{(b,n)\in\mathcal{A}} \tilde{F}_{rbnt} \;\; r \in \mathcal{R}$, and then:

$$\bar{I}_{qbt} = \bar{I}_{qbt-1} - \sum_{(b,n)\in\mathcal{A}} \bar{F}_{qbnt} \;\; q \in \mathcal{Q} \tag{7.36}$$

When $X_{bdt} = True$, then $C^{\mathrm{L}}_{qd} F_{bdt} \leq \sum_{r\in\mathcal{R}} \tilde{F}_{rbdt} \hat{C}^0_{qr} \leq C^U_{qd} F_{bdt} \quad q \in Q$, so:

$$C^{\mathrm{L}}_{qd} F_{bdt} \leq \bar{F}_{qbdt} \leq C^U_{qd} F_{bdt} \quad q \in Q \tag{7.37}$$

The same procedure can be applied to obtain valid upper and lower bounds for the variables.

It is clear then, considering constraints (7.34), (7.35), (7.36) and (7.37), that for a feasible point

$(I_{nt}, F_{nn't}, FD_{dt}, \tilde{I}_{rbt}, \tilde{F}_{rnn't}, X_{nn't}, YB_{bt})$ in $(\mathbb{R}-\mathbb{SB})$ it is possible to set $\bar{I}_{qbt} = \sum_{r\in\mathcal{R}} \tilde{I}_{rbt} \hat{C}^0_{qr}$ and $\bar{F}_{qnn't} = \sum_{r\in\mathcal{R}} \tilde{F}_{rnn't} \hat{C}^0_{qr}$ and obtain the point $(I_{nt}, F_{nn't}, FD_{dt}, \bar{I}_{rbt}, \bar{F}_{rnn't}, X_{nn't}, YB_{bt})$

that is feasible for ($\mathbb{R}-\mathbb{SF}$). This means that any $(I_{nt}, F_{nn't}, FD_{dt}, X_{nn't}, YB_{bt})$ that is feasible for ($\mathbb{R}-\mathbb{SB}$) is also feasible for ($\mathbb{R}-\mathbb{SF}$) $\square$.

Note that Theorem 7.3.1 considers the linear relaxations ($\mathbb{R}-\mathbb{SB}$) and ($\mathbb{R}-\mathbb{SF}$) without the McCormick envelopes. The relaxations using the McCormick envelopes depend on the bounds of $\tilde{I}_{rbt}$ and $\hat{I}_{rbt}$.

Table 7.8 compares the value of the objective function of the LGDP relaxation of ($\mathbb{C}$), ($\mathbb{SF}$) and ($\mathbb{SB}$) for 9 instances. The LGDP relaxation was obtained using McCormick envelopes. All instances have 240 binary variables. The solutions were obtained using CPLEX 12.6. All values reported were below 1% gap after 1800 seconds of computational time. Values are normalized to the best known feasible solution. An asterisk $*$ marks the instances in which the value of the Boolean variables in the LGDP relaxation is the same as their value in the optimal solution to the GDP. i indicates those relaxed solution that will lead to an infeasible subproblem when the set of Boolean variables YB_{bt} is fixed accordingly. Instances with 1, 5 and 10 specifications and 1, 5 and 10 sources are used for the comparison. All the instances have 6 time periods and same network topology as the motivating example. Three conclusions can be inferred from the results:

1. For the examples tested, ($\mathbb{SB}$) is stronger than ($\mathbb{C}$) and ($\mathbb{SF}$) when the bilinear terms in the source based model are relaxed using McCormick envelopes.

2. The difference between relaxations is larger when the number of specifications is high and the number of sources is low.

3. In general, the size of the relaxed ($\mathbb{C}$) formulation strongly depends on the number of specifications, whereas the size of the relaxed ($\mathbb{SB}$) model depends on the number of sources.

The difference between the values of the normalized relaxations may not seem that significant at first. However, the feasibility of the subproblem when the set of discrete variables YB_{bt} is fixed according to the solution of the relaxed LGDP is crucial for the decomposition algorithm. In 6 of the 9 examples, the upper bound provided by the relaxation of the source based model ($\mathbb{SB}$) is the same as the best known solution. Furthermore, when fixing

Table 7.8: Comparison between the LGDP relaxation of different formulations.

			# Variables			# Constraints			Normalized relaxation		
Ex.	$\|\mathcal{R}\|$	$\|\mathcal{Q}\|$	($\mathbb{C}$)	($\mathbb{SF}$)	($\mathbb{SB}$)	($\mathbb{C}$)	($\mathbb{SF}$)	($\mathbb{SB}$)	($\mathbb{C}$)	($\mathbb{SF}$)	($\mathbb{SB}$)
1	2	1	681	889	793	1558	2582	2054	1.007	1.020^i	1.000^*
2	2	5	1385	1849	793	4342	7158	2822	1.012	1.026^i	1.000^*
3	2	10	2265	3049	793	7846	12902	3782	1.012^i	1.026^i	1.000^*
4	5	1	681	889	2713	1558	2582	5894	1.006^i	1.006	1.000^*
5	5	5	1385	1849	2713	4342	7158	6662	1.022	1.022	1.011
6	5	10	2265	3049	2713	7846	12902	7622	1.022^i	1.022^i	1.006
7	10	1	681	889	1513	1558	2582	3494	1.000^*	1.000^*	1.000^*
8	10	5	1385	1849	1513	4342	7158	4262	1.005	1.005	1.005
9	10	10	2265	3049	1513	7846	12902	5222	1.005	1.005	1.005

the Boolean variables YB_{bt} form the LGDP relaxation of ($\mathbb{SB}$), all solutions are feasible to the original problem. In the concentration ($\mathbb{C}$) and split fraction ($\mathbb{SF}$) models 3 and 4 instances become infeasible, when fixing the value of YB_{bt} from the LGDP relaxation.

Note that the q and pq-formulations are also exploiting the idea of sources or commodities to model the blending process. Even though the ideas are similar, these formulation have clear differences with the source based model. The proportion variables in the q-formulation denote the fraction that each source contributes to the total incoming flow to the blending tank, which implies that the sum of the fractions over all sources add to 1. Instead of following the fraction of the total flow that corresponds to each source, the source based model tracks the actual amount of source in each and every stream in the system. Also, the source based model uses splits fractions in order to ensure consistency in the discharge. This is not necessary in the traditional q-formulation, since the pooling problem does not consider inventories. Gupte et al.[105] proposed an extension of the q-formulation to handle inventories and semi-continuous flows. However, their model requires the introduction of more bilinear terms with up to five indexes per term. This implies that the number of bilinear terms will increase drastically even with small instances. Finally, the classical Haverly pooling problem is used to illustrate the difference between the traditional formulations in the pooling community and the new formulation presented in this report. See Appendix D for details.

7.3.2 Using redundant constraints in the ($\mathbb{C}$) model

The linear constraints of the source based model ($\mathbb{SB}$) can be used as redundant constraints in the concentration model ($\mathbb{C}$). This allows to obtain stronger LGDP relaxations. The new model ($\mathbb{CSB}$) (hybrid of the ($\mathbb{C}$) and ($\mathbb{SB}$) models) will increase in size but will have a stronger LGDP relaxation. The model is as follows:

($\mathbb{CSB}$):

$$\max \quad \sum_{t\in\mathcal{T}}\Big[\sum_{(n,d)\in\mathcal{A}}\beta_d^T F_{ndt} - \sum_{(s,n)\in\mathcal{A}}\beta_s^T F_{snt} - \sum_{(n,n')\in\mathcal{A}}(\alpha^N_{nn'}x_{nn't} + \beta^N_{nn'}F_{nn't})\Big] \tag{7.38}$$

s.t.

$$I_{st} = I_{st-1} + F^{\mathrm{IN}}_{st} - \sum_{(s,n)\in\mathcal{A}} F_{snt} \qquad s\in\mathcal{S},\, t\in\mathcal{T} \tag{7.39a}$$

$$I_{dt} = I_{dt-1} + \sum_{(n,d)\in\mathcal{A}} F_{ndt} - FD_{dt} \qquad d\in\mathcal{D},\, t\in\mathcal{T} \tag{7.39b}$$

$$F_{nn't} = \sum_{r\in\mathcal{R}} \tilde{F}_{rnn't} \qquad (n,n')\in\mathcal{A},\, t\in\mathcal{T} \tag{7.40a}$$

$$I_{bt} = \sum_{r\in\mathcal{R}} \tilde{I}_{rbt} \qquad b\in\mathcal{B},\, t\in\mathcal{T} \tag{7.40b}$$

$$\begin{bmatrix} X_{nbt} \\ F^{\mathrm{L}}_{nb} \le F_{nbt} \le F^{\mathrm{U}}_{nb} \end{bmatrix} \vee \begin{bmatrix} \neg X_{nbt} \\ F_{nbt} = 0 \end{bmatrix} \qquad (n,b)\in\mathcal{A},\, t\in\mathcal{T} \tag{7.41}$$

$$\begin{bmatrix} X_{sdt} \\ F^{\mathrm{L}}_{sd} \le F_{sdt} \le F^{\mathrm{U}}_{sd} \\ C^{\mathrm{L}}_{qd} \le C^{\mathrm{IN}}_{qs} \le C^{\mathrm{U}}_{qd} \quad q\in\mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{sdt} \\ F_{sdt} = 0 \end{bmatrix} \qquad (s,d)\in\mathcal{A},\, t\in\mathcal{T} \tag{7.42}$$

$$\begin{bmatrix} X_{bdt} & \\ F_{bd}^{\mathrm{L}} \le F_{bdt} \le F_{bd}^{\mathrm{U}} & \\ C_{qd}^{\mathrm{L}} \le C_{qbt-1} \le C_{qd}^{\mathrm{U}} & q \in \mathcal{Q} \\ C_{qd}^{\mathrm{L}} F_{bdt} \le \sum_{r\in\mathcal{R}} \tilde{F}_{rbdt} \hat{C}_{qr}^{0} \le C_{qd}^{\mathrm{U}} F_{bdt} & q \in \mathcal{Q} \\ C_{qd}^{\mathrm{L}} I_{bt-1} \le \sum_{r\in\mathcal{R}} \tilde{I}_{rbt-1} \hat{C}_{qr}^{0} \le C_{qd}^{\mathrm{U}} I_{bt-1} & q \in \mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{bdt} \\ F_{bdt} = 0 \end{bmatrix} \quad (b,d) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.43}$$

$$\begin{bmatrix} YB_{bt} & \\ I_{bt} = I_{bt-1} + \sum_{(n,b)\in\mathcal{A}} F_{nbt} & \\ I_{bt} C_{qbt} = I_{bt-1} C_{qbt-1} + \sum_{(s,b)\in\mathcal{A}} F_{sbt} C_{qs}^{\mathrm{IN}} & \\ + \sum_{(b',b)\in\mathcal{A}} F_{b'bt} C_{qb't-1} & q \in \mathcal{Q} \\ \tilde{I}_{rbt} = \tilde{I}_{rbt-1} + \sum_{(n,b)\in\mathcal{A}} \tilde{F}_{rnbt} & r \in \mathcal{R} \end{bmatrix} \vee \begin{bmatrix} \neg YB_{bt} & \\ I_{bt} = I_{bt-1} - \sum_{(b,n)\in\mathcal{A}} F_{bnt} & \\ C_{qbt} = C_{qbt-1} & q \in \mathcal{Q} \\ \tilde{I}_{rbt} = \tilde{I}_{rbt-1} & \\ - \sum_{(b,n)\in\mathcal{A}} \tilde{F}_{rbnt} & r \in \mathcal{R} \end{bmatrix} b \in \mathcal{B}, t \in \mathcal{T} \tag{7.44}$$

$$X_{nbt} \Rightarrow YB_{bt} \qquad b \in \mathcal{B},\, n \in \check{\mathcal{N}}_b,\, t \in \mathcal{T} \tag{7.45a}$$

$$X_{bnt} \Rightarrow \neg YB_{bt} \qquad b \in \mathcal{B},\, n \in \hat{\mathcal{N}}_b,\, t \in \mathcal{T} \tag{7.45b}$$

$$I_n^{\mathrm{L}} \le I_{nt} \le I_n^{\mathrm{U}} \qquad n \in \mathcal{N},\, t \in \mathcal{T} \tag{7.46a}$$

$$F_{nn'}^{\mathrm{L}} \le F_{nn't} \le F_{nn'}^{\mathrm{U}} \qquad (n,n') \in \mathcal{A},\, t \in \mathcal{T} \tag{7.46b}$$

$$FD_{dt}^{\mathrm{L}} \le FD_{dt} \le FD_{dt}^{\mathrm{U}} \qquad d \in \mathcal{D},\, t \in \mathcal{T} \tag{7.46c}$$

$$C_q^{\mathrm{L}} \le C_{qbt} \le C_q^{\mathrm{U}} \qquad q \in \mathcal{Q},\, b \in \mathcal{B},\, t \in \mathcal{T} \tag{7.46d}$$

$$I_b^{\mathrm{L}} \le \tilde{I}_{rbt} \le I_b^{\mathrm{U}} \qquad r \in \mathcal{R},\, b \in \mathcal{B},\, t \in \mathcal{T} \tag{7.46e}$$

$$F_{nn'}^{\mathrm{L}} \le \tilde{F}_{rnn't} \le F_{nn'}^{\mathrm{U}} \qquad r \in \mathcal{R},\, (n,n') \in \mathcal{A},\, t \in \mathcal{T} \tag{7.46f}$$

$$\tilde{F}_{rsnt}|_{r=s} = F_{snt} \qquad (s,n) \in \mathcal{A},\ t \in \mathcal{T} \quad (7.47a)$$

$$\tilde{F}_{rbnt}|_{r=b} = F_{bnt} \qquad (b,n) \in \mathcal{A},\ t = 1 \quad (7.47b)$$

$$X_{nn't} \in \{True, False\} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \quad (7.48a)$$

$$YB_{bt} \in \{True, False\} \qquad b \in \mathcal{B},\ t \in \mathcal{T} \quad (7.48b)$$

In addition to the constraints in the concentration model ($\mathbb{C}$), ($\mathbb{CSB}$) includes the last two inequalities in the first term of the disjunction (7.43), the last equations in disjunction (7.44), and equalities (7.40) and (7.47). Note that all of these inequalities are linear.

Consider the concentration model ($\mathbb{C}$), the source based model ($\mathbb{SB}$), and the hybrid model ($\mathbb{CSB}$). The LGDP relaxation of ($\mathbb{CSB}$) is tighter than the LGDP relaxation of the other two, as stated in the following theorem:

Theorem 7.3.2 *Let ($\mathbb{R}-\mathbb{C}$), ($\mathbb{R}-\mathbb{SB}$) and ($\mathbb{R}-\mathbb{CSB}$) be, respectively, a linear relaxation of ($\mathbb{C}$), ($\mathbb{SB}$) and ($\mathbb{CSB}$) in which the nonlinear constraints are removed from the problem formulation. Then ($\mathbb{R}-\mathbb{CSB}$) $\subseteq$ ($\mathbb{R}-\mathbb{SB}$) and ($\mathbb{R}-\mathbb{CSB}$) $\subseteq$ ($\mathbb{R}-\mathbb{C}$).*

The proof of Theorem 7.3.2 is trivial, since ($\mathbb{R}-\mathbb{CSB}$) includes all of the constraints of ($\mathbb{R}-\mathbb{C}$) and ($\mathbb{R}-\mathbb{SB}$).

In summary, we have presented four formulations in this chapter: the concentration model ($\mathbb{C}$), the split fraction model ($\mathbb{SF}$), the source based model ($\mathbb{SB}$), and the hybrid model ($\mathbb{CSB}$). We can stablish the following relations between the LGDP relaxation of these models: ($\mathbb{R}-\mathbb{CSB}$) $\subseteq$ ($\mathbb{R}-\mathbb{SB}$) $\subseteq$ ($\mathbb{R}-\mathbb{SF}$), and ($\mathbb{R}-\mathbb{CSB}$) $\subseteq$ ($\mathbb{R}-\mathbb{C}$). Therefore, when removing the nonlinear constraints from the formulations, ($\mathbb{CSB}$) is stronger than the other formulations. Note that a linear relaxation of the different formulations can be achieved by using McCormick[67] envelopes of the bilinear terms. In such a case, the strength of the linear relaxation also depends on the bounds of the variables involved in the bilinear terms. In real applications, it is likely that the bounds for total flow and concentration are stronger than the bounds for individual specification inventories and split fractions. In such cases, the advantage of ($\mathbb{CSB}$) over ($\mathbb{R}-\mathbb{SB}$), and ($\mathbb{R}-\mathbb{SF}$) is further increased.

Table 7.9: Bilinear terms in GDP formulations. $\hat{B} = (b, b') \in \mathcal{A}$, $\hat{N}_b = (b, n) \in \mathcal{A}$.

Model	Bilinear terms	Motivating Example $\lvert\mathcal{Q}\rvert = 5, \lvert\mathcal{T}\rvert = 6$ $I_b^0 = 0, \lvert\mathcal{R}\rvert = 2$	Motivating Example $\lvert\mathcal{Q}\rvert = 5, \lvert\mathcal{T}\rvert = 6$ $I_b^0 > 0, \lvert\mathcal{R}\rvert = 10$
($\mathbb{CSB}$)	$\lvert\mathcal{Q}\rvert \left[\lvert\mathcal{B}\rvert\lvert\mathcal{T}\rvert + \lvert\hat{\mathcal{B}}\rvert(\lvert\mathcal{T}\rvert - 1)\right]$	640	640
($\mathbb{SB}$)	$\lvert\hat{\mathcal{N}}_b\rvert(\mathcal{T} - 1)(1 + \lvert\mathcal{R}\rvert)$	480	1760

Table 7.10: Fraction of instances for which a feasible solution was found in less than 30 minutes.

Solver	($\mathbb{CSB}$)	($\mathbb{C}$)
SCIP	**0.42**	0.31
BARON	0.29	0.21
ANTIGONE	0.31	0.29

The number of bilinear terms in ($\mathbb{CSB}$) is the same as in ($\mathbb{C}$). The number of bilinear terms in ($\mathbb{SB}$) depends not only on $\mathcal{Q}$, $\mathcal{T}$ and $\mathcal{B}$, but also on $\mathcal{S}$ and the number of blending tanks with $I_b^0 > 0$. Table 7.9 presents the number of bilinear terms for ($\mathbb{CSB}$) and for ($\mathbb{SB}$) for two instances. Both instances have the same topology and $|\mathcal{Q}| = 5$ and $|\mathcal{T}| = 6$. However, the initial inventory of all the blending tanks in the first instance is zero. The initial inventory of all blending tanks in the second instance is greater than zero. It is clear from Table 7.9 that the number of bilinear terms for the ($\mathbb{SB}$) can change drastically for "similar" instances (480 vs.1760).

The MINLP reformulation of the concentration model, with and without redundant constraints (($\mathbb{C}$), ($\mathbb{CSB}$)) was tested in 48 instances. Half of these instances include initial inventory and the other half do not (See Section 7.5 for more details on the instances). Table 7.10 shows the fraction of the instances for which the solver could find at least one feasible solution. The global solvers BARON 14.0, ANTIGONE 1.1, and SCIP 3.1 were used.

In general, a feasible solution is obtained for a larger number of instances if the problem is modeled using the redundant constraints. For instance, in the case of SCIP, the number of instances for which SCIP can find a solution increase from 15 to 20 out of 48 instances. In

addition, SCIP performs better than its competitors, since it can find a feasible solution in 42% of the instances against the 29% and 31% of BARON and ANTIGONE, respectively. Due to this superior performance, SCIP 3.1 is used as a reference for comparison in the computational results.

It can be seen that the performance of the solver is better when the redundant constraints are added to the concentration model. Nevertheless, the number of instances for which a feasible solution was found is still small. This motivates the need to develop a specialized algorithm that can better exploit the structure of the problem.

7.4 Iterative Two-Stage Decomposition Algorithm

Considering the performance of commercial solvers and the potential advantages of the ($\mathbb{CSB}$) formulation, a decomposition algorithm is proposed next. As mentioned before, if the operating mode of the blending tanks is fixed, the resulting GDP becomes easier to solve, due to a reduction in size and complexity. In fact, all variables related to incoming arcs to a blending tank (i.e. F_{nbt} and x_{nbt}) will be removed from the model when the tank is discharging. Similarly, if the tank is being charged, all outgoing flows from the blending tank (i.e. F_{bnt} and x_{bnt}) will be set to zero. Furthermore, the number of bilinear terms will decrease compared to the original GDP in the following circumstances:

1. If the blending tank is discharging at time t, the equations that describe the operation are linear for that period (i.e., the second disjunct of disjunction (7.44) is True). Thus, all bilinearities related to that blending tank and time period are eliminated from the model. In addition, if a blending tank is in idle mode, it can be set to discharge mode in order to avoid considering unnecessary bilinear terms.

2. The bilinear term $F_{b'bt}C_{qb't-1}$ has to be included if and only if tank b' is discharging ($\neg Y B_{b't}$) and tank b is charging ($Y B_{bt}$) at time t. Therefore, if blending tank b has an incoming stream from a supply tank, i.e. it is in charge mode, but there is no other blending tank (b'), connected to tank b, that is discharging at that time t, bilinear terms of the form $F_{b'bt}C_{qb't-1}$ are unnecessary and can be eliminated.

To exploit these ideas, the proposed algorithm decomposes the GDP model into two levels. The first level, or master problem, is a linear relaxation of the original GDP that provides rigorous upper bounds for the profit. The second level, or subproblem, is a smaller GDP in which the set of discrete variables YB_{bt} is fixed. The subproblem, when a feasible solution is found, provides a feasible solution to the original GDP and a rigorous lower bound. These problems are solved successively until the gap between the upper and lower bounds is within a tolerance. Figure 7.5 presents the flow diagram of the algorithm.

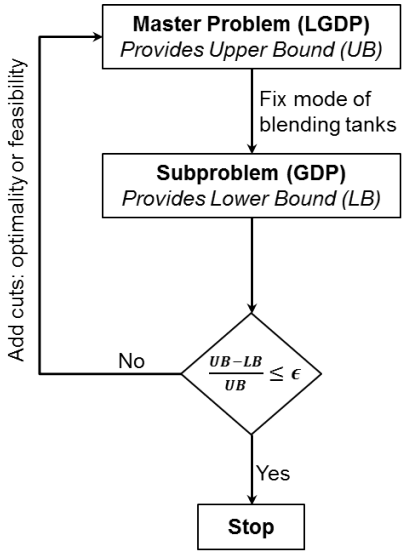

Figure 7.5: Decomposition Algorithm

The solution of the master problem is used to define the subproblem, which is more tractable than the original problem. A master problem with a tight relaxation is crucial for the success of the algorithm, since its solution will be used to fixed the operating mode of the tanks. The feasibility of the subproblem will strictly depend on the solution of the master problem.

As mentioned in the introduction, there are many relaxation techniques that can be used to construct the master problem. In the algorithm, the master problem is a linear relaxation of ($\mathbb{CSB}$) in which the non-convex constraints are dropped. Optimality and/or feasibility cuts are added in the form of integer cuts, eliminating regions already evaluated in previous

iterations. Note that McCormick envelopes could be used for linearly relaxing ($\mathbb{CSB}$). However, from computational experiments we observed that dropping the nonlinearities improved the performance of the algorithm. In particular, the master problem solves faster, and we did not observe a significant difference in the number of iterations of the algorithm. We acknowledge that for other instances the use of McCormick envelopes could help the algorithm to perform better.

The subproblem can be solved using a global optimization solver or through a specialized technique that ensures global optimality. The concentration model plus the source based redundant constraints ($\mathbb{CSB}$) is used in the subproblem.

7.4.1 Description of the algorithm

The following master problem is a linear relaxation of the ($\mathbb{CSB}$) in which the nonlinear constraints were dropped. Also, optimality and/or feasibility cuts are added in the form of integer cuts:

($\mathbb{MP}$):

$$\max \quad Z \tag{7.49}$$

s.t.

$$Z \leq \sum_{t\in\mathcal{T}} \Big[\sum_{(n,d)\in\mathcal{A}} \beta_d^T F_{ndt} - \sum_{(s,n)\in\mathcal{A}} \beta_s^T F_{snt} - \sum_{(n,n')\in\mathcal{N}} (\alpha^N_{nn'} y_{nn't} + \beta^N_{nn'} F_{nn't}) \Big] \tag{7.50}$$

$$I_{st} = I_{st-1} + F^{\text{IN}}_{st} - \sum_{(s,n)\in\mathcal{A}} F_{snt} \qquad s\in\mathcal{S},\, t\in\mathcal{T} \tag{7.51a}$$

$$I_{dt} = I_{dt-1} + \sum_{(n,d)\in\mathcal{A}} F_{ndt} - FD_{dt} \qquad d\in\mathcal{D},\, t\in\mathcal{T} \tag{7.51b}$$

$$F_{nn't} = \sum_{r\in\mathcal{R}} \tilde{F}_{rnn't} \qquad n \in \mathcal{N},\, n' \in \hat{\mathcal{N}}_n,\, t \in \mathcal{T} \tag{7.52a}$$

$$I_{bt} = \sum_{r\in\mathcal{R}} \tilde{I}_{rbt} \qquad b \in \mathcal{B},\, t \in \mathcal{T} \tag{7.52b}$$

$$\begin{bmatrix} X_{nbt} \\ F^{\mathrm{L}}_{nb} \le F_{nbt} \le F^{\mathrm{U}}_{nb} \end{bmatrix} \vee \begin{bmatrix} \neg X_{nbt} \\ F_{nbt} = 0 \end{bmatrix} \qquad (n,b) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.53}$$

$$\begin{bmatrix} X_{sdt} \\ F^{\mathrm{L}}_{sd} \le F_{sdt} \le F^{\mathrm{U}}_{sd} \\ C^{\mathrm{L}}_{qd} \le C^{\mathrm{IN}}_{qs} \le C^{\mathrm{U}}_{qd} \end{bmatrix} \vee \begin{bmatrix} \neg X_{sdt} \\ F_{sdt} = 0 \end{bmatrix} \qquad (s,d) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.54}$$

$$\begin{bmatrix} X_{bdt} & \\ F^{\mathrm{L}}_{bd} \le F_{bdt} \le F^{\mathrm{U}}_{bd} & \\ C^{\mathrm{L}}_{qd} \le C_{qbt-1} \le C^{\mathrm{U}}_{qd} & q \in \mathcal{Q} \\ C^{\mathrm{L}}_{qd} F_{bdt} \le \sum_{r\in\mathcal{R}} \tilde{F}_{rbdt} \hat{C}^0_{qr} \le C^{\mathrm{U}}_{qd} F_{bdt} & q \in \mathcal{Q} \\ C^{\mathrm{L}}_{qd} I_{bt-1} \le \sum_{r\in\mathcal{R}} \tilde{I}_{rbt-1} \hat{C}^0_{qr} \le C^{\mathrm{U}}_{qd} I_{bt-1} & q \in \mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{bdt} \\ F_{bdt} = 0 \end{bmatrix} \qquad (b,d) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.55}$$

$$\begin{bmatrix} YB_{bt} & & \\ yb_{bt} = 1 & & \\ I_{bt} = I_{bt-1} + \sum_{(n,b)\in\mathcal{A}} F_{nbt} & & \\ \tilde{I}_{rbt} = \tilde{I}_{rbt-1} & & \\ + \sum_{(n,b)\in\mathcal{A}} \tilde{F}_{rnbt} & r \in \mathcal{R} & \end{bmatrix} \vee \begin{bmatrix} \neg YB_{bt} & & \\ yb_{bt} = 0 & & \\ I_{bt} = I_{bt-1} - \sum_{(b,n)\in\mathcal{A}} F_{bnt} & & \\ C_{qbt} = C_{qbt-1} & q \in \mathcal{Q} & \\ \tilde{I}_{rbt} = \tilde{I}_{rbt-1} - \sum_{(b,n)\in\mathcal{A}} \tilde{F}_{rbnt} & r \in \mathcal{R} & \end{bmatrix} \qquad b \in \mathcal{B}, t \in \mathcal{T} \tag{7.56}$$

$$
\begin{aligned}
Z \leq -(UB - Z^i)\Big(\sum_{\substack{b\in B,\, t\in T:\\ \hat{yb}^i_{bt}=1}} yb_{bt} - \sum_{\substack{b\in B,\, t\in T:\\ \hat{yb}^i_{bt}=0}} yb_{bt} \Big) \\
+(UB - Z^i)\Big(\sum_{b\in B,\, t\in T} (\hat{yb}^i_{bt}) - 1 \Big) + UB \qquad i \in \mathcal{I}_O
\end{aligned} \tag{7.57}
$$

$$
\sum_{\substack{b\in B,\, t\in T:\\ \hat{yb}^i_{bt}=1}} (1 - yb_{bt}) \; + \sum_{\substack{b\in B,\, t\in T:\\ \hat{yb}^i_{bt}=0}} yb_{bt} \geq 1 \qquad i \in \mathcal{I}_F \tag{7.58}
$$

$$X_{nbt} \Rightarrow YB_{bt} \qquad (n,b) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.59a}$$

$$X_{bnt} \Rightarrow \neg YB_{bt} \qquad (b,n) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.59b}$$

$$I^{\mathrm{L}}_n \leq I_{nt} \leq I^{\mathrm{U}}_n \qquad n \in \mathcal{N},\, t \in \mathcal{T} \tag{7.60a}$$

$$F^{\mathrm{L}}_{nn'} \leq F_{nn't} \leq F^{\mathrm{U}}_{nn'} \qquad (n,n') \in \mathcal{A},\, t \in \mathcal{T} \tag{7.60b}$$

$$FD^{\mathrm{L}}_{dt} \leq FD_{dt} \leq FD^{\mathrm{U}}_{dt} \qquad d \in \mathcal{D},\, t \in \mathcal{T} \tag{7.60c}$$

$$I^{\mathrm{L}}_b \leq \tilde{I}_{rbt} \leq I^{\mathrm{U}}_b \qquad r \in \mathcal{R},\, b \in \mathcal{B},\, t \in \mathcal{T} \tag{7.60d}$$

$$F^{\mathrm{L}}_{nn'} \leq \tilde{F}_{rnn't} \leq F^{\mathrm{U}}_{nn'} \qquad r \in \mathcal{R},\, (n,n') \in \mathcal{A},\, t \in \mathcal{T} \tag{7.60e}$$

$$0 \leq \xi_{bnt} \leq 1 \qquad (b,n) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.60f}$$

$$\tilde{F}_{rsnt}|_{r=s} = F_{snt} \qquad (s,n) \in \mathcal{A},\, t \in \mathcal{T} \tag{7.61a}$$

$$\tilde{F}_{rbnt}|_{r=b} = F_{bnt} \qquad (b,n) \in \mathcal{A},\, t = 1 \tag{7.61b}$$

$$X_{nn't} \in \{True, False\} \qquad (n,n') \in \mathcal{A},\, t \in \mathcal{T} \tag{7.62a}$$

$$YB_{bt} \in \{True, False\} \qquad b \in \mathcal{B},\, t \in \mathcal{T} \tag{7.62b}$$

Note that variable yb_{bt} is introduced in the formulation. This variable takes the value of the binary variable that corresponds to YB_{bt} in the (BM) reformulation of the GDP (i.e. $yb_{bt} = 1$, when $YB_{bt} = True$). It is necessary to introduce the variable to add the enumeration cuts (7.57) and (7.58), which are added in the form of integer cuts that

eliminate regions already evaluated in previous iterations. $\mathcal{I}_F$ is the set of enumeration cuts that are added when a subproblem is infeasible[70]. $\mathcal{I}_O$ is the set of enumeration cuts that are added otherwise[71]. Z is the value of the objective function, UB a global upper bound for the GDP, and Z^i an upper bound for the objective function corresponding to the solution $\hat{yb}_{bt}^{\,i}$. (7.58) will eliminate from the feasible space those solutions for the master problem that resulted in infeasible subproblems. When yb_{bt} is different from $\hat{yb}_{bt}^{\,i}$, then $\sum_{b\in B,\, t\in T:\ \hat{yb}_{bt}^{i}=1} yb_{bt} - \sum_{b\in B,\, t\in T:\ \hat{yb}_{bt}^{i}=0} yb_{bt}$ is smaller than $\sum_{b\in B,\, t\in T}(\hat{yb}_{bt}^{\,i})$ and (7.57) becomes $Z \leq UB$ (or an even weaker cut). When $yb_{bt} = \hat{yb}_{bt}^{\,i}$, then $\sum_{b\in B,\, t\in T:\ \hat{yb}_{bt}^{i}=1} yb_{bt} - \sum_{b\in B,\, t\in T:\ \hat{yb}_{bt}^{i}=0} yb_{bt} = \sum_{b\in B,\, t\in T}(\hat{yb}_{bt}^{\,i})$. In such a case, (7.57) becomes $Z \leq Z^i$ and the cut is valid (since Z^i is an upper bound of the objective function in the solution $\hat{yb}_{bt}^{\,i}$).

For a given $\tilde{YB}_{bt}^{\text{fix}} \in \{True, False\}$, $b \in \mathcal{B}$, $t \in \mathcal{T}$, consider the following subproblem (which is the ($\mathbb{CSSB}$) model with tanks fixed in "charge" or "discharge" mode):

($\mathbb{SP}$):

$$\max \quad \sum_{t\in\mathcal{T}} \Big[\sum_{(n,d)\in\mathcal{A}} \beta_d^T F_{ndt} - \sum_{(s,n)\in\mathcal{A}} \beta_s^T F_{snt} - \sum_{(n,n')\in\mathcal{A}} (\alpha_{nn'}^N x_{nn't} + \beta_{nn'}^N F_{nn't}) \Big] \tag{7.63}$$

s.t.

$$I_{st} = I_{st-1} + F_{st}^{\text{IN}} - \sum_{(s,n)\in\mathcal{A}} F_{snt} \qquad s \in \mathcal{S},\ t \in \mathcal{T} \tag{7.64a}$$

$$I_{dt} = I_{dt-1} + \sum_{(n,d)\in\mathcal{A}} F_{ndt} - FD_{dt} \qquad d \in \mathcal{D},\ t \in \mathcal{T} \tag{7.64b}$$

$$F_{nn't} = \sum_{r\in\mathcal{R}} \tilde{F}_{rnn't} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \tag{7.65a}$$

$$I_{bt} = \sum_{r\in\mathcal{R}} \tilde{I}_{rbt} \qquad b \in \mathcal{B},\ t \in \mathcal{T} \tag{7.65b}$$

$$\begin{bmatrix} X_{nbt} \\ F_{nb}^{\mathrm{L}} \leq F_{nbt} \leq F_{nb}^{\mathrm{U}} \end{bmatrix} \vee \begin{bmatrix} \neg X_{nbt} \\ F_{nbt} = 0 \end{bmatrix} \qquad (n,b) \in \mathcal{A},\, t \in \mathcal{T}, \tilde{YB}_{bt}^{\mathrm{fix}} = True \quad (7.66)$$

$$\begin{bmatrix} X_{sdt} \\ F_{sd}^{\mathrm{L}} \leq F_{sdt} \leq F_{sd}^{\mathrm{U}} \\ C_{qd}^{\mathrm{L}} \leq C_{qs}^{\mathrm{IN}} \leq C_{qd}^{\mathrm{U}} \quad q \in \mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{sdt} \\ F_{sdt} = 0 \end{bmatrix} \qquad (s,d) \in \mathcal{A},\, t \in \mathcal{T} \quad (7.67)$$

$$\begin{bmatrix} X_{bdt} & \\ F_{bd}^{\mathrm{L}} \leq F_{bdt} \leq F_{bd}^{\mathrm{U}} & \\ C_{qd}^{\mathrm{L}} \leq C_{qbt-1} \leq C_{qd}^{\mathrm{U}} & q \in \mathcal{Q} \\ C_{qd}^{\mathrm{L}} F_{bdt} \leq \sum_{r \in \mathcal{R}} \tilde{F}_{rbdt} \hat{C}_{qr}^{0} \leq C_{qd}^{\mathrm{U}} F_{bdt} & q \in \mathcal{Q} \\ C_{qd}^{\mathrm{L}} I_{bt-1} \leq \sum_{r \in \mathcal{R}} \tilde{I}_{rbt-1} \hat{C}_{qr}^{0} \leq C_{qd}^{\mathrm{U}} I_{bt-1} & q \in \mathcal{Q} \end{bmatrix} \vee \begin{bmatrix} \neg X_{bdt} \\ F_{bdt} = 0 \end{bmatrix} \quad \begin{matrix} (b,d) \in \mathcal{A}, t \in \mathcal{T}, \\ \tilde{YB}_{bt}^{\mathrm{fix}} = False \end{matrix} \quad (7.68)$$

$$I_{bt} = I_{bt-1} + \sum_{(n,b) \in \mathcal{A}} F_{nbt} \qquad b \in \mathcal{B},\, t \in \mathcal{T}, \tilde{YB}_{bt}^{\mathrm{fix}} = True \quad (7.69)$$

$$\begin{aligned} I_{bt} C_{qbt} = {} & I_{bt-1} C_{qbt-1} + \sum_{(s,b) \in \mathcal{A}} F_{sbt} C_{qs}^{\mathrm{IN}} \\ & + \sum_{\substack{(b',b) \in \mathcal{A} \\ \tilde{YB}_{b't}^{\mathrm{fix}} = False}} F_{b'bt} C_{qb't-1} \end{aligned} \qquad q \in \mathcal{Q}, r \in \mathcal{R}, b \in \mathcal{B}, t \in \mathcal{T}, \tilde{YB}_{bt}^{\mathrm{fix}} = True \quad (7.70)$$

$$\tilde{I}_{rbt} = \tilde{I}_{rbt-1} + \sum_{(n,b) \in \mathcal{A}} \tilde{F}_{rnbt} \qquad r \in \mathcal{R}, b \in \mathcal{B},\, t \in \mathcal{T}, \tilde{YB}_{bt}^{\mathrm{fix}} = True \quad (7.71)$$

$$I_{bt} = I_{bt-1} - \sum_{(b,n)\in\mathcal{A}} F_{bnt} \qquad b \in \mathcal{B},\ t \in \mathcal{T}, \tilde{YB}^{\text{fix}}_{bt} = False \tag{7.72a}$$

$$C_{qbt} = C_{qbt-1} \qquad q \in \mathcal{Q}, b \in \mathcal{B},\ t \in \mathcal{T}, \tilde{YB}^{\text{fix}}_{bt} = False \tag{7.72b}$$

$$\tilde{I}_{rbt} = \tilde{I}_{rbt-1} - \sum_{(b,n)\in\mathcal{A}} \tilde{F}_{rbnt} \qquad r \in \mathcal{R}, b \in \mathcal{B},\ t \in \mathcal{T}, \tilde{YB}^{\text{fix}}_{bt} = False \tag{7.72c}$$

$$F_{bdt} = 0 \qquad (b,d) \in \mathcal{A},\ t \in \mathcal{T}, \tilde{YB}^{\text{fix}}_{bt} = True \tag{7.73a}$$

$$F_{nbt} = 0 \qquad (n,b) \in \mathcal{A},\ t \in \mathcal{T}, \tilde{YB}^{\text{fix}}_{bt} = False \tag{7.73b}$$

$$I^{\text{L}}_{n} \leq I_{nt} \leq I^{\text{U}}_{n} \qquad n \in \mathcal{N},\ t \in \mathcal{T} \tag{7.74a}$$

$$F^{\text{L}}_{nn'} \leq F_{nn't} \leq F^{\text{U}}_{nn'} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \tag{7.74b}$$

$$FD^{\text{L}}_{dt} \leq FD_{dt} \leq FD^{\text{U}}_{dt} \qquad d \in \mathcal{D},\ t \in \mathcal{T} \tag{7.74c}$$

$$C^{\text{L}}_{q} \leq C_{qbt} \leq C^{\text{U}}_{q} \qquad q \in \mathcal{Q},\ b \in \mathcal{B},\ t \in \mathcal{T} \tag{7.74d}$$

$$I^{\text{L}}_{b} \leq \tilde{I}_{rbt} \leq I^{\text{U}}_{b} \qquad r \in \mathcal{R},\ b \in \mathcal{B},\ t \in \mathcal{T} \tag{7.74e}$$

$$F^{\text{L}}_{nn'} \leq \tilde{F}_{rnn't} \leq F^{\text{U}}_{nn'} \qquad r \in \mathcal{R},\ (n,n') \in \mathcal{A},\ t \in \mathcal{T} \tag{7.74f}$$

$$\tilde{F}_{rsnt}|_{r=s} = F_{snt} \qquad (s,n) \in \mathcal{A},\ t \in \mathcal{T} \tag{7.75a}$$

$$\tilde{F}_{rbnt}|_{r=b} = F_{bnt} \qquad (b,n) \in \mathcal{A},\ t = 1 \tag{7.75b}$$

$$X_{nn't} \in \{True, False\} \qquad (n,n') \in \mathcal{A},\ t \in \mathcal{T} \tag{7.76a}$$

$$YB_{bt} \in \{True, False\} \qquad b \in \mathcal{B},\ t \in \mathcal{T} \tag{7.76b}$$

Note that the summation of streams that contains the bilinear terms in (7.70) only involves the blending tanks that are operating as "discharge" ($\tilde{YB}^{\text{fix}}_{b't} = False$) at a given time period.

The decomposition algorithm is as follows:

0. Specify gap $\epsilon > 0$. Set $UB = \inf$, $LB = -\inf$, $i = 1$, $\mathcal{I}_O = \{\emptyset\}$, and $\mathcal{I}_F = \{\emptyset\}$;

1. Solve ($\mathbb{MP}$). Let $\tilde{YB}_{bt}^{\text{fix}}$ be the value of YB_{bt} at the optimal solution. Let $\hat{yb}_{bt}^{i}$ be the binary representation of the Boolean parameter $\tilde{YB}_{bt}^{\text{fix}}$ (i.e. if $\tilde{YB}_{bt}^{\text{fix}} = True$ then $\hat{yb}_{bt}^{i} = 1$, and if $\tilde{YB}_{bt}^{\text{fix}} = False$ then $\hat{yb}_{bt}^{i} = 0$). Let UB be the value of the optimal objective function.

2. Solve ($\mathbb{SP}$) using $\tilde{YB}_{bt}^{\text{fix}}$ with optimality gap $\epsilon_{SP} \leq \epsilon$.

If ($\mathbb{SP}$) is infeasible, let $i \in I_F$, and go to 3.

If ($\mathbb{SP}$) is feasible, let $i \in I_O$. Let Z^{i*} be the value of the optimal objective function, and Z^i be the upper bound of the objective function. If $Z^{i*} > LB$ then set $LB = Z^{i*}$, let $(I_{nt}^*, F_{nn't}^*, C_{qbt}^*, \tilde{I}_{rbt}^*, \tilde{F}_{rnn't}^*, X_{nn't}^*, YB_{bt}^*)$ be the optimal values of the variables in ($\mathbb{SP}$) and go to 3. If $Z^{i*} \leq LB$ go to 3.

3. If $(UB - LB)/LB \leq \epsilon$, stop with optimal solution $(I_{nt}^*, F_{nn't}^*, C_{qbt}^*, \tilde{I}_{rbt}^*, \tilde{F}_{rnn't}^*, X_{nn't}^*, YB_{bt}^*)$. Else, set $i = i + 1$ and go to 1.

Theorem 7.4.1 *The decomposition algorithm converges to the global optimal solution, within ϵ optimality gap, after a finite number of iterations.*

Proof. Enumeration cut (7.58) guarantees that infeasible solutions are not revisited again by the master problem. Cut (7.57) ensures that if a feasible solution is revisited, then the UB from ($\mathbb{MP}$) equals the upper bound of ($\mathbb{SP}$) for that solution (Z^i). Since $\epsilon_{SP} \leq \epsilon$, then $(UB - LB)/LB \leq \epsilon$. □.

Two phases were implemented for the algorithm. Both phases follow the same steps, but different stopping criteria. In the first phase, the stopping criteria of the master and the subproblem are the maximum execution time and the optimality gap. In the second phase, the optimality gap is the only criterion. The objective of the first phase is to quickly find feasible solutions by limiting the time limit for solving the master and subproblem. Instead of focusing on a region, the algorithm is allowed to move to the next iteration and try a different configuration of tanks after a small amount of time. In the second phase, the objective is to find the optimal solution for the problem within a tolerance. In order to guarantee global optimality, the master and the subproblem have to be solved, at least, to

the specified optimality gap of the algorithm.

7.4.2 Illustration of the algorithm

In order to illustrate the decomposition algorithm, consider the motivating example presented in section 7.2.1. It has an optimal solution of 177.5. The iterations of the algorithm for this example are explained below.

Step 0. The iteration counter is set to $i = 1$. The maximum execution time of the algorithm is set to 30 minutes and the optimality gap is set to 0.01%

Phase 1. The maximum execution time of the master problem is set to 30 seconds and the optimality gap is set to 0.5%. For the subproblem, the maximum execution time is 100 seconds and the optimality gap is 0.5%. The maximum duration of the first phase is 15 minutes and the optimality gap is set to 0.5%

Iteration 1.

Step 1.1: Master problem. The MILP reformulation of the LGDP relaxation of the hybrid formulation ($\mathbb{CSB}$) is solved using CPLEX 12.6. The optimality gap after 4 seconds is below 0.5%. The best possible objective value provided by the solver is an upper bound for the original GDP. The optimal solution is not a true upper bound because the MILP is not solved to the tolerance of the algorithm. The upper bound is set to $UB = 177.8$ and the solution of the master problem is stored for later use.

Step 1.2: Subproblem. The MINLP reformulation of ($\mathbb{CSB}$), in which the operating mode of blending tanks has been fixed according to the solution of the master problem shown in Table 7.11, is solved using SCIP 3.1. The subproblem is feasible with a solution of 177.3. SCIP is able to close the gap to less than 0.5% in a second. The lower bound is set to $LB = 177.3$.

Step 1.3: Stopping criteria. Since the gap between the lower and upper bounds is less than the tolerance of the first phase, gap$= 0.3\% \leq 0.5\%$, the algorithm proceeds to

Table 7.11: Value of Boolean variables YB_{bt} at the relaxed solution.

	YB^*_{1bt}					
Blending tank	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
1	*True*		*True*			
2	*True*		*True*			
3		*True*				
4		*True*				
5		*True*		*True*		
6		*True*		*True*		
7			*True*		*True*	
8			*True*		*True*	

the second phase. The iteration counter is set to $i = 2$.

The gap between the upper bound 177.8 and the lower bound 177.3 is very small. However, for illustration purposes, the phase 2 of the algorithm is presented for the example.

Note that the algorithm only needs one iteration and less than 5 seconds to find a good feasible solution. Neither BARON 14.0, SCIP 3.1 or ANTIGONE 1.1 are able to find a solution in 30 minutes when solving the original MINLP formulation by Kolodziej et al.[79].

Phase 2. The optimality gap for the master and the subproblem is set equal to the tolerance of the algorithm, 0.01%. Time restrictions do not apply in the second phase. Since no cuts are added, the master and the subproblem in the first iteration of the second phase are the same as in the previous iteration, but the optimization has a different stopping criteria.

Iteration 2.

Step 2.1: Master problem. The optimal solution is found after 50 seconds. The new upper bound is $UB = 177.5$. At the solution, the operating modes of the blending tanks are the same as in step 1.2., which means that in the previous iteration CPLEX 12.6 found the optimal solution to the relaxed problem but it did not have time to prove global optimality.

Step 1.2: Subproblem. The GDP is the same as in the previous iteration, which means

that, after finding the same solution of 177.3, SCIP continues the search until the gap is less than 0.01% and the new lower bound increases to $LB = 177.5$.

Step 1.3: Stopping criteria. Since the gap between the lower and upper bounds is less than the tolerance, gap $= 0\%$, the algorithm stops.

In summary, the decomposition algorithm only requires two iterations and less than two minutes to find the global optimum. In fact, a good feasible solution is found in only a few seconds. The correct combination of blending tanks obtained from a tight LGDP relaxation in the master problem and the critical reduction in the number of binary variables and bilinear terms leads to a feasible and more tractable GDP in the subproblem, as will be shown in the next section.

7.5 Computational Results

In this section, we present the computational results of applying the algorithm described in the previous section to several instances. The MINLP reformulation of the GDP problems were also solved with the global optimization solver SCIP 3.1 for comparison. There are rules that could be used for deciding if the algorithm moves from one phase to another or from one iteration to the next. In this study, the stopping criteria used in the master problem, in the subproblem and in the first and second phases, are the same as the stopping criteria used to illustrate the algorithm in section 7.4.2.

48 instances were tested. All instances have eight blending tanks and the same topology. They can be divided in two groups: instances with initial inventory and instances without initial inventory. In each group, all combinations of instances with 1, 2, 5 and 10 specifications and 6 and 8 periods of time were generated. Table 7.12 shows the size of the instances in terms of the number of variables, constraints and bilinear terms. The values of the parameters were generated randomly.

The algorithm and models were implemented in GAMS[21]. All computations were per-

Table 7.12: Size of the instances for the ($\mathbb{C}$) formulation. The number in parenthesis indicates the number of instances in each group.

Instances	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{Q}\rvert$	Binary var.	Bilinear terms	Variables	Constraints
A(6)	6	1	240	128	552	984
B(6)	6	2	240	256	600	1176
C(6)	6	5	240	640	772	1752
D(6)	6	10	240	1280	984	2712
E(6)	8	1	320	176	736	1312
F(6)	8	2	320	352	800	1568
G(6)	8	5	320	889	992	2336
H(6)	8	10	320	1760	1312	3616

formed on a Dell PowerEdge T410 computer with twelve Intel Xeon processors at 2.67 GHz each, 16 GB of RAM, and running Ubuntu Server 14.04 LTS (64-bit).

The decomposition algorithm is able to find at least a feasible solution for 45 out of the 48 instances generated. The three instances that were unsolved had 8 periods and 10 specifications. SCIP 3.1 can only find solutions for 20 instances as was shown earlier in Table 7.10. Figure 7.6 shows the performance of the decomposition algorithm and the MINLP reformulation of ($\mathbb{CSB}$) using SCIP 3.1. The figure shows the average normalized upper and lower bounds. The average normalized lower bound provides the average best objective function value (ANBOFV)

The figure shows that the decomposition algorithm performs better than SCIP. After approximately two minutes, the ANBOFV of the instances solved with the algorithm is close to 0.5, whereas SCIP is below 0.3. As the execution continues, the normalized lower bound keeps increasing. After 600 seconds, the average solution is within 0.1 from the best known objective function value. After 1200 seconds, the average normalized lower and upper bounds are within 0.03 for the decomposition algorithm, while the solutions provided by SCIP are far from the best known solutions. Another important result is the value of the upper bound provided by the master problem, which is practically equal to the best known solution since the first iteration. This shows the tightness of ($\mathbb{CSB}$) when the nonconvex constrains are relaxed.

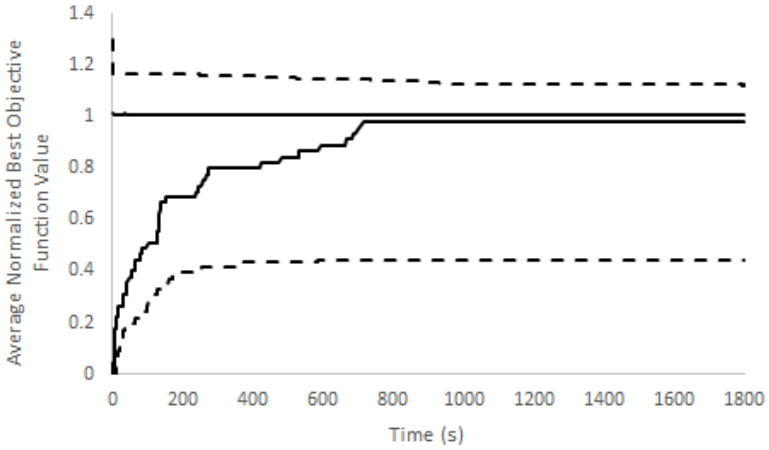

Figure 7.6: Evolution of the average normalized upper and lower bounds, for the decomposition algorithm (solid line) and SCIP 3.1 (dashed line) when tested in 48 instances. The graph contains the 45 instances for which a solution could be found.

Table 7.13 shows the problem size, fraction of blending tanks in charge mode at the solution, the normalized upper bound and the time to get it, of the master problem for the first iteration, all averaged for the 48 instances. Notice that the average time to get a good upper bound for the original GDP problem is less than a minute, only few seconds in some cases. Also, the fraction of tanks that are doing blending is low compared with those that are discharging or in idle mode.

In the subproblem, the decomposition algorithm can find at least a feasible solution to 26 of the instances in the first iteration with a relative lower bound of 0.99. This implies that the values of the Boolean variables representing the mode of operation of the tanks given by the master problem is very close to the optimal solution for half of the instances. Table 7.14 shows the problem size and normalized lower bound and time for the first iteration of the subproblem.

The reduction in the number of binary variables and bilinear terms is essential to the success of the algorithm. On average, the number of binary variables drops 70% when compared to the original MINLP reformulation of the GDP. Similarly, the number of bilinear

Table 7.13: Average values for the Master Problem at the first iteration (MILP reformulation of LGDP relaxation).

Instances	Variables	Const.	Binary Var.	Fraction $YB_{bt} = True$	Normalized UB	Time (s)
A(6)	1584	1896	240	0.30	1.001	5.2
B(6)	1584	2088	240	0.30	1.007	14.9
C(6)	1584	2664	240	0.29	1.007	22.5
D(6)	1584	3624	240	0.31	1.037	22.8
E(6)	2072	2528	320	0.34	1.001	15.6
F(6)	2072	2784	320	0.34	1.007	11.0
G(6)	2072	3552	320	0.34	1.004	24.3
H(6)	2072	4832	320	0.34	1.011	41.3

Table 7.14: Average values for the Subproblem at the first iteration.

Instance	Variables	Const.	Binary Var	Bilinear Terms	Normalized LB	Time (s)
A(6)	857	1776	56	40	0.83	7.2
B(6)	951	2302	59	83	0.66	67.4
C(6)	1058	3880	57	200	0.83	68.3
D(6)	1301	6510	58	405	0.33	68.8
E(6)	1094	2408	69	45	1.00	38.3
F(6)	1186	3130	72	91	0.33	157.2
G(6)	1400	5296	75	235	0.32	207.8
H(6)	1729	8906	76	470	0.00	177.3

terms decreases by 70%. These reductions are the main reasons why the MINLP global solver used in the subproblem can find feasible solutions. Note that the lower bound is very close to the best known objective function value for those instances with 6 time periods and 1, 2 and 5 specifications. However, the value of the feasible solution for instances with 10 specifications is not as good. When dealing with 8 time periods, only those with a single specification have good lower bounds. Nevertheless, the solution for those instances with 2 and 5 specifications is still good considering that it is the first iteration.

In conclusion, the decomposition algorithm performs better than SCIP 3.1 for the 48 instances generated. The algorithm is able to find a good feasible solution for 45 of the instances in less than two minutes. The tightness of the relaxation of the master problem and the reduction in the number of binary variables and bilinear terms in the subproblem are key to the success of the algorithm.

7.6 Conclusions

In this chapter, we have addressed the multiperiod blending problem, which frequently arises in the petroleum and petrochemical industry. Our main goal has been to develop new formulations and new algorithms for obtaining good feasible solutions in few minutes. We have presented two principal contributions towards solving multiperiod blending problems more effectively.

First, we have presented a source based formulation. The sources in a system are the supplies and initial inventories. They can be interpreted as raw materials of known composition. The model uses flow and inventory variables to track down each one of the sources along the network. The notion of split fraction is used to guarantee that the outflows from a tank have the same composition. These are the only nonlinearities in the model. The composition of a stream is determined from the compositions of each of the sources present in the stream. Since the latter are parameters in the model, the specification requirement constraints are linear. Lastly, we found redundant linear constraints that can be added to this model in order to improve its relaxation. In the context of a branch-and-bound search, this

speeds up the convergence by reducing the number of open nodes. It was shown that the number of instances for which a feasible solution can be found using a global optimization solver increases when adding the redundant constraints (from 31 % to 41 % using SCIP 3.1)

Second, we have proposed a solution procedure that takes advantage of the operational assumption of non-simultaneous inlet/outlet streams in the blending tanks. Under this assumption, we can think of two non-coincident modes of operation for each blending tank at any time period: charge mode or discharge mode. This restriction can be modeled using disjunctions. The GDP formulation leads to a reduction in the number of bilinear terms and generates a favorable structure that can be exploited in a decomposition algorithm. Thus, an iterative two-stage MILP-MINLP decomposition method for the global optimization of the multiperiod blending problem is proposed. The first stage, or master problem, is a linear GDP relaxation of the original GDP and provides rigorous upper bounds. The second stage, or subproblem, is a smaller GDP in which the set of the binary variables representing the modes of operation for the blending tanks is fixed accordingly to the solution of the master problem. The subproblem, when a feasible solution is found, provides a feasible solution to the original GDP and a rigorous lower bound. These problems are solved successively until the gap between the upper and lower bound is closed.

The decomposition algorithm was tested in 48 instances and compared against the global optimization solver SCIP 3.1. The results show that the algorithm performs better than SCIP 3.1. In fact, the algorithm is able to find feasible solutions for 45 out of the 48 instances, whereas SCIP could only find solutions for 20 instances. Feasible solutions are obtained in less than two minutes. After less than 15 minutes, the solutions obtained with the algorithm are within 3% of the best known solutions, whereas the solutions provided by SCIP are at around 60% from the optimal values. The tightness of the relaxation of the source based formulation when the nonconvex constraints are relaxed is reinforced by the values of the upper bound given by the master problem. They are practically equal to the best known objective function value since the first iteration. The better performance of the algorithm when compared with SCIP can be explained by the reduction in the number of binary variables and bilinear terms in the subproblem.

Chapter 8

Conclusions

8.1 Summary of thesis

In this section we summarize the major findings and accomplishments in each chapter.

8.1.1 Improved Big-M reformulation for GDP problems

In chapter 2 we have proposed a new MILP/MINLP reformulation for GDP problems, namely the multiple-parameter Big-M (MBM). This new reformulation is similar to the BM but, instead of using one M-parameter for each constraint, it uses several. This reformulation can provide stronger continuous relaxations than the traditional BM in disjunctions with more than 2 disjunctive terms. In the traditional BM, the M-parameter must be large enough to be valid, but as small as possible to be tight. In addition to using the physical meaning of the constraints, optimal and good M-parameters of a constraint can be obtained by solving optimization problems. In section 2.2 we present two optimization problems to obtain such parameters: (2.1) which is a GDP, and (2.2) which is an LP or NLP depending on the original GDP.

In MBM, instead of having one M-parameter when a disjunctive term is not active, there

is a different M-parameter if a different disjunctive term is active. The new formulation (MBM) involves the same number of variables and constraints as (BM). Also, as proven in Theorem 2.3.1, (MBM) is at least as tight as (BM). For example, in the illustrative minimization example (2.7) the optimal objective function is -9.472, the continuous relaxation of (BM) gives a value of -10.493, and the continuous relaxation of (MBM) gives -9.735. Clearly, the optimal objective value of the continuous relaxation of (MBM) is much better. The new reformulation was tested with instances of the process network problem and the design of multi-product batch plant problem.

For the four instances of the process network problems, the lower bound of the continuous relaxation is improved considerably. For example, in instance Proc-1-21 the optimal objective function is 17.2, the continuous relaxation of (BM) gives a value of 1.7, and the continuous relaxation of (MBM) gives 10.0. The time to solve process network problems is reduced by 50% to 90% when using (MBM) instead of (BM). For example, instance Proc-1-31 is solved in 1.2 seconds, instead of 6.5; and instance Proc-1-48 is solved in 5.7 seconds instead of 63.7. The number of nodes is also reduced by 50% to 90% when using (MBM) instead of (BM).

In the 6 instances of the multi-product batch plant problem, the continuous relaxation is the same. However, the (MBM) formulation requires fewer nodes than the (BM) in 5 of the 6 instances. In instance BatchS101006 the difference is drastic, since the (BM) requires $10,894$ nodes and the (MBM) only $1,595$. In other instances the difference is smaller. For example, in the largest instance (BatchS201210), the (BM) requires $13,774$ nodes and the (MBM) $10,158$. There is only one instance in which the (BM) requires fewer nodes: instance BatchS151208. For this example, (MBM) solves faster in 4 instances, ties in 1 instance, and is slower in 1 instance.

It is clear that the MBM can be better than the BM in disjunctions with more than two disjunctive terms. However, one of the main limitations of this reformulation is that in order to be better than the BM it requires good M-parameters. Simply using valid M-parameters that are bad (i.e. very large) will typically yield reformulations in which the BM and MBM have the same continuous relaxation. While problem knowledge is a good alternative for obtaining good M-parameters, it requires input from the modeler into the reformulation.

Solving (2.2) provides an automated method for obtaining good M-parameters. However, this method can be expensive and, in some cases, it may require more time to obtain good M-parameters than to solve the original problem.

8.1.2 Algorithmic approach for improved mixed-integer reformulations of convex GDP problems

In chapter 3 we have proposed an algorithmic approach to exploit the advantages of the BM and HR reformulations for convex GDP problems. In particular, the algorithms obtain formulations that have tight continuous relaxations with a small increase in the number of variables and constraints. There are three main stages in this method: (a) Feasibility check in individual disjunctions through the solution of continuous programs; (b) application of basic steps in specific disjunctions; (c) hybrid BM/HR reformulation of the GDP.

The algorithm was tested with 36 instances. The results show that the algorithm generates formulations with better continuous relaxations, while having a small increase in problem size. In 28 of the 36 the relaxation after applying the algorithm provides better bounds than the relaxation of the HR, and in the remaining 8 both provide the same bounds. In some cases, the algorithm provides a relaxation that is actually the optimal value of the objective function. For example, the solution to the continuous relaxation of instance Process-8 is $1,098$, which is the value of the optimal solution to the problem. In most cases, the gap improves around 20%-40% compared to the HR. For example, in C-Lay-5-2 the solution is $11,472$, the (BM) and (HR) relaxations are 0, and the relaxation after the algorithm is $4,203$. The HR generates larger MILP/MINLP reformulations than the algorithm in 25 of the 36 instances. In most of the instances in which the algorithm generates larger instances, they are not much larger. For example, in instance C-Lay-5-4 the MINLP after the algorithm has 392 variables and 952 constraints, while the HR of the original GDP generates an MINLP with 292 variables and 608 constraints. Note that for this instance the lower bound of the HR is 0, and the MINLP after the algorithm is $2,695$ (the optimal solution is $10,876$). Over all instances, the algorithm leads to a reduction in solution times as presented in Figure 3.4.

The results in chapter 3 show that the proposed algorithm can provide improved MINLP reformulations for convex GDP problems. However, there is a downside to this algorithm. The algorithm iteratively applies basic steps in the "key disjunction". Therefore, the selected disjunction grows exponentially with the iterations. Also, if a basic step that generates no improvement is applied, the disjunction grows exponentially without providing any improvement. Furthermore, the final formulation will also increase in size without providing improvement in the relaxation. In chapter 3 we have proposed heuristics for the selection of basic steps, but different problems will perform better with different heuristics. The downside of the algorithm is that its performance strongly depends on the selected disjunctions to which the basic step is applied (i.e. it strongly depends on the heuristics in the application of basic steps).

8.1.3 Cutting plane algorithm for convex GDP

In chapter 4 we have presented an alternative method for exploiting the advantages of the BM and the HR after basic steps. The algorithm generates the cuts by solving a separation problem (NLP problem (4.3)). The NLP finds a point in the continuous relaxation of the tight formulation that minimizes the distance to the optimal solution of the continuous relaxation of the BM. If the point obtained by the separation problem is different from the optimal solution of the continuous relaxation of the BM, it is possible to obtain a valid inequality that cuts-off the optimal solution of the continuous relaxation of the BM. The derivation of this cut has been presented in Propositions 4.2.4 and 4.2.5 (Proposition 4.2.6 when the norm-2 is used in (4.3)). The resulting MINLP after deriving cuts has the same number of variables than the BM, and the number of constraints is the same as the BM plus the number of cuts. In chapter 4 the cuts have been derived only at the root node, but it is possible to apply this method at different nodes in a disjunctive branch and bound.

The algorithm was tested with 19 nonlinear convex GDP problems, using different strategies for the selection of basic steps and number of cuts. The resulting MINLP after 3 cuts, with the strategy $K5$-$I1$ described in chapter 4, was shown to be better than the BM and HR for the tested instances. In particular, its continuous relaxation provides better bounds

than the HR in 12 of the instances and the same bound in 7, and it generates smaller MINLP models than the HR in every instance. Furthermore, its continuous relaxation is better than that of the BM in every instance, while having the same number of variables and only 3 more constraints. An example of the improved MINLP is instance Clay54. The optimal solution is $10,876$, the solution to continuous relaxation of the BM and HR is 0, while the continuous relaxation of the BM after the cuts yields a value of $4,417$. The number of variables for the BM, HR, and BM after cuts are 32, 232, and 32, respectively. The number of constraints for the BM, HR, and BM after cuts are 178, 608, 181, respectively.

The improved formulations typically result in improved solution times. For example, in the same instance (Clay54), the solution times for solving the MINLP that results from the BM, HR, and BM after cuts are $2,075$ s, $5,175$ s, and 306 s. The time to generate the cuts in this instance is 34 s, so the improvement in total solution time is considerable.

8.1.4 Lagrangean relaxation of the HR of linear GDP problems and its use in the disjunctive branch and bound

In chapter 5 we have proposed a Lagrangean relaxation that can be applied to any linear GDP. The Lagrangean relaxation is an MILP that can be solved as an LP, as was proven in Property 5.3.3. Furthermore, we had proven in Property 5.3.2 that it can be solved by solving several small LP problems in parallel. While the proposed Lagrangean relaxation can be used in different ways to improve GDP solution methods, in chapter 5 we have explored its use as primal heuristic in a disjunctive branch and bound.

We have tested the proposed modified disjunctive branch and bound with 300 instances, 100 of each of the 3 test problems. Over all instances, the proposed algorithm (ALG) solves more instances in 2 hours than three alternative disjunctive branch and bound methods: BM, HR, and BM with random primal heuristic (RAN). Of the 300 instances, the BM, HR, RAN, and ALG solve 253, 266, 269, and 296, respectively. For the 3 test problems, the performance in terms of number of nodes is better for the proposed algorithm than for the rest.

In terms of solution time, the performance varies by problem. The proposed algorithm and the heuristic one are the best for the unstructured GDP instances. They solve more instances than the rest within the two hour time limit; 98% for the proposed and heuristic algorithms vs. 81% for the BM, and 72% HR. For the strip packing problem, the HR disjunctive branch and bound performs the best, while the proposed algorithm performs third in terms of solution time. For the contracts problem, the proposed algorithm performs the best by solving 98% of the instances, while the BM, HR, and random heuristic algorithm solve 72%, 94%, and 71%.

The proposed Lagrangean relaxation has important properties and can be useful in improving GDP solution methods. However, the current implementation proposed disjunctive branch and bound is still a prototype. Its performance in considerably worse than commercial LP-based branch and bound methods. In order for the disjunctive branch and bound to be competitive, it is necessary to improve the algorithmic implementation as well as to extend the algorithm to nonlinear convex problems.

8.1.5 Cutting planes for improved global logic-based outer- approximation of nonconvex GDP problems

In chapter 6 we have presented a global logic-based outer-approximation (GLBOA), which terminates in a finite number of iterations, as was shown in Theorem 6.2.1. The general idea of the algorithm is to have a linear master GDP that overestimates the feasible region of the GDP. This master problem provides a valid lower bound (in a minimization problem), and the selection of only one disjunctive term in each of the disjunctions. With the alternative provided by the master problem, an NLP subproblem is solved to global optimality. After solving this subproblem, infeasibility or optimality integer cuts are added to the master problem. In addition to the basic GLBOA, we have proposed two enhancements to the algorithm. The first enhancement is to partition the method into two phases. This partition allows finding feasible solutions faster. We further improve the first phase by including a penalty cost in the objective function that drives diversity in the finding of feasible solutions. The second enhancement is a cutting plane method to improve the

lower bounding of the algorithm. The cuts are obtained by solving problem (6.5), which is an NLP, to global optimality. The cuts are valid for the original GDP, as presented in Theorem 6.3.1.

The algorithm was tested with 20 instances of layout-optimization of screening systems in recovered paper production, 20 of reactor-separator process superstructure, and a more realistic problem of designing a distillation column for the separation of benzene and toluene with ideal equilibrium. The results show that the algorithm with the different enhancements perform better than commercial general purpose global optimization solvers. They also show that the different enhancements are important in the overall performance of the algorithm.

The GLBOA without any enhancement can be useful in some problems. For example, in the reactor-separator process superstructure BARON 14.0.3 can find a feasible solution in only 2 of the 20 instances, while the basic algorithm finds a feasible solutions in 12 of the instances. In some cases, the basic GLBOA is not useful. For example, in the layout-optimization of screening systems in recovered paper production problem BARON performs better than the basic GLBOA.

The two phases can considerably improve the performance of the GLBOA in some problems. For example, in the design of a distillation column for the separation of benzene and toluene the basic algorithm does not find a solution while the two-phase modified versions finds the best-known solution. The additional enhancement of favoring diverse solutions by a penalty in the objective function is also beneficial in some problems. For example, in the reactor-separator process superstructure the two-stage algorithm without a penalty finds a feasible solutions in 12 of the instances, but the two-stage version with the penalty finds a feasible solution in the 20 instances.

The cutting plane enhancement can considerably improve the lower bound of the master problem when the linear relaxation is obtained by dropping the nonlinear constraints. For example, in the reactor-separator process superstructure the average relative lower bound of the master problem without nonlinear constraints is 0.6. However, by including the cuts the algorithm improves and provides a relative lower bound of 0.9. In the tested examples,

the cuts do not improve much the lower bound of the algorithm when linear overestimators are used to relax the nonlinear constraints.

8.1.6 GLBOA for the global optimization of a source based model of the multiperiod blending problem

In chapter 7, we have addressed the multiperiod blending problem, which frequently arises in the petroleum and petrochemical industry. We have presented two principal contributions towards solving multiperiod blending problems more effectively: a source based model and the application of the GLBOA in its solution.

In Theorem 7.3.1, the source based GDP was proven to be tighter than the split fraction GDP. Furthermore, Theorem 7.3.2 shows that linear constraints from the source based model can be added to the total flow and composition model in order to improve its relaxation. For the MINLP reformulation of the GDP, the number of instances for which a feasible solution can be found using a global optimization solver increases when adding the redundant constraints. BARON 14.0 finds feasible solution in 29% of the instances when including these redundant constraints, and only in 21% when they are not included. ANTIGONE 1.1 improves to 31% from 29%. Finally, SCIP 3.1 finds feasible solutions in 42% of the instances when including the redundant constraints, and only in 31% without them.

For the second contribution, we have applied a modified version of the GLBOA from chapter 6 to solve the GDP with the source based redundant constraints. The results show that the algorithm performs better than solving the MINLP reformulation of the original GDP. The comparisons are presented against SCIP 3.1, which performs better than the other general purpose global solvers for this problem. The algorithm finds feasible solutions for 45 of the 48 instances, whereas SCIP only finds solutions for 20 instances. The algorithm finds a feasible solution in 26 of the 48 instances in the first iteration, which takes less than 4 minutes.

The developments in this chapter allow to solve problems with up to 4 blending tanks,

10 qualities and 8 time periods. These parameters are a considerable improvement to previous work in which the largest solvable instances involved 4 blending tanks, 2 qualities and 4 time periods[79]. However, the numbers still fall short for real-operation problem sizes. Gasoline blending involves 4 blending tanks and around 10 qualities, which is close to the instances solved in this chapter. However, it typically requires the solution of 20 time periods. The difference is even larger in crude oil blending problems, which involve instances with over 40 tanks and 20 time periods.

8.2 Research contributions

The major contributions of this thesis can be summarized as follows:

1. Proposed a new Big-M reformulation for GDP problems, that is at least as tight as the traditional Big-M and requires the same number of variables and constraints.
2. Proposed a heuristic for the application of basic steps, which includes the number of terms in the disjunction and its characteristic value (Definition 3.2.1).
3. Developed an algorithm to improve the MILP/MINLP reformulation of convex GDP problems. This algorithm makes use of basic steps and a hybrid BM/HR reformulation of GDP problems.
4. Proposed a separation problem to obtain valid inequalities for the Big-M reformulation of convex GDP problems. The inequalities cut off points that are valid for the continuous relaxation of the BM, but not for the continuous relaxation of the HR after the application of basic steps.
5. Proved that the cutting planes are valid, and applied them to improve the BM reformulation of convex GDP problems.
6. Proposed a Lagrangean relaxation of the HR of linear GDP problems. Proved that such a relaxation, which is an MILP, can be solved by solving several small LP problems in parallel.

7. Implemented a prototype disjunctive branch and bound, in which the proposed Lagrangean relaxation is used as a primal heuristic.

8. Proposed the use of a global logic-based outer-approximation for the solution of nonconvex process design problems. Showed through several instances that the partition of the algorithm into two phases helps to find feasible solutions faster.

9. Developed a novel separation problem to obtain cutting planes that improve the linear relaxation of nonconvex GDP problems. The cutting planes cut off points that are outside of the convex hull of the feasible region of disjunctive terms.

10. Presented a new GDP model for the multiperiod blending problem using redundant constraints from a source-based model that provides stronger linear relaxations than other models.

11. Applied a modified version of the GLBOA that finds feasible solutions for the multiperiod blending problem faster than using general purpose global algorithms in the MINLP reformulation of the GDP model.

8.3 Future research directions

8.3.1 Improve heuristics for the application of basic steps

While there are some rules on when not to apply basic steps, heuristics to decide which disjunction to intersect are needed in practical implementations. One such heuristic is presented in Section 3.2.3. This heuristic is good for packing problems (strip packing, constraint layout, and farm layout). In particular, the original disjunctions in these types of problems establish the relative position between two rectangles. By using the heuristic in Section 3.2.3, the basic steps are applied first over all of the disjunctions that establish the relative position among 3 or more rectangles. For example, if the first selected disjunction was the one corresponding relative position of rectangle A and B, then the second disjunction selected will involve either A or B. In this example, consider that the second

selected disjunction corresponds to the relative position of B and F, then the third selected disjunction will be the one that establishes the relative position of A and F. Thus, in this example, the resulting disjunction completely establishes the relative positions among A, B, and F. The rule in Section 3.2.3 also selects the "bigger rectangles".

While this rule is useful for packing problems, it is not necessarily so for other problems. The heuristics for the application of basic steps have to be based on problem structure. Developing good heuristics for different types of problems would be beneficial for the different methods that make use of basic steps.

8.3.2 Include cutting planes in a disjunctive branch and bound

The application of the cutting planes presented in chapter 4 to a convex GDP considerably helped the MINLP reformulation in many cases. This method can be considered as the application of cutting planes at the root note. However, these cutting planes can be further applied in subsequent nodes in a disjunctive branch and bound. The use of these cuts in different nodes of the disjunctive branch and bound can help to prune nodes by improving the relaxation. This direction is a natural extension to the work presented in this thesis, that combines the work of chapters 4 and 5.

8.3.3 Extend the Lagrangean relaxation to nonlinear convex GDP

The Lagrangean relaxation presented in chapter 5 can be extended to nonlinear convex GDP. The first two properties still hold: it can be applied to any convex GDP problem, and it can be separated into several smaller problems. However, the solution to the Lagrangean relaxation will not necessarily yield 0-1 values to the binary variables. To prove that there exists an optimal solution to the Lagrangean relaxation that assigns 0-1 values to the binary variables, it is necessary to extend Corollary 2.7 from Ruiz and Grossmann[15]. If Corollary 2.7 is extended for this problem, then an integer solution to the Lagrangean relaxation can be obtained by applying simple algebra to any non-integer solution of the Lagrangean relaxation.

Extending the Lagrangean dual to nonlinear convex GDP problems will allow its use as a primal heuristic in a nonlinear disjunctive branch and bound. Considering the positive results obtained for the linear case, it seems like an important research direction.

8.3.4 Embed logic constraints in the branching decisions of the disjunctive branch and bound

Using the logic propositions in GDP for the branching strategy can lead to significant reductions in solution times. Specifically, it can allow the pruning of nodes without requiring to solve LP/NLP problems. Integrating logic propositions with optimization can be achieved through resolution techniques[64]. Extensive work in this area has been done by Hooker[27].

GDP provides a modeling framework that is compatible with the integrated methods developed by Hooker[27]. Furthermore, the disjunctive branch and bound as solution method for GDP problems can greatly benefit form these integrated methods. Current GDP formulations accept Boolean logic in the logic propositions. Extending GDP form to include even more general type of constraints can also be fruitful.

8.3.5 Improve the implementation of the prototype disjunctive branch and bound

The current implementation of the branch and bound presented in chapter 5 is a prototype. The algorithm is coded in GAMS, and it requires the generation of an LP/NLP at every single node. It also requires the generation of many nodes, with their corresponding information, that are pruned because the parent node has a value that is worse than the best known integer solution. In addition, the inclusion of the logic propositions in the branching strategy is inefficient and complicated using GAMS. For these reasons, another programming language, as well as improvements in the code, are required to obtain a competitive algorithm.

8.3.6 Test the GLBOA with more realistic design problems

In chapter 6, the GLBOA is tested with several instances of two toy problems and with one instance of a more realistic one. In the realistic instance, the two-phase version of the algorithm performs well in terms of finding good feasible solutions. However, the cutting planes derived in this chapter show no effect in this example. Testing the GLBOA and its corresponding enhancement in more instances of realistic problems will help to prove the value of this approach.

8.3.7 Extend the cutting plane method in GLBOA to nonconvex NLPs

In chapter 6, separation problem (6.5) allows the derivation of valid cutting planes in nonconvex feasible regions (that correspond to a disjunctive term). This method can be applied in general to any nonconvex NLP problem. Performing the separation problem to the full NLP is not practical, since the separation problem would become harder to solve than the original NLP. However, it is possible to select a subset of the nonconvex constraints and use a convex relaxation of the remaining ones.

In some cases, generating valid cutting planes for a nonconvex NLP can help to solve the problem to global optimality faster. In particular, when the convex relaxation of the NLP is performed, these valid cutting planes may provide tighter relaxations. Therefore, some regions in the spatial branch and bound are likely to be pruned at earlier stages of the algorithm. The theory for this extension needs to be developed an proved, and computational results are required to show that the strategy can be useful in some instances.

8.3.8 Extend the implementation of GDP models and tools in optimization modeling software

In recent years, GDP as a modeling framework has been increasingly used in process systems engineering[16]. However, GDP has not been as successful in other areas of opera-

tions research. The adequate implementation of GDP as a modeling language in existing software would greatly increase the exposure of GDP in different areas.

Modeling languages such as GAMS[21] and PYOMO[106] have already provided useful progress in this regard. The extended mathematical programming framework (EMP) in GAMS reformulates GDPs as MILP/ MINLPs, allowing the user to directly model GDPs and to specify the use of either the BM or the HR reformulation. Furthermore, LOGMIP[107] is a code in C that not only reformulates GDPs to MILP/ MINLPs, but also allows the logic-based outer-approximation as solution method (as an alternative to direct reformulation). LOGMIP can be used as an stand-alone code or through GAMS. Similarly, PYOMO allows the modeling of problems as GDPs. The algorithms presented in chapters 3 and 4 are available in PYOMO. Therefore, the user can obtain the BM, HR, or the improved formulations (presented in chapters 3 and 4) by changing a single line of the code in PYOMO.

The advantages of having multiple models by simple changes in a code are clear. Furthermore, the additional tools and advantages of solving GDP problems can be exploited using these frameworks. These implementations are incomplete in terms of solution methods, and they are still not very intuitive. Improving these implementations will help to boost the use of GDP as modeling framework, generating more interest in GDP. As a result, GDP will benefit from theoretical developments and improvements in existing algorithms.

8.3.9 Extend the GDP framework to include different types of variables and constraints

Although the GDP framework allows the representation of a wide-range of discrete - continuous problems, in some cases it is convenient to represent constraints using other frameworks. For example, the modeling of assignment constraints or of knapsack constraints can be easily achieved with MILP. Furthermore, these types of constraints can be solved efficiently as MILPs. Representing them through the current GDP framework would involve additional effort and perhaps result in inefficient models.

The use of different types of constraints and variables will result in more efficient methods

and algorithms. While a first natural step is to involve discrete variables in the current GDP framework, other types of constraints can also be explored.

8.4 Papers produced from this dissertation

Grossmann, I. E., Trespalacios, F. *Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming.* AIChE Journal. 2013; 59(9), 3276-3295.

Trespalacios, F., Grossmann, I. E. *Review of mixed-integer nonlinear and Generalized Disjunctive Programming methods.* Chemie Ingenieur Technik. 2014; 86(7), 991-1012.

Trespalacios, F., Grossmann, I. E. *Algorithmic approach for improved mixed-integer reformulations of convex Generalized Disjunctive Programs.* INFORMS Journal on Computing. 2014; 27(1), 59-74.

Trespalacios, F., Grossmann, I. E. *Cutting plane algorithm for convex Generalized Disjunctive Programs.* INFORMS Journal on Computing. 2014; Accepted for publication.

Trespalacios, F., Grossmann, I. E. *Lagrangean relaxation of the Hull-Reformulation of linear Generalized Disjunctive Programs and its use in disjunctive branch and bound.* European Journal of Operational Research. 2015; Submitted for publication.

Trespalacios, F., Grossmann, I. E. *Cutting planes for improved global logic-based outer-approximation for the synthesis of process networks.* Computers & Chemical Engineering. 2015; Submitted for publication.

Lotero, I., Trespalacios, F., Papageorgiou, D. J., Cheon M., Grossmann, I. E. *An MILP-MINLP decomposition method for the global optimization of a source based model of the multiperiod blending problem.* Computers & Chemical Engineering. 2015; Submitted for publication.

8.5 Short notes produced from this dissertation

Trespalacios, F., Grossmann, I. E. *Improved Big-M reformulation for Generalized Disjunctive Programs.* Computers & Chemical Engineering. 2015; 76(8), 98103.

Trespalacios, F., Grossmann, I. E. *Symmetry breaking for Generalized Disjunctive Programming formulation of the strip packing problem.* Annals of Operations Research. 2015; Submitted for publication.

8.6 Book chapters produced from this dissertation

Trespalacios, F., Grossmann, I. E. *Review of mixed-integer nonlinear and Generalized Disjunctive Programming applications in process systems engineering.* SIAM. 2014; submitted for publication.

Appendix A

GDP formulations for the strip packing problem

A.1 Introduction

The two-dimensional strip packing is a special case of "Cutting and Packing" problems that arises in many applications. The problem seeks to pack a given set of rectangles into a strip of given width in order to minimize its length. Some applications in which the strip packing problem arises are: Cutting pieces from wooden boards, cutting pieces from glass or steel sheets, optimal layout of industrial facilities, etc. Lodi et al.[108] present a comprehensive survey of applications and methods. Following the typology proposed by Wascher et al.[109] for cutting and packing problems, the two-dimensional strip packing is classified as two-dimensional open dimension problem (2-ODP).

Although there exist several heuristic and meta heuristic algorithms for solving the problem[110], exact algorithms have been proposed for the solution of the two-dimensional strip packing. Martello et al.[111] present a branch and bound algorithm. The lower bound of the nodes in this method is obtained considering the geometry of the problem. Alvarez-Valdes et al.[112] improve the branch and bound method of Martello et al. by obtaining stronger

lower bounds and deriving new dominance conditions.

In addition to specialized algorithms, the two-dimensional strip packing problem has been formulated as a mixed-integer linear program (MILP). Westerlund et al.[113] present an MILP model for the N-dimensional allocation, which includes the strip packing problem. Castro and Oliveira[114] present two different MILP formulations for the problem, based on scheduling models. One formulation follows a discrete-space approach, and it is based on the Resource-Task Network process representation. The other formulation uses a hybrid continuous/discrete representation, in which the width of the strip is discrete and the height is treated as continuous.

In this section, we present a GDP model for the strip packing problem. This model, which is a modification of the model by Sawaya and Grossmann[29], uses additional constraints within the disjunctions in order to break symmetry in the solutions. The model is further improved for the case in which the heights and lengths of the rectangles are integer (which is true in general with a simple transformation, as long as the lengths and heights of the rectangles are rational numbers). The model is reformulated with the three GDP-to-MILP schemes (BM, MBM, and HR) and solved as an MILP. The new formulations with the alternative GDP-to-MILP reformulations are tested with 100 random instances. The results show that the new symmetry-breaking formulation is solved faster than the formulation by Sawaya and Grossmann. The results also show that the BM reformulation solves faster than the other GDP-to-MILP reformulations (MBM and HR).

A.2 GDP formulation of the two-dimensional strip packing problem

The two-dimensional strip packing problem consists on placing a given set of N rectangles in a strip. The height and length of each rectangle is known ($H_i, L_i; i \in N$), and the strip has width W. The rectangles cannot be rotated. The objective is to minimize the total length of the strip. Figure A.1 illustrates the strip packing problem.

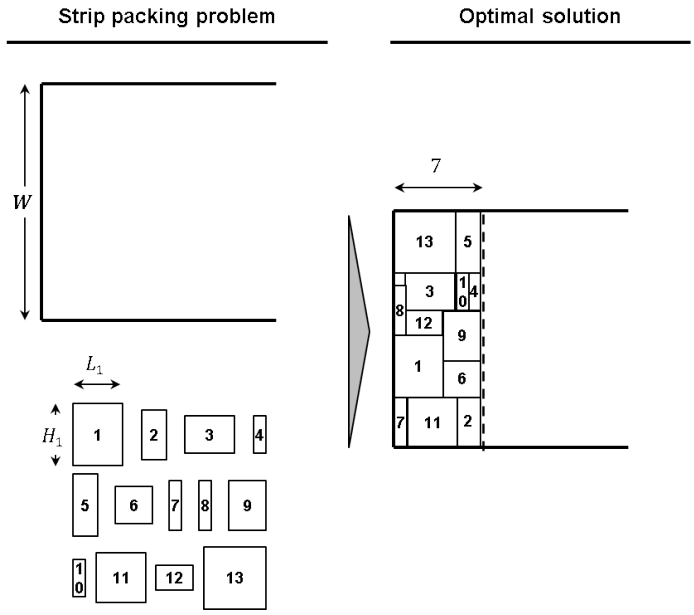

Figure A.1: Illustration of the two-dimensional strip packing problem.

The GDP formulation of this problem, presented by Sawaya and Grossmann[29], is as follows:

$$
\begin{aligned}
& & & min\ lt & & \\
& s.t. & & lt \geq x_i + L_i & & i \in N \\
& & & \begin{bmatrix} Z^1_{ij} \\ x_i + L_i \leq x_j \end{bmatrix} \vee \begin{bmatrix} Z^1_{ji} \\ x_j + L_j \leq x_i \end{bmatrix} & & \\
& & & \vee \begin{bmatrix} Z^2_{ij} \\ y_i - H_i \geq y_j \end{bmatrix} \vee \begin{bmatrix} Z^2_{ji} \\ y_j - H_j \geq y_i \end{bmatrix} & i,j \in N, i<j & \quad \text{(SG)} \\
& & & Z^1_{ij} \veebar Z^1_{ji} \veebar Z^2_{ij} \veebar Z^2_{ji} & i,j \in N, i<j & \\
& & & 0 \leq x_i \leq UB - L_i & i \in N & \\
& & & H_i \leq y_i \leq W & i \in N & \\
& & & Z^1_{ij}, Z^2_{ij} \in \{True, False\} & i,j \in N, i \neq j &
\end{aligned}
$$

In (SG), the continuous variables (x_i, y_i) represent the coordinates of the upper-left corner

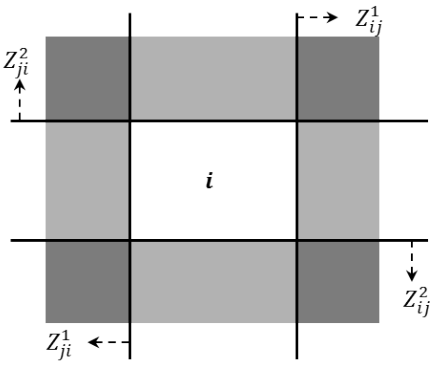

Figure A.2: Illustration of the relative position of j with respect to rectangle i.

of rectangle i. The objective is to minimize the distance lt. The global constraints ($lt \geq x_i + L_i$) enforce that the total distance is greater than the x coordinate of the right edge of each rectangle. The logic variables indicate the relative position between two rectangles: $Z^1_{ij} = True$ if rectangle i is to the left of rectangle j, $Z^1_{ji} = True$ vice versa; $Z^2_{ij} = True$ if rectangle i is on top of rectangle j, $Z^2_{ji} = True$ vice versa. The disjunction establishes the four possible relative positions between each pair of rectangles (i.e. rectangle i is to the left, right, top or bottom of rectangle j). Exactly one alternative must be selected ($Z^1_{ij} \veebar Z^1_{ji} \veebar Z^2_{ij} \veebar Z^2_{ji}$). For the selected alternative the corresponding constraints are enforced (e.g. if $Z^1_{ij} = True$ then $x_i + L_i \leq x_j$ is enforced). For the terms not selected (e.g. $Z^1_{ji} = False$) the corresponding constraints are ignored. The continuous variables have upper and lower bounds, where UB is an upper bound for the length of the strip (a simple upper bound can be obtained with $UB = \sum_i L_i$).

Figure A.2 represents the disjunction in the formulation. The figure shows the possible positions for a rectangle j with respect to a rectangle i. Note that there are two alternative decisions that are feasible for the darker regions of the figure. For example, if j is both above and to the right of i it is possible to set $Z^1_{ij} = True; Z^1_{ji} = Z^2_{ij} = Z^2_{ji} = False$ or $Z^2_{ji} = True; Z^1_{ji} = Z^1_{ji} = Z^2_{ij} = False$. If $Z^1_{ij} = True$ the constraints corresponding to Z^2_{ji} are ignored (which means that they may or may not be satisfied).

For this problem, the BM reformulation is as follows:

$$
\begin{aligned}
& & & min\ lt & & \\
& s.t. & & lt \geq x_i + L_i & & i \in N \\
& & & x_i + L_i \leq x_j + UB(1 - z^1_{ij}) & & i,j \in N, i < j \\
& & & x_j + L_j \leq x_i + UB(1 - z^1_{ji}) & & i,j \in N, i < j \\
& & & y_i - H_i \geq y_j - W(1 - z^2_{ij}) & & i,j \in N, i < j \\
& & & y_j - H_j \geq y_i - W(1 - z^2_{ji}) & & i,j \in N, i < j \\
& & & z^1_{ij} + z^1_{ji} + z^2_{ij} + z^2_{ji} = 1 & & i,j \in N, i < j \\
& & & 0 \leq x_i \leq UB - L_i & & i \in N \\
& & & H_i \leq y_i \leq W & & i \in N \\
& & & z^1_{ij}, z^2_{ij} \in \{0,1\} & & i,j \in N, i \neq j
\end{aligned}
\tag{SG-BM}
$$

In (SG-BM), the Boolean variables (Z^1_{ij}, Z^2_{ij}) are transformed to binary variables with a one-to-one correspondence (i.e. $z^1_{ij} = 1$ is equivalent to $z^1_{ij} = True$, while $z^1_{ij} = 0$ is equivalent to $z^1_{ij} = False$). Exactly one disjunctive term must be selected ($z^1_{ij} + z^1_{ji} + z^2_{ij} + z^2_{ji} = 1$). When a disjunctive term is selected, the corresponding constraints are enforced (e.g. If $z^1_{ij} = 1$, then $x_i + L_i \leq x_j$). When it is not selected, the corresponding constraints become redundant (e.g. If $z^1_{ji} = 0$, then $x_j + L_j \leq x_i + UB$).

The MBM reformulation is as follows:

$$
\begin{aligned}
& & & min\ lt & & \\
& s.t. & & lt \geq x_i + L_i & & i \in N \\
& & & x_i + L_i \leq x_j + UB(z^1_{ji} + z^2_{ij} + z^2_{ji}) & & i,j \in N, i < j \\
& & & x_j + L_j \leq x_i + UB(z^1_{ij} + z^2_{ij} + z^2_{ji}) & & i,j \in N, i < j \\
& & & y_i - H_i \geq y_j - W(z^1_{ij} + z^1_{ji} + z^2_{ji}) & & i,j \in N, i < j \\
& & & y_j - H_j \geq y_i - W(z^1_{ij} + z^1_{ji} + z^2_{ij}) & & i,j \in N, i < j \\
& & & z^1_{ij} + z^1_{ji} + z^2_{ij} + z^2_{ji} = 1 & & i,j \in N, i < j \\
& & & 0 \leq x_i \leq UB - L_i & & i \in N \\
& & & H_i \leq y_i \leq W & & i \in N \\
& & & z^1_{ij}, z^2_{ij} \in \{0,1\} & & i,j \in N, i \neq j
\end{aligned}
\tag{SG-MBM}
$$

Problem (SG-MBM) follows the same rationale as (SG-BM). Note that when a disjunctive term is selected (e.g. $z^1_{ij} = 1$) the other disjunctive terms in that disjunction are zero (e.g. $z^1_{ji} = z^2_{ij} = z^2_{ji} = 0$). Therefore, when a disjunctive term is selected the constraints are enforced (e.g. $x_i + L_i \leq x_j$). When it is not selected, the corresponding constraints become redundant. Note that in (SG-MBM), the constraints have the same M-parameter for any disjunctive term that is selected (UB in the first two and W in the last two). Because of this, the continuous relaxation of (SG-BM) and (SG-MBM) represent exactly the same feasible region. This is not true in general, as will be presented in the next section.

The HR reformulation of this problem is as follows:

$$
\begin{aligned}
& & & min\ lt & & \\
& s.t. & & lt \geq x_i + L_i & & i \in N \\
& & & x_i = \nu_i^{ij} + \nu_i^{ji} & & i,j \in N, i \neq j \\
& & & y_i = \mu_i^{ij} + \mu_i^{ji} & & i,j \in N, i \neq j \\
& & & \nu_i^{ij} + L_i z_{ij}^1 \leq \nu_j^{ij} & & i,j \in N, i < j \\
& & & \nu_j^{ji} + L_j z_{ji}^1 \leq \nu_i^{ji} & & i,j \in N, i < j \\
& & & \mu_i^{ij} - H_i z_{ij}^2 \geq \mu_j^{ij} & & i,j \in N, i < j \\
& & & \mu_j^{ji} - H_j z_{ji}^2 \geq \mu_i^{ji} & & i,j \in N, i < j \\
& & & z_{ij}^1 + z_{ji}^1 + z_{ij}^2 + z_{ji}^2 = 1 & & i,j \in N, i < j \\
& & & 0 \leq \nu_i^{ij} \leq (UB - L_i) z_{ij}^1 & & i,j \in N, i \neq j \\
& & & 0 \leq \nu_i^{ji} \leq (UB - L_i) z_{ji}^1 & & i,j \in N, i \neq j \\
& & & H_i z_{ij}^2 \leq \mu_i^{ij} \leq (W) z_{ij}^2 & & i \in N, i \neq j \\
& & & H_i z_{ji}^2 \leq \mu_i^{ji} \leq (W) z_{ji}^2 & & i \in N, i \neq j \\
& & & 0 \leq x_i \leq UB - L_i & & i \in N \\
& & & H_i \leq y_i \leq W & & i \in N \\
& & & z_{ij}^1, z_{ij}^2 \in \{0,1\} & & i,j \in N, i \neq j
\end{aligned}
\qquad \text{(SG-HR)}
$$

In (SG-HR), the Boolean variables (Z_{ij}^1, Z_{ij}^2) are transformed to binary variables as before, and exactly one disjunctive term must be selected ($z_{ij}^1 + z_{ji}^1 + z_{ij}^2 + z_{ji}^2 = 1$). The variables are disaggregated so that one variable is included for every disjunctive term in which it appears. When a disjunctive term is selected, the disaggregated variable must lie within the variable bounds (e.g. If $z_{ij}^1 = 1$, then $0 \leq \nu_i^{ij} \leq (UB - L_i)$ and $0 \leq \nu_j^{ij} \leq (UB - L_j)$). When a term is not selected, its corresponding disaggregated variable becomes zero (e.g. If $z_{ji}^1 = 0$, then $0 \leq \nu_i^{ji} \leq 0$ and $0 \leq \nu_j^{ji} \leq 0$). The constraints of the selected disjunctions are enforced to the disaggregated variables (e.g. If $z_{ij}^1 = 1$, then $\nu_i^{ij} + L_i \leq \nu_j^{ij}$). The constraints corresponding to not selected terms are trivially satisfied ($0 \leq 0$).

It is easy to see that HR is a larger formulation than BM. However, it provides a formulation that is as strong, or stronger, than the BM.

A.3 Symmetry-breaking GDP formulation

It is clear from Figure A.2 that some regions can be represented with the selection of two different disjunctive terms. Because of that, problem (SG-BM) has several symmetric solutions.

An alternative formulation to break some of the symmetry in the problem is as follows:

$$
\begin{aligned}
& & & min\ lt & & \\
& s.t. & & lt \geq x_i + L_i & & i \in N \\
& & & \begin{bmatrix} Z^1_{ij} \\ x_i + L_i \leq x_j \end{bmatrix} \vee \begin{bmatrix} Z^1_{ji} \\ x_j + L_j \leq x_i \end{bmatrix} & & \\
& & & \vee \begin{bmatrix} Z^2_{ij} \\ y_i - H_i \geq y_j \\ x_i + L_i \geq x_j \\ x_j + L_j \geq x_i \end{bmatrix} \vee \begin{bmatrix} Z^2_{ji} \\ y_j - H_j \geq y_i \\ x_i + L_i \geq x_j \\ x_j + L_j \geq x_i \end{bmatrix} & & i,j \in N, i < j \qquad \text{(S1)} \\
& & & Z^1_{ij} \veebar Z^1_{ji} \veebar Z^2_{ij} \veebar Z^2_{ji} & & i,j \in N, i < j \\
& & & 0 \leq x_i \leq UB - L_i & & i \in N \\
& & & H_i \leq y_i \leq W & & i \in N \\
& & & Z^1_{ij}, Z^2_{ij} \in \{True, False\} & & i,j \in N, i \neq j
\end{aligned}
$$

The modified disjunction in (S1) separates the feasible region that corresponds to $Z^2_{ij} = True$ (and $Z^2_{ji} = True$) from the feasible region that corresponds to $Z^1_{ij} = True$ and to $Z^1_{ji} = True$. The disjunction is illustrated in Figure A.3. In the Figure A.3.a), the pattern to represent the feasible region of $Z^1_{ij} = True$ is different from the pattern used to represent $Z^2_{ij} = True$. The reason for this is to illustrate that for $Z^1_{ij} = True$ the complete rectangle i needs to be positioned in the grey region. In contrast, it is possible to have $Z^2_{ij} = True$ a as long as a part of the rectangle is in the striped region. The feasible region

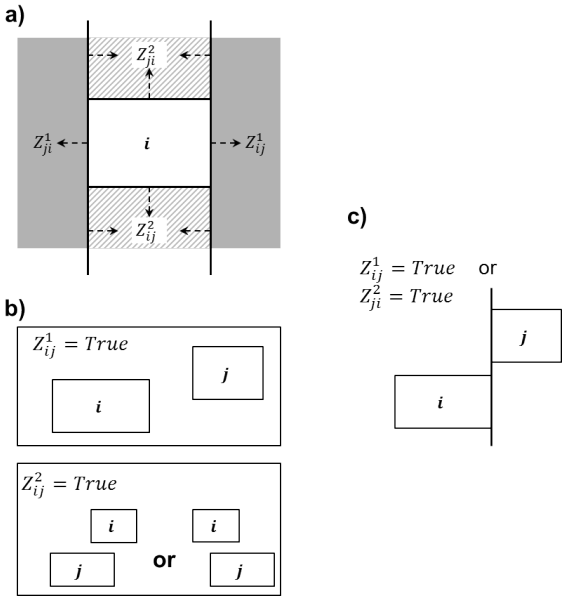

Figure A.3: Illustration of the relative position of j with respect to rectangle i for the symmetry breaking formulation (S1).

that corresponds to $Z^2_{ij} = True$ represents not only i above j, but also i is not to the right nor to the left of j. Figure A.3.b) illustrates a case in which $Z^1_{ij} = True$ (i.e. i is to the right of j) and two cases in which $Z^2_{ij} = True$ (i.e. i is above j, and i it is not to the right nor to the left of j). Note that this strategy could be applied to an alternative formulation in which $Z^2_{ij} = True$ represents only that i is above j, and $Z^1_{ij} = True$ represents i left j and also i is not above or below j. Both of these formulations serve the purpose of breaking symmetric solutions, and in this work we will focus problem (S1).

Problem (S1) breaks some of the symmetry. However, if a rectangle j is on one side of rectangle i, but the edge of rectangle j is aligned to i, there is still symmetry. Figure A.3.c) illustrates such a case. In the Figure, j is above i and not to the right nor to the left or i (it satisfies the constraints corresponding to $Y^2_{ji} = True$). The figure shows that j is also to the right of i (it satisfies the constraints associated to $Y^1_{ji} = True$).

If H_i and L_i are integer, it is possible to break this symmetry with the following formulation (note that if H_i and L_i are rational numbers they can be transformed into integer

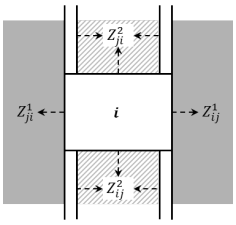

Figure A.4: Illustration of the relative position of j with respect to rectangle i for the symmetry breaking formulation (S2).

values):

$$
\begin{aligned}
& min\ lt \\
s.t.\quad & lt \geq x_i + L_i && i \in N \\
& \begin{bmatrix} Z^1_{ij} \\ x_i + L_i \leq x_j \end{bmatrix} \vee \begin{bmatrix} Z^1_{ji} \\ x_j + L_j \leq x_i \end{bmatrix} \\
& \vee \begin{bmatrix} Z^2_{ij} \\ y_i - H_i \geq y_j \\ x_i + L_i \geq x_j + 1 \\ x_j + L_j \geq x_i + 1 \end{bmatrix} \vee \begin{bmatrix} Z^2_{ji} \\ y_j - H_j \geq y_i \\ x_i + L_i \geq x_j + 1 \\ x_j + L_j \geq x_i + 1 \end{bmatrix} && i,j \in N, i < j \qquad \text{(S2)} \\
& Z^1_{ij} \veebar Z^1_{ji} \veebar Z^2_{ij} \veebar Z^2_{ji} && i,j \in N, i < j \\
& 0 \leq x_i \leq UB - L_i && i \in N \\
& H_i \leq y_i \leq W && i \in N \\
& Z^1_{ij}, Z^2_{ij} \in \{True, False\} && i,j \in N, i \neq j
\end{aligned}
$$

Problem (S2) includes a "+1" in the right hand side of the constraints corresponding to Z^2_{ij}. The feasible region of the disjunction in this formulation is presented in Figure A.4.

The BM and HR reformulations (S1) and (S2) are obtained in the same manner as (SG-BM) and (SG-HR) were obtained. We refer to the BM and HR formulation of (S1) as (S1-BM) and (S1-HR), and the BM and HR of (S2) as (S2-BM) and (S2-HR). However, in this case the MBM of (S1) and (S2) is not the same as the BM. The MBM of (S1) is as

follows:

$$
\begin{aligned}
& & & min\ lt & & \\
& s.t. & & lt \geq x_i + L_i & & i \in N \\
& & & x_i + L_i \leq x_j + UBz^1_{ji} + (L_i + L_j)(z^2_{ij} + z^2_{ji}) & & i,j \in N, i < j \\
& & & x_j + L_j \leq x_i + UBz^1_{ij} + (L_i + L_j)(z^2_{ij} + z^2_{ji}) & & i,j \in N, i < j \\
& & & y_i - H_i \geq y_j - W(z^1_{ij} + z^1_{ji} + z^2_{ji}) & & i,j \in N, i < j \\
& & & x_i + L_i \geq x_j - (UB - L_i - L_j)z^1_{ij} - (L_j - L_i)z^1_{ji} & & i,j \in N, i < j \\
& & & x_j + L_j \geq x_i - (UB - L_i - L_j)z^1_{ji} - (L_i - L_j)z^1_{ij} & & i,j \in N, i < j \\
& & & y_j - H_j \geq y_i - W(z^1_{ij} + z^1_{ji} + z^2_{ij}) & & i,j \in N, i < j \\
& & & z^1_{ij} + z^1_{ji} + z^2_{ij} + z^2_{ji} = 1 & & i,j \in N, i < j \\
& & & 0 \leq x_i \leq UB - L_i & & i \in N \\
& & & H_i \leq y_i \leq W & & i \in N \\
& & & z^1_{ij}, z^2_{ij} \in \{0,1\} & & i,j \in N, i \neq j
\end{aligned}
$$

(S1-MBM)

In (S1-MBM), multiple M-parameters are used for the formulation. When a term is selected, all other binary variables in that disjunction become zero (e.g. If $z^2_{ij} = 1$, then $z^1_{ij} = z^1_{ji} = z^2_{ji} = 0$). Because of this, when a disjunctive term is selected the corresponding constraints are enforced. For example, if $z^2_{ij} = 1$ then $x_i + L_i \geq x_j - (UB - L_i - L_j)0 - (L_j - L_i)0$, so $x_i + L_i \geq x_j$. When a term is not selected its corresponding constraints become redundant, but the M-parameter of the constraint depends on the disjunctive term that is selected. For example, if $z^2_{ij} = 1$ then the constraint that corresponds to Z^1_{ij} becomes redundant as follows: $x_i + L_i \leq x_j + (0)UB + (L_i + L_j)(1)$, so $x_i \leq x_j + L_j$. However, if $z^1_{ji} = 1$ then the constraint that corresponds to Z^1_{ij} becomes redundant as follows: $x_i + L_i \leq x_j + UB$. Also note that the constraints representing "j is not to the right nor to the left of i" are the same in $Z^2_{ij} = True$ and $Z^2_{ji} = True$. Because of this, the constraints in the MBM reformulation only appear once. Finally, note that any solution that is feasible for the continuous relaxation of (S1-MBM) is also feasible for the continuous relaxation of (S1-BM). However, not every solution that is feasible for the continuous relaxation of

(S1-BM) is feasible for the continuous relaxation of (S1-MBM).

The MBM of (S2) can be obtained in a similar manner, and we refer to it as (S2-MBM).

A.4 Numerical results

The different formulations were tested 100 random instances of the strip packing problem. The range of values of the random parameters is as follows: $N = 5\text{-}14; W = 10\text{-}20; L_i = 1\text{-}5; H_i = 2\text{-}5$. All of the instances were solved using GAMS 24.3.3[21], using an Intel(R) Core(TM) i7 CPU 2.93 GHz and 4 GB of RAM. Gurobi 5.6.3[115] was used for the solution of the MILPs.

Figure A.5 compares the performance curves of the three GDP formulations ((SG), (S1) and (S2)). Figure A.5.a) compares these three formulations using the BM reformulation, Figure A.5.b) using MBM, and Figure A.5.c) using HR. The plots present the performance of the formulations for the 100 instances, showing the percentage of problems that are solved vs. time. It is clear from the three plots that (S2) is the best formulation, independently of which reformulation is used. It can also be observed that (S1) is better than (SG) in all cases.

Figure A.6 presents the performance of the different GDP-to-MILP reformulations (BM, MBM, and HR) for the three GDP problems. It is clear from the three figures that the BM reformulation is solved faster than the MBM and the HR, and that the MBM is solved faster than the HR. The HR yields a larger formulation than the BM, so its performance can be worse than the BM in some problems (such as this one). However, it is unexpected that the BM performs better than the MBM. Both of the reformulations provide an MILP with the same number of variables and constraints. For (SG), the continuous relaxation both formulations represent the exact same feasible region. However, for (S1) and (S2) the constraints of the continuous relaxation of MBM dominate the constraints of the continuous relaxation of BM (i.e. any solution that satisfies the constraints in the continuous relaxation MBM also satisfies the constraints in the continuous relaxation of BM). The opposite is not true (it is easy to find solutions that are feasible for the continuous relaxation

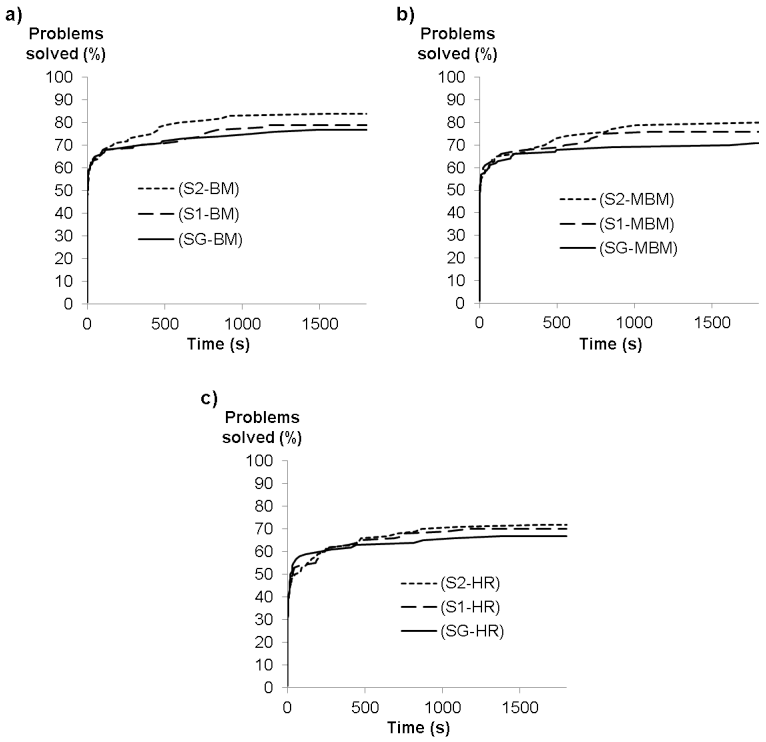

Figure A.5: Performance curve for solving (SG), (S1), and (S2) using reformulations: a) BM, b) MBM, and c) HR.

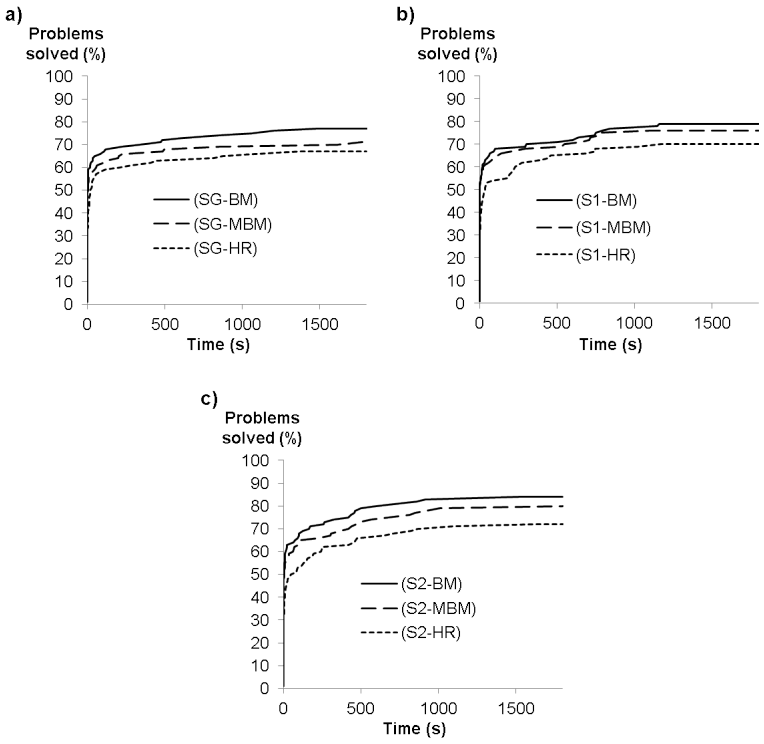

Figure A.6: Performance curve for the different GDP-to-MILP reformulations, for: a) (SG), b) (S1), and c) (S2).

BM and not for the continuous relaxation of the MBM). Because of this, it is expected that the MBM performs better than the BM. One possible explanation of why the BM is performing better than the MBM is how the solver handles both formulations. In particular the pre-solve, heuristics, and branching strategies may be better equipped to handle traditional Big-M type of formulations. Another possibility is that the BM provides integer solutions more often than the MBM in the nodes of the branch and bound.

A.5 Conclusions

This paper has presented an alternative generalized disjunctive (S1) programming formulation, which has the property of partially breaking the symmetry in the straightforward

formulation (SG). Furthermore, a sharper reformulation (S2) has been presented for the case of integer lengths of the rectangles. Numerical results have shown that for each of these three models, the big-M reformulation as mixed-integer linear program performs faster than the hull reformulation and than a modified big-M reformulation. Furthermore, for each of these reformulations (S2) outperforms (S1), which in turn outperforms (SG).

Appendix B

Convex GDP examples

B.1 Process network

The process network problem is a classic optimization problem in process design. The model seeks to maximize the profit of selling a set of products taking into account the cost of raw materials and equipment. Figure B.1 illustrates the superstructure for a process with potentially 8 units. The model that describes the performance of each unit is normally large and quite complex. In this example, however, the process is simplified to single input-output relations that give rise to a convex GDP[15]. The GDP problem formulation is

as follows:

$$
\begin{aligned}
& min\ Z = \sum_{i \in I} c_i + \sum_{j \in J} p_j x_j + \alpha \\
s.t. \quad & \sum_{j \in J} r_{jn} x_j \leq 0 && \forall n \in N \\
& \begin{bmatrix} Y_i \\ \sum_{j \in J^i} d_{ij}(e^{x_j/t_{ij}} - 1) - \sum_{j \in J^i} s_{ij} x_j \leq 0 \\ c_i = \gamma_i \end{bmatrix} \vee \begin{bmatrix} \neg Y_i \\ x_j = 0 \quad \forall j \in J^i \\ c_i = 0 \end{bmatrix} && i \in I \\
& \Omega(Y) = True \\
& c_i, x_j \geq 0 \\
& Y_i \in \{True, False\}
\end{aligned}
\tag{B.1}
$$

In (B.1) c_i is the cost associated to each equipment $i \in I$. x_j represents each of the flows $j \in J$, and p_j the profit or cost associated to each one. The global constraints represent the mass balance in each of the $n \in N$ nodes, where r_{jn} is the coefficient of the mass balance for flow j. There is a disjunction for each unit i. If a unit is selected ($Y_i = True$) then the corresponding mass balance has to be satisfied, and the cost of the unit c_i takes the value associated to that equipment γ_i. If it is not selected ($Y_i = False$ or, equivalently, $\neg Y_i = True$), then all the flows $j \in J^i$ in and out that equipment become 0, and the cost c_i also becomes 0. Finally $\Omega(Y) = True$ represents the topology of the superstructure.

B.2 Farm layout (Flay)

In the farm layout problem the objective is to determine the width and length of a number of rectangles with fixed area in order to minimize the total perimeter. Figure B.2 illustrates

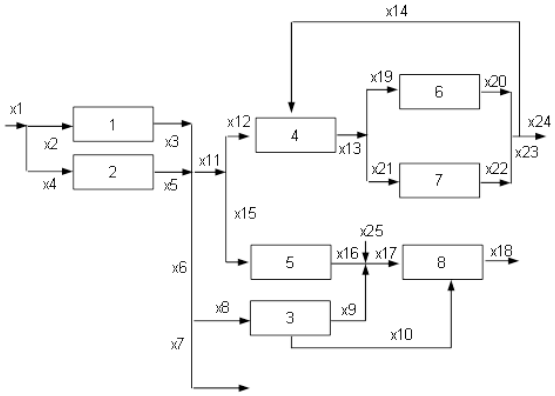

Figure B.1: Superstructure illustration of an 8-equipment process network

this problem, which can be formulated as the following convex GDP[13]:

$$
\begin{aligned}
& min\ Z = 2(Length + Width) && \\
s.t.\quad & Length \geq x_i + L_i && i \in N \\
& Width \geq y_i + W_i && i \in N \\
& A_i/W_i - L_i \leq 0 && i \in N \\
& \begin{bmatrix} Y_{ij}^1 \\ x_i + L_i \leq x_j \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^2 \\ x_j + L_j \leq x_i \end{bmatrix} && \\
& \vee \begin{bmatrix} Y_{ij}^3 \\ y_i + W_i \leq y_j \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^4 \\ y_j + W_j \leq y_i \end{bmatrix} && i,j \in N, i < j \\
& Y_{ij}^1 \veebar Y_{ij}^2 \veebar Y_{ij}^3 \veebar Y_{ij}^4 && i,j \in N, i < j \\
& 0 \leq Length \leq Length^{up};\ \ 0 \leq Width \leq Width^{up} && \\
& L_i^{lo} \leq L_i \leq L_i^{up};\ \ W_i^{lo} \leq W_i \leq L_i^{up} && i \in N \\
& 0 \leq x_i \leq Length^{up} - L_i^{lo};\ \ 0 \leq y_i \leq Width^{up} - L_i^{lo} && i \in N \\
& Y_{ij}^1, Y_{ij}^2, Y_{ij}^3, Y_{ij}^4 \in \{True, False\} && i,j \in N, i < j
\end{aligned}
\tag{B.2}
$$

In formulation (B.2) the variables x_i and y_i represent the coordinates of lower-left corner of each rectangle $i \in N$, while L_i and W_i represent their corresponding length and width.

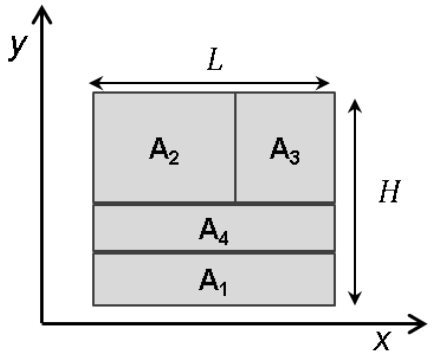

Figure B.2: Illustration of farm layout problem

$Length$ and $Width$ represent the length and width of the total area. A_i is the given area for each rectangle. Similarly to the strip packing problem (SG), there is one disjunction for each pair of rectangles. Each term in the disjunction represents the possible relative position between the two rectangles: rectangle i is either to the left, or to the right, or below, or above rectangle j, respectively.

B.3 Constrained layout (Clay)

The constrained layout problem is similar to the strip packing problem, but the rectangles in this case have to be packed inside a set of fixed circles. The objective function is to minimize the distance in x and y axis, with a cost associated to every pair of rectangles. Figure B.3 illustrates the constrained layout problem. It can be formulated as the following

convex GDP[13]:

$$
\begin{aligned}
& min\ Z = \sum_i \sum_j c_{ij}(delx_{ij} + dely_{ij}) \\
s.t.\quad & delx_{ij} \geq x_i - x_j && i,j \in N, i<j \\
& delx_{ij} \geq x_j - x_i && i,j \in N, i<j \\
& dely_{ij} \geq y_i - y_j && i,j \in N, i<j \\
& dely_{ij} \geq y_j - y_i && i,j \in N, i<j \\
& \begin{bmatrix} Y_{ij}^1 \\ x_i + L_i/2 \leq x_j - L_j/2 \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^2 \\ x_j + L_j/2 \leq x_i - L_i/2 \end{bmatrix} \\
& \vee \begin{bmatrix} Y_{ij}^3 \\ y_i + H_i/2 \leq y_j - H_j/2 \end{bmatrix} \vee \begin{bmatrix} Y_{ij}^4 \\ y_j + H_j/2 \leq y_i - H_i/2 \end{bmatrix} && i,j \in N, i<j \\
& \underset{t\in T}{\vee} \begin{bmatrix} W_{it} \\ (x_i + L_i/2 - xc_t)^2 + (y_i + H_i/2 - yc_t)^2 \leq r_t^2 \\ (x_i + L_i/2 - xc_t)^2 + (y_i - H_i/2 - yc_t)^2 \leq r_t^2 \\ (x_i - L_i/2 - xc_t)^2 + (y_i + H_i/2 - yc_t)^2 \leq r_t^2 \\ (x_i - L_i/2 - xc_t)^2 + (y_i - H_i/2 - yc_t)^2 \leq r_t^2 \end{bmatrix} && i \in N \\
& Y_{ij}^1 \veebar Y_{ij}^2 \veebar Y_{ij}^3 \veebar Y_{ij}^4 && i,j \in N, i<j \\
& \underset{t\in T}{\veebar} W_{it} && i \in N \\
& 0 \leq x_i \leq x_i^{up} && i \in N \\
& 0 \leq y_i \leq y_i^{up} && i \in N \\
& Y_{ij}^1, Y_{ij}^2, Y_{ij}^3, Y_{ij}^4 \in \{True, False\} && i,j \in N, i<j \\
& W_{it} \in \{True, False\} && i \in N, t \in T
\end{aligned}
\tag{B.3}
$$

In formulation (B.3) x_i and y_i represent the coordinates of the centre of the rectangles $i \in N$. $delx_{ij}$ and $dely_{ij}$ represent the distance between two rectangles $i,j \in N, i<j$, and c_{ij} is the cost associated with these. The first disjunctions, similarly to strip packing and farm layout problems, ensures that there is no overlap by expressing the possible

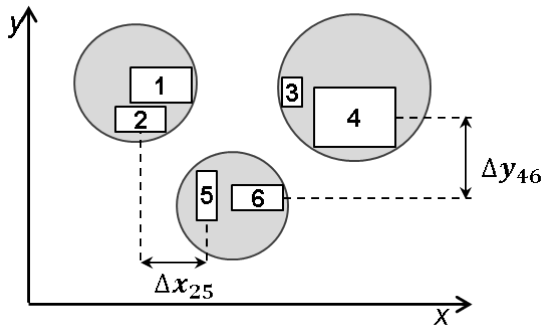

Figure B.3: Illustration of constrained layout problem

relative position between rectangles i and j. The second set of disjunctions ensure that every rectangle i is inside one of the $t \in T$ circles. For a circle t, its coordinates (xc_t,yc_t) and its radius r_t are given.

B.4 Design of multi-product batch plant (Batch)

This problem seeks to minimize the investment cost in the design of a plant with multiple units in parallel and intermediate storage tanks[116]. The design involves selecting the number of parallel units, volume of the equipment, and volume and location of the intermediate

storage tanks. This problem can be convexified[107], and the formulation is as follows:

$$
\begin{aligned}
& min\ Z = \alpha_1 \sum_j \exp(n_j + m_j + \beta_1 v_j) + \alpha_2 \sum_{Tj} \exp(\beta_2 v_{Tj}) \\
s.t.\quad & v_j \geq \ln(S_{ij}) + b_{ij} - n_j && \forall i, j \\
& e_i \geq \ln(T_{ij}) - b_{ij} - m_j && \forall i, j \\
& H \geq \sum_i (Q_i e_i) \\
& \begin{bmatrix} YS_j \\ v_{Tj} \geq \ln(S_j^*) + b_{ij+1} \quad \forall i \\ v_{Tj} \geq \ln(S_j^*) + b_{ij} \quad \forall i \\ b_{ij} - b_{ij+1} \leq \ln(S_{ij}^*) \quad \forall i \\ b_{ij} - b_{ij+1} \geq -\ln(S_{ij}^*) \quad \forall i \end{bmatrix} \vee \begin{bmatrix} \neg YS_j \\ v_{Tj} = 0 \\ b_{ij} - b_{ij+1} = 0 \quad \forall i \end{bmatrix} && \forall j < |J| \\
& \begin{bmatrix} YM_{j,1} \\ m_j = \ln(1) \end{bmatrix} \vee \ldots \vee \begin{bmatrix} YM_{j,maxp} \\ m_j = \ln(maxp) \end{bmatrix} && \forall j \\
& \begin{bmatrix} YN_{j,1} \\ n_j = \ln(1) \end{bmatrix} \vee \ldots \vee \begin{bmatrix} YN_{j,maxp} \\ n_j = \ln(maxp) \end{bmatrix} && \forall j \\
& YM_{j,1} \veebar \ldots \veebar YM_{j,maxp} && \forall j \\
& YN_{j,1} \veebar \ldots \veebar YN_{j,maxp} && \forall j \\
& YS_j, YM_{j,p}, YN_{j,p} \in \{True, False\} && \forall j, p = 1, ..., maxp
\end{aligned}
\tag{B.4}
$$

Nomenclature for design of a multi-product batch plant example.

Given:

$\alpha_1, \alpha_2, \beta_1, \beta_2$: Coefficients for the capital cost of the units and intermediate storage tanks.

$i \in I$: products.

$j \in J$: stages.

H: horizon time.

Q_i: production rate of product i.

T_{ij}: processing time of product i at stage j.

S_{ij}: size factor of product i at stage j.

S_j^*: size factor for intermediate storage tank.

S_{ij}^*: size factor for stages.

Determine:

B_{ij}: batch size product i at stage j.

E_i: production cycle time / batch size i.

M_j: number of units in parallel out-of-phase at stage j.

N_j: number of units in parallel in phase at stage j.

V_j: Unit size of stage j.

V_{Tj}: size of intermediate storage tank between stage j and $j+1$.

In order to convexify the problem, the following variables are introduced:

$b_{ij} = \ln(B_{ij})$

$e_i = \ln(E_i)$

$m_j = \ln(M_j)$

$n_j = \ln(N_j)$

$v_j = \ln(V_j)$

$v_{Tj} = \ln(V_{Tj})$

Appendix C

Nonconvex GDP examples

C.1 Layout optimization of screening systems in recovered paper production.

Nomenclature:

SETS:

$J = \{fib, st\}$ Components (fibre is the "good component" and stickies is the "bad component").

$N = S \cup \{ta, tr\}$: Total nodes in the system (possible screens, total accept, and total reject).

S: Possible screens.

PARAMETERS (all parameters are greek letters or capital letters):

α_s: Exponent coefficient for cost in screen s

$\beta_{s,j}$: Acceptance factor beta for screen s and component j.

C_s^1: Cost coefficient 1 for screen s.

C_s^2: Cost coefficient 2 for screen s.

C_{st}^{up}: Maximum percentage of inlet stickies accepted in the total accepted flow.

F_j^0: Source flow of component j.

$[F_s^{in,lo}, F_s^{in,up}]$: Lower and upper bound of flow into screen s.

W^1, W^2, W^3: Weighting factors in objective function for lost fire, accepted stickies, and capital cost respectively.

CONTINUOUS (POSITIVE) VARIABLES:

c_s: Cost of screen s.

f_s: Total inlet flow into screen s.

$f_{n,j}^I$: Inlet flow of component j into node n.

$f_{s,j}^A$: Accepted flow of component j from screen s.

$f_{s,j}^R$: Rejected flow of component j form screen s.

$m_{s,n,j}^A$: Accepted flow of component j from screen s to node n.

$m_{s,n,j}^R$: Rejected flow of component j from screen s to node n.

$m_{n,j}^0$: Flow of component j from source to node n.

r_s: Reject rate of screen s.

BOOLEAN VARIABLES:

Y_s: Selection of screen s.

$YA_{s,n}$: Existence of accepted flow from screen s to node n.

$YR_{s,n}$: Existence of rejected flow from screen s to node n.

$Y0_n$: Existence of flow from source to node n.

GDP model:

$$min \quad W^1 f^I_{tr,fib} + W^2 f^I_{ta,st} + W^3 \sum_{s \in S} c_s$$

$$\begin{aligned}
s.t. \quad & f^I_{ta,st} \leq C^{up}_{st} F^0_{st} \\
& f^I_{s,j} = f^A_{s,j} + f^R_{s,j} && s \in S, j \in J \\
& f_s = \sum_{j \in J} f^I_{s,j} && s \in S \\
& f^I_{n,j} = m^0_{n,j} + \sum_{\substack{s \in S \\ s \neq n}} \left(m^A_{s,n,j} + m^R_{s,n,j} \right) && n \in N, j \in J \\
& F^0_j = \sum_{n \in N} m^0_{n,j} && j \in J \\
& \begin{bmatrix} Y_s \\ F^{in,lo}_s \leq f^I_s \leq F^{in,up}_s \\ f^R_{s,j} = f^I_{s,j} (r_s)^{\beta_{s,j}} \;\; j \in J \\ c_s = C^1_s (f^I_s)^{\alpha_s} + C^2_s (1 - r_s) \end{bmatrix} \vee \begin{bmatrix} \neg Y_s \\ f^I_s = 0 \\ c_s = 0 \end{bmatrix} && s \in S \\
& \begin{bmatrix} YA_{s,n} \\ m^A_{s,n,j} = f^A_{s,j} \;\; j \in J \end{bmatrix} \vee \begin{bmatrix} \neg YA_{s,n} \\ m^A_{s,n,j} = 0 \;\; j \in J \end{bmatrix} && s \in S, n \in N, n \neq s \\
& \begin{bmatrix} YR_{s,n} \\ m^R_{s,n,j} = f^R_{s,j} \;\; j \in J \end{bmatrix} \vee \begin{bmatrix} \neg YR_{s,n} \\ m^R_{s,n,j} = 0 \;\; j \in J \end{bmatrix} && s \in S, n \in N, n \neq s \\
& \bigvee_{n \in N} \begin{bmatrix} Y0_n \\ m^0_{n,j} = F^0_j \;\; j \in J \\ m^0_{n',j} = 0 \;\; n' \neq n, j \in J \end{bmatrix} \\
& \veebar_{n \in N} Y0_n \\
& YA_{s,n} \vee YR_{s,n} \Rightarrow Y_s && s \in S, n \in N, n \neq s \\
& YA_{s',s} \vee YR_{s',s} \Rightarrow Y_s && (s, s') \in S, s' \neq s \\
& YA_{s',s} \veebar YA_{s,s'} && (s, s') \in S, s' \neq s \\
& YR_{s',s} \veebar YR_{s,s'} && (s, s') \in S, s' \neq s \\
& YA_{s,n} \veebar YR_{s,n} && s \in S, n \in N, n \neq s \\
& YA_{s,n} \veebar YR_{s,n} && s \in S, n \in N, n \neq s
\end{aligned} \tag{C.1}$$

C.2 Reactor-separator process superstructure.

Nomenclature:

SETS:

I: Components (a, b, c).

$K = R \cup S$: Total processing units.

R: Reactors.

S: Separation units.

P: Sources of raw materials.

PARAMETERS:

Raw materials.

$C^0_{p,i}$: Molar concentration of component i in raw material p.

PR^0_p: Cost of raw material p.

$F^{0,up}_p$: Maximum availability of raw material p.

Demand.

M^D_c: Minimum mol fraction of component c in demand stream.

F^D: Minimum demand (total flow).

Separation:

$\xi_{s,i} \in \{0, 1\}$: $\xi_{s,i} = 1$ of component i exits outlet stream $out1$ in separation process s. $\xi_{s,i} = 0$ of component i exits outlet stream $out2$.

Reactors:

γ_r: Minimum concentration ratio between component a and b.

k_r: Reaction rate constant for reactor r.

r_r: Design residence time for reactor r.

All units:

α_k: Cost coefficient for unit k.

β_k: Cost exponent for unit k.

$PP^0_{k,k'}$: Cost of installing a pipeline between unit k and unit k'.

Variable bounds:

$[F^{lo}_k, F^{up}_k]$: Lower and upper bound of total flow into unit k.

$[C^{lo}_{k,i}, C^{up}_{k,i}]$: Lower and upper bound of molar concentration of i into unit k.

$[\mu^{lo}_{k,k'}, \mu^{up}_{k,k'}]$: Lower and upper bound of total flow from unit k into unit k'.

$[\nu^{lo}_{k,k',i}, \nu^{up}_{k,k',i}]$: Lower and upper bound of molar concentration of component i in stream from unit k into unit k'.

CONTINUOUS (POSITIVE) VARIABLES:

F^{in}_k: Inlet stream for unit k.

F^{out}_r: Outlet stream of reactor r.

F^{out1}_s, F^{out2}_s: Outlet streams 1 and 2 of separation unit s.

$C^{in}_{k,i}$: Molar concentration of i in the inlet stream for unit k.

$C^{out}_{r,i}$: Molar concentration of i in the outlet stream of reactor r.

$C^{out1}_{s,i}, C^{out2}_{s,i}$: Molar concentration of i in the outlet streams $(out1, out2)$ of unit s.

$\mu_{r,k}$: Total flow from reactor r to unit k.

$\nu_{r,k,i}$: Molar concentration of component i in stream from reactor r into unit k.

$\mu^{out1}_{s,k}, \mu^{out2}_{s,k}$: Total flow from outlet streams $(out1, out2)$ of unit s to unit k.

$\nu^{out1}_{s,k,i}, \nu^{out2}_{s,k,i}$: Molar concentration of component i in stream from outlet streams $(out1, out2)$ of unit s to unit k.

$\mu_{p,k}^{raw}$: Flow of raw material from source p ro unit k.

μ_r^D: Flow of outlet stream from reactor r to demand.

$\mu_s^{out1,D}, \mu_s^{out2,D}$: Flow of outlet streams $(out1, out2)$ from separation unit s to demand.

$\nu_{r,i}^D$: Molar concentration of component i in flow of outlet stream from reactor r to demand.

$\nu_{s,i}^{out1,D}, \mu_{s,i}^{out2,D}$: Molar concentration of component i in flow of outlet streams $(out1, out2)$ from separation unit s to demand.

PU_k: Cost of unit k.

$PP_{k,k'}$: Cost of pipeline from unit k to unit k'.

PU^T, PP^T, PR^T: Total cost of units, pipelines and raw materials.

BOOLEAN VARIABLES:

Y_k: selection of unit k.

$YF_{r,k}$: Existence of flow between reactor r and unit k.

$YF_{s,k}^{out1}, YF_{s,k}^{out2}$: Existence of flow between outlet streams from separation unit s and unit k.

For clarity in the GDP model, we partition the model in several sections. The first section includes constraints related to total costs and demand satisfaction:

$$
\begin{aligned}
& min \quad PU^T + PP^T + PR^T \\
s.t. \quad & PU^T = \sum_{k \in K} PU_k \\
& PP^T = \sum_{k \in K} \sum_{\substack{k' \in K \\ k \neq k'}} PP_{k,k'} \\
& PR^T = \sum_{p \in P} PR_p^0 \sum_{k \in K} \mu_{p,k}^{raw} \\
& \sum_{r \in R} \mu_r^D + \sum_{s \in S} (\mu_s^{out1,D} + \mu_s^{out2,D}) \geq F^D \\
& M_c^D \left(\sum_{r \in R} \mu_r^D \sum_{i \in I} \nu_{r,i}^D + \sum_{s \in S} (\mu_s^{out1,D} \sum_{i \in I} \nu_{s,i}^{out1,D} + \mu_s^{out2,D} \sum_{i \in I} \nu_{s,i}^{out2,D}) \right) \\
& \leq \sum_{r \in R} \mu_r^D \nu_{r,c}^D + \sum_{s \in S} (\mu_s^{out1,D} \nu_{s,c}^{out1,D} + \mu_s^{out2,D} \nu_{s,c}^{out2,D}) \\
& \sum_{k \in K} \mu_{p,k}^{raw} \leq F_p^{0,up} \qquad p \in P
\end{aligned}
\tag{C.2}
$$

The next constraints represent the mixing and splitting before and after unit k:

$$
\begin{aligned}
&F_k^{in} = \sum_{\substack{r\in R\\ r\neq k}} \mu_{r,k} + \sum_{\substack{s\in S\\ s\neq k}} (\mu_{s,k}^{out1} + \mu_{s,k}^{out2}) + \sum_{p\in P} \mu_{p,k}^{raw} && k \in K\\
&F_k^{in} C_{k,i}^{in} = \sum_{\substack{r\in R\\ r\neq k}} \mu_{r,k}\nu_{r,k,i}\\
&+ \sum_{\substack{s\in S\\ s\neq k}} (\mu_{s,k}^{out1}\nu_{s,k,i}^{out1} + \mu_{s,k}^{out2}\nu_{s,k,i}^{out2}) + \sum_{p\in P} \mu_{p,k}^{raw} C_{p,i}^{0} && k \in K, i \in I\\
&F_r^{out} = \sum_{\substack{k\in K\\ k\neq r}} \mu_{r,k} + \mu_r^D && r \in R\\
&F_s^{out1} = \sum_{\substack{k\in K\\ k\neq s}} \mu_{s,k}^{out1} + \mu_s^{out1,D} && s \in S\\
&F_s^{out2} = \sum_{\substack{k\in K\\ k\neq s}} \mu_{s,k}^{out2} + \mu_s^{out2,D} && s \in S\\
&C_{r,i}^{out} = \nu_{r,k,i} && r \in R, k \in K, k \neq r, i \in I\\
&C_{r,i}^{out} = \nu_{r,i}^{D} && r \in R, i \in I\\
&C_{s,i}^{out1} = \nu_{s,k,i}^{out1} && s \in S, k \in K, k \neq s, i \in I\\
&C_{s,i}^{out2} = \nu_{s,k,i}^{out2} && s \in S, k \in K, k \neq s, i \in I\\
&C_{s,i}^{out1} = \nu_{s,i}^{out1,D} && s \in S, i \in I\\
&C_{s,i}^{out2} = \nu_{s,i}^{out2,D} && s \in S, i \in I
\end{aligned}
\tag{C.3}
$$

The following constraints represent the selection or not of processing units:

$$
\begin{aligned}
& F_s^{in} = F_s^{out1} + F_s^{out2} && s \in S \\
& F_r^{in} = F_r^{out} && r \in R \\
& \left[\begin{array}{c} Y_s \\ F_s^{in} C_{s,i}^{in} = \xi_{s,i} F_s^{out1} C_{s,i}^{out1} + (1-\xi_{s,i}) F_s^{out1} C_{s,i}^{out1} \;\; i \in I \\ F_s^{out1} \sum_{i \in I} C_{s,i}^{in} = F_s^{in} \sum_{i \in I} \xi_{s,i} C_{s,i}^{in} \\ F_s^{out2} \sum_{i \in I} C_{s,i}^{in} = F_s^{in} \sum_{i \in I} (1-\xi_{s,i}) C_{s,i}^{in} \\ C_{s,i}^{out1} = 0 \;\; i \in I, \xi_{s,i} = 0 \\ C_{s,i}^{out2} = 0 \;\; i \in I, \xi_{s,i} = 1 \\ F_s^{lo} \le F_s^{in} \le F_s^{up} \\ C_{s,i}^{lo} \le C_{s,i}^{in} \le C_{s,i}^{up} \;\; i \in I \\ PU_s = \alpha_s (F_s^{in})^{\beta_s} \end{array}\right] \vee \left[\begin{array}{c} \neg Y_s \\ F_s^{in} = 0 \end{array}\right] && s \in S \\
& \left[\begin{array}{c} Y_r \\ C_{r,b}^{in} \ge \gamma_r C_{r,a}^{in} \\ (C_{r,b}^{out} C_{r,a}^{in}) / (C_{r,b}^{in} C_{r,a}^{out}) = exp((C_{r,b}^{in} - C_{r,a}^{in}) k_r t_r) \\ C_{r,b}^{out} = C_{r,b}^{in} - C_{r,a}^{in} + C_{r,a}^{out} \\ C_{r,c}^{out} = C_{r,c}^{in} + C_{r,a}^{in} - C_{r,a}^{out} \\ F_r^{lo} \le F_r^{in} \le F_r^{up} \\ C_{r,i}^{lo} \le C_{s,i}^{in} \le C_{r,i}^{up} \;\; i \in I \\ PU_r = \alpha_r (F_r^{in})^{\beta_r} \end{array}\right] \vee \left[\begin{array}{c} \neg Y_r \\ F_r^{in} = 0 \end{array}\right] && r \in R
\end{aligned}
\tag{C.4}
$$

The last set of equations represents the existence or not of flow between units.

$$\begin{bmatrix} YF^{out1}_{s,k} \\ PP_{s,k} = PP^{0}_{s,k} \\ \mu^{lo}_{s,k} \le \mu^{out1}_{s,k} \le \mu^{up}_{s,k} \\ \nu^{lo}_{s,k,i} \le \nu^{out1}_{s,k,i} \le \nu^{up}_{s,k,i} \;\; i \in I \end{bmatrix} \vee \begin{bmatrix} \neg Y^{out1}_{s,k} \\ \mu_{s,k} = 0 \end{bmatrix} \quad s \in S$$

$$\begin{bmatrix} YF^{out2}_{s,k} \\ PP_{s,k} = PP^{0}_{s,k} \\ \mu^{lo}_{s,k} \le \mu^{out2}_{s,k} \le \mu^{up}_{s,k} \\ \nu^{lo}_{s,k,i} \le \nu^{out2}_{s,k,i} \le \nu^{up}_{s,k,i} \;\; i \in I \end{bmatrix} \vee \begin{bmatrix} \neg Y^{out2}_{s,k} \\ \mu_{s,k} = 0 \end{bmatrix} \quad s \in S$$

$$\begin{bmatrix} YF_{r,k} \\ PP_{r,k} = PP^{0}_{r,k} \\ \mu^{lo}_{r,k} \le \mu_{r,k} \le \mu^{up}_{r,k} \\ \nu^{lo}_{r,k,i} \le \nu_{r,k,i} \le \nu^{up}_{r,k,i} \;\; i \in I \end{bmatrix} \vee \begin{bmatrix} \neg Y_{r,k} \\ \mu_{r,k} = 0 \end{bmatrix} \quad r \in R \tag{C.5}$$

$$YF_{r,k} \Rightarrow Y_r r \in R, k \in K, k \neq r$$
$$YF^{out1}_{s,k} \Rightarrow Y_s s \in S, k \in K, k \neq s$$
$$YF^{out2}_{s,k} \Rightarrow Y_s s \in S, k \in K, k \neq s$$
$$YF_{r,k} \Rightarrow Y_k r \in R, k \in K, k \neq r$$
$$YF^{out1}_{s,k} \Rightarrow Y_k s \in S, k \in K, k \neq s$$
$$YF^{out2}_{s,k} \Rightarrow Y_k s \in S, k \in K, k \neq s$$

Appendix D

Haverly pooling problem

A well-known benchmark problem is a small problem proposed by Haverly[76]. The problem is defined as follows: There is a single pool that receives supplies from two different sources A and B, with different specification content. A third supply C is not fed into the pool but is directly mixed with the two outflows from the pool. The specification parameters for the streams going into the pool are 3% for A, 1% for B, and 2% for C. The blending of flows from the pool and from the supply stream C produces products X and Y, which have to adhere to the specifications of maximum 2.5% and 1.5%, respectively. The maximum demands for products X and Y are 100 and 200, respectively. The objective is to minimize the cost of the blending operation while meeting the demand requirements. As opposed to the multiperiod blending problem, supply and demand flows do not depend on time and they are the decision variables. Figure D.1 illustrates the topology and parameters of the network.

The problem was modeled using *p*, *q*, *pq* and ($\mathbb{SB}$) formulations. Total flows, denoted by the variable $F_{nn'}$, are present in all models. The rest of the variables are different. Analyzing each formulation: (i) the *p*-formulation is based on the concentration value of the specification, C_q, in the pool and its outputs, (ii) the *q* and *pq*-formulations are based on the fraction of incoming flow to the pool that is contributed by each supply, q_s, and (iii) the source-based model is based on individual flows per source, $\tilde{F}_{snn'}$, and split fractions,

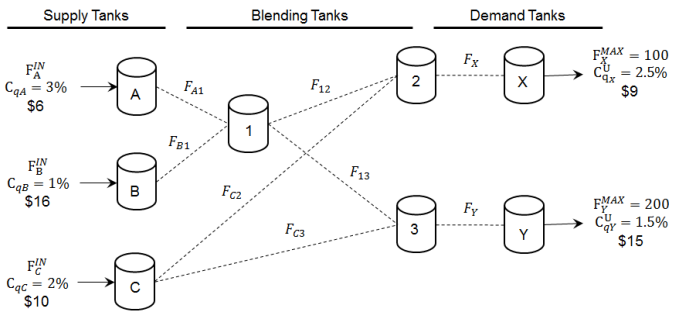

Figure D.1: Sketch of Haverly Problem

Table D.1: Optimal solution of the relaxed LP for the Haverly problem with different formulations.

	# Variables				**# Constrains**				**Normalized relaxation**			
Linear relaxation	**p**	**q**	**pq**	$\mathbb{SB}$	**p**	**q**	**pq**	$\mathbb{SB}$	**p**	**q**	**pq**	$\mathbb{SB}$
McCormick	14	15	15	27	19	26	30	49	1.25	6.125	1.25	1.25
w/o McCormick	12	11	11	**21**	8	6	6	**19**	5.25	9.75	9.75	**1.25**

ξ_{1n}, that represent the fraction of the total outgoing flow from the pool that is being sent to each of the mixers.

Another important difference is the number of variables, constraints and bilinear terms. The size of the problem and the number of bilinear terms increase from left to right in the table. Therefore, the only reason to choose the source-based model over, for example, the concentration model, is if the former had a tighter LP relaxation than the latter. Table D.1 shows the size and the optimal solution of the relaxed LP for the four formulations. Two solutions are displayed. The first row corresponds to the optimal solution when the bilinear terms are replaced by their McCormick envelopes. The second row has the solutions when the nonlinear constraints are dropped entirely. Note that, for the McCormick envelopes, the bounds on the variables are, $C_q = \{1, 3\}$, $q_s = \{0, 1\}$, $\xi_{1n} = \{0, 1\}$, $F_{nn'} = \tilde{F}_{snn'} = \{0, 300\}$.

Two conclusion can be drawn. First, the optimum of the relaxed LP is the same for the *p*, *pq* and ($\mathbb{SB}$) formulations when the bilinear terms are replaced by their McCormick envelopes. However, the number of variables and constraints is larger for the latter. Sec-

ondly, when the non-convex equations are eliminated from the formulations, the source-based model is tighter than the rest of the formulations. In the traditional pooling formulations, the nonconvexities appear in the specification requirements constraints, whereas in the source-based model they are only present in the "split fractions equations". This difference explains the significant improvement in the tightness of the relaxation when the non-linear terms are dropped.

The following table presents the p-formulation, q-formulation, pq-formulation and ($\mathbb{SB}$) formulation for the Haverly pooling problem.

Constraints	p-formulation	q-formulation/pq-formulation	(𝕊𝔹)
Mass balance	$F_A^{IN} = F_{A1}$ $F_B^{IN} = F_{B1}$ $F_C^{IN} = F_{C2} + F_{C3}$ $F_{A1} + F_{B1} = F_{12} + F_{13}$ $F_X = F_{12} + F_{C2}$ $F_Y = F_{13} + F_{C3}$	$F_A^{IN} = q_A(F_{12} + F_{13})$ $F_B^{IN} = q_B(F_{12} + F_{13})$ $F_C^{IN} = F_{C2} + F_{C3}$ $F_X = F_{12} + F_{C2}$ $F_Y = F_{13} + F_{C3}$ $q_A + q_B = 1$	$F_A^{IN} = F_{A1}$ $F_B^{IN} = F_{B1}$ $F_C^{IN} = F_{C2} + F_{C3}$ $F_{A1} + F_{B1} = F_{12} + F_{13}$ $F_X = F_{12} + F_{C2}$ $F_Y = F_{13} + F_{C3}$
Spec. mass balance	$3F_{A1} + 1F_{B1}$ $= C_q(F_{12} + F_{13})$		
Flows Source mass bal. Split fractions			$F_{A1} = \tilde{F}_{A,A1}$ $F_{B1} = \tilde{F}_{B,B1}$ $F_{C2} = \tilde{F}_{C,C2}$ $F_{C3} = \tilde{F}_{C,C3}$ $F_{12} = \tilde{F}_{A,12} + \tilde{F}_{B,12}$ $F_{13} = \tilde{F}_{A,13} + \tilde{F}_{B,13}$ $\tilde{F}_{A,A1} = \tilde{F}_{A,12} + \hat{F}_{A,13}$ $\tilde{F}_{B,B1} = \tilde{F}_{B,12} + \tilde{F}_{B,13}$ $F_{12} = \xi_{12}(F_{A1} + F_{B1})$ $F_{13} = \xi_{13}(F_{A1} + F_{B1})$ $\tilde{F}_{A,12} = \xi_{12}\tilde{F}_{A,A1}$ $\tilde{F}_{B,12} = \xi_{12}\tilde{F}_{B,B1}$ $\tilde{F}_{A,13} = \xi_{13}\tilde{F}_{A,A1}$ $\tilde{F}_{B,13} = \xi_{13}\tilde{F}_{B,B1}$ $\xi_{12} + \xi_{13} = 1$
Redundant Const.		$q_A F_{12} + q_B F_{12} = F_{12}$ $q_A F_{13} + q_B F_{13} = F_{13}$ $q_A F_{12} + q_A F_{13} = 300 * q_{A1}$ $q_B F_{12} + q_B F_{13} = 300 * q_{B1}$	
Spec. requirements Demands	$C_q F_{12} + 2F_{C2} \leq 2.5F_X$ $C_q F_{13} + 2F_{C3} \leq 1.5F_Y$ $F_X \leq 100$ $F_Y \leq 200$	$3q_A F_{12} + q_B F_{12} + 2F_{C2} \leq 2.5F_X$ $3q_A F_{13} + q_B F_{13} + 2F_{C3} \leq 1.5F_Y$ $F_X \leq 100$ $F_Y \leq 200$	$3\tilde{F}_{A,12} + \tilde{F}_{B,12} + 2\tilde{F}_{C,C2} \leq 2.5F_X$ $3\tilde{F}_{A,13} + \tilde{F}_{B,13} + 2\tilde{F}_{C,C3} \leq 1.5F_Y$ $F_X \leq 100$ $F_Y \leq 200$

Bibliography

[1] Raman R, Grossmann I. E. Modelling and computational techniques for logic based integer programming. *Computers & Chemical Engineering.* 1994;18:563–578.

[2] Floudas C. A, Gounaris C. E. A review of recent advances in global optimization. *Journal of Global Optimization.* 2009;45:3–38.

[3] Biegler L. T, Grossmann I. E. Retrospective on optimization. *Computers & Chemical Engineering.* 2004;28:1169–1192.

[4] Bixby R, Rothberg E. Progress in computational mixed integer programminga look back from the other side of the tipping point. *Annals of Operations Research.* 2007;149:37–41.

[5] Belotti P, Kirches C, Leyffer S, Linderoth J, Luedtke J, Mahajan A. Mixed-integer nonlinear optimization. *Acta Numerica.* 2013;22:1–131.

[6] Wolsey L. A, Nemhauser G. L. *Integer and combinatorial optimization*. John Wiley & Sons 2014.

[7] Lee S, Grossmann I. E. New algorithms for nonlinear generalized disjunctive programming. *Computers & Chemical Engineering.* 2000;24:2125–2141.

[8] Grossmann I. E, Lee S. Generalized convex disjunctive programming: Nonlinear convex hull relaxation. *Computational Optimization and Applications.* 2003;26:83–100.

[9] Williams H. P. *Model building in mathematical programming*. John Wiley & Sons 2013.

[10] Grossmann I. E, Trespalacios F. Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. *AIChE Journal.* 2013;59:3276–3295.

[11] Ceria S, Soares J. Convex programming for disjunctive convex optimization. *Mathematical Programming.* 1999;86:595–614.

[12] Balas E. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods.* 1985;6:466–486.

[13] Sawaya N. *Reformulations, relaxations and cutting planes for generalized disjunctive programming*. PhD thesis 2006.

[14] Stubbs R. A, Mehrotra S. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical programming.* 1999;86:515–532.

[15] Ruiz J. P, Grossmann I. E. A hierarchy of relaxations for nonlinear convex generalized disjunctive programming. *European Journal of Operational Research.* 2012;218:38–47.

[16] Trespalacios F, Grossmann I. E. Review of Mixed-Integer Nonlinear and Generalized Disjunctive Programming Methods. *Chemie Ingenieur Technik.* 2014;86:991–1012.

[17] Beaumont N. An algorithm for disjunctive programs. *European Journal of Operational Research.* 1990;48:362–371.

[18] Türkay M, Grossmann I. E. Logic-based MINLP algorithms for the optimal synthesis of process networks. *Computers & Chemical Engineering.* 1996;20:959–978.

[19] Vielma J. P. Mixed integer linear programming formulation techniques. *To appear in SIAM Review.* July 2014.

[20] CMU , IBM . CMU-IBM Open Source MINLP Project

[21] Brooke A, Kendrick D, Meeraus A, Raman . GAMS, a Users Guide. *The Scientific Press.* 1998.

[22] Mahajan A. Presolving mixed–integer linear programs. *Wiley Encyclopedia of Operations Research and Management Science.* 2010.

[23] Savelsbergh M. W. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing.* 1994;6:445–454.

[24] Achterberg T. SCIP: solving constraint integer programs. *Mathematical Programming Computation.* 2009;1:1–41.

[25] Lodi A. Mixed integer programming computation. in *50 Years of Integer Programming 1958-2008*:619–645Springer 2010.

[26] Sawaya N, Grossmann I. A hierarchy of relaxations for linear generalized disjunctive programming. *European Journal of Operational Research.* 2012;216:70–82.

[27] Hooker J. N. *Integrated Methods for Optimization*. Springer US 2012.

[28] Vecchietti A, Lee S, Grossmann I. E. Modeling of discrete/continuous optimization problems: characterization and formulation of disjunctions and their relaxations. *Computers & Chemical Engineering.* 2003;27:433–448.

[29] Sawaya N. W, Grossmann I. E. A cutting plane method for solving linear generalized disjunctive programming problems. *Computers & Chemical Engineering.* 2005;29:1891–1913.

[30] Trespalacios F, Grossmann I. E. Algorithmic approach for improved mixed-integer reformulations of convex Generalized Disjunctive Programs. *INFORMS Journal on Computing.* 2014;27:59–74.

[31] Guignard M. Lagrangean relaxation. *Top.* 2003;11:151–200.

[32] Fisher M. L. The Lagrangian relaxation method for solving integer programming problems. *Management science.* 2004;50:1861–1871.

[33] Rong A, Lahdelma R, Luh P. B. Lagrangian relaxation based algorithm for trigeneration planning with storages. *European Journal of Operational Research.* 2008;188:240–257.

[34] Terrazas-Moreno S, Trotter P. A, Grossmann I. E. Temporal and spatial Lagrangean decompositions in multi-site, multi-period production planning problems with sequence-dependent changeovers. *Computers & Chemical Engineering.* 2011;35:2913–2928.

[35] Cornuejols G, Fisher M. L, Nemhauser G. L. Exceptional paper-location of bank accounts to optimize float: An analytic study of exact and approximate algorithms. *Management science.* 1977;23:789–810.

[36] Carøe C. C, Schultz R. Dual decomposition in stochastic integer programming. *Operations Research Letters.* 1999;24:37–45.

[37] Geoffrion A. M. *Lagrangean relaxation for integer programming*. Springer 1974.

[38] Balas E. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics.* 1998;89:3–44.

[39] Hooker J. N. Logic-based methods for optimization in *Principles and Practice of Constraint Programming*:336–349Springer 1994.

[40] CPLEX I. I. V12. 1: Users Manual for CPLEX. *International Business Machines Corporation.* 2009;46:157.

[41] GAMS . General Algebraic Modeling System (GAMS) Release 24.4.5 Washington, DC, USA 2015.

[42] Trespalacios F, Grossmann I. E. Symmetry breaking for generalized disjunctive programming formulation of the strip packing problem. *Annals of Operations Research.* ;Submitted for publication.

[43] Grossmann I. E. Mixed-integer programming approach for the synthesis of integrated process flowsheets. *Computers & Chemical Engineering.* 1985;9:463–482.

[44] Grossmann I. E. Mixed-integer nonlinear programming techniques for the synthesis of engineering systems. *Research in Engineering Design.* 1990;1:205–228.

[45] Friedler F, Tarjan K, Huang Y, Fan L. Graph-theoretic approach to process synthesis: axioms and theorems. *Chemical Engineering Science.* 1992;47:1973–1988.

[46] Smith E, Pantelides C. Design of reaction/separation networks using detailed models. *Computers & Chemical Engineering.* 1995;19:83–88.

[47] Papalexandri K. P, Pistikopoulos E. N. Generalized modular representation framework for process synthesis. *AIChE Journal.* 1996;42:1010–1032.

[48] Yeomans H, Grossmann I. E. A systematic modeling framework of superstructure optimization in process synthesis. *Computers & Chemical Engineering.* 1999;23:709–731.

[49] Horst R, Tuy H. *Global optimization: Deterministic approaches*. Springer Science & Business Media 1996.

[50] Adjiman C. S, Androulakis I. P, Floudas C. A. A global optimization method, αBB, for general twice-differentiable constrained NLPsII. Implementation and computational results. *Computers & Chemical Engineering.* 1998;22:1159–1179.

[51] Misener R, Floudas C. A. ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization.* 2014;59:503–526.

[52] Sahinidis N. V. BARON: A general purpose global optimization software package. *Journal of global optimization.* 1996;8:201–205.

[53] Tawarmalani M, Sahinidis N. V. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming.* 2005;103:225–249.

[54] Belotti P, Lee J, Liberti L, Margot F, Wächter A. Branching and bounds tighteningtechniques for non-convex MINLP. *Optimization Methods & Software.* 2009;24:597–634.

[55] Lin Y, Schrage L. The global solver in the LINDO API. *Optimization Methods & Software.* 2009;24:657–668.

[56] Bussieck M. R, Vigerske S. MINLP solver software. *Wiley Encyclopedia of Operations Research and Management Science.* 2010.

[57] Kesavan P, Allgor R. J, Gatzke E. P, Barton P. I. Outer approximation algorithms for separable nonconvex mixed-integer nonlinear programs. *Mathematical Programming.* 2004;100:517–535.

[58] Duran M. A, Grossmann I. E. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical programming.* 1986;36:307–339.

[59] Fletcher R, Leyffer S. Solving mixed integer nonlinear programs by outer approximation. *Mathematical programming.* 1994;66:327–349.

[60] Hooker J. N, Ottosson G. Logic-based Benders decomposition. *Mathematical Programming.* 2003;96:33–60.

[61] Benders J. F. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik.* 1962;4:238–252.

[62] Geoffrion A. M. Generalized benders decomposition. *Journal of optimization theory and applications.* 1972;10:237–260.

[63] Li X, Tomasgard A, Barton P. I. Nonconvex generalized Benders decomposition for stochastic separable mixed-integer nonlinear programs. *Journal of optimization theory and applications.* 2011;151:425–454.

[64] Raman R, Grossmann I. E. Symbolic integration of logic in mixed-integer linear programming techniques for process synthesis. *Computers & Chemical Engineering.* 1993;17:909–927.

[65] Bergamini M. L, Aguirre P, Grossmann I. Logic-based outer approximation for globally optimal synthesis of process networks. *Computers & Chemical Engineering.* 2005;29:1914–1933.

[66] Smith E. M, Pantelides C. C. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering.* 1999;23:457–478.

[67] McCormick G. P. Computability of global solutions to factorable nonconvex programs: Part IConvex underestimating problems. *Mathematical programming.* 1976;10:147–175.

[68] Tawarmalani M, Sahinidis N. V. *Convexification and global optimization in continuous and mixed-integer nonlinear programming: theory, algorithms, software, and applications*;65. Springer Science & Business Media 2002.

[69] Adjiman C. S, Dallwig S, Floudas C. A, Neumaier A. A global optimization method, αBB, for general twice-differentiable constrained NLPsI. Theoretical advances. *Computers & Chemical Engineering.* 1998;22:1137–1158.

[70] Jeroslow R. G. *Logic-based decision support: Mixed integer model formulation*. Elsevier 1989.

[71] Laporte G, Louveaux F. V. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations research letters.* 1993;13:133–142.

[72] Khajavirad A, Sahinidis N. V. Convex envelopes generated from finitely many compact convex sets. *Mathematical Programming.* 2013;137:371–408.

[73] Fügenschuh A, Hayn C, Michaels D. Mixed-integer linear methods for layout-optimization of screening systems in recovered paper production. *Optimization and Engineering.* 2014;15:533–573.

[74] Yeomans H, Grossmann I. E. Disjunctive programming models for the optimal design of distillation columns and separation sequences. *Industrial & Engineering Chemistry Research.* 2000;39:1637–1648.

[75] Kelly J, Mann J. Crude oil blend scheduling optimization: an application with multimillion dollar benefits-Part 1: Process/plant optimization. *Hydrocarbon Processing.* 2003;82:47–53.

[76] Haverly C. A. Studies of the behavior of recursion for the pooling problem. *ACM SIGMAP Bulletin.* 1978:19–28.

[77] Ben-Tal A, Eiger G, Gershovitz V. Global minimization by reducing the duality gap. *Mathematical programming.* 1994;63:193–212.

[78] Audet C, Brimberg J, Hansen P, Digabel S. L, Mladenović N. Pooling problem: Alternate formulations and solution methods. *Management science.* 2004;50:761–776.

[79] Kolodziej S. P, Grossmann I. E, Furman K. C, Sawaya N. W. A discretization-based approach for the optimization of the multiperiod blend scheduling problem. *Computers & Chemical Engineering.* 2013;53:122–142.

[80] Foulds L, Haugland D, Jörnsten K. A bilinear approach to the pooling problem. *Optimization.* 1992;24:165–180.

[81] Androulakis I, Maranas C, Floudas C. αBB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization.* 1995;7:337–363.

[82] Adhya N, Tawarmalani M, Sahinidis N. V. A Lagrangian approach to the pooling problem. *Industrial & Engineering Chemistry Research.* 1999;38:1956–1972.

[83] Quesada I, Grossmann I. E. Global optimization of bilinear process networks with multicomponent flows. *Computers & Chemical Engineering.* 1995;19:1219–1242.

[84] Sherali H. D, Adams W. P. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*;31. Springer Science & Business Media 1998.

[85] Karuppiah R, Furman K. C, Grossmann I. E. Global optimization for scheduling refinery crude oil operations. *Computers & Chemical Engineering.* 2008;32:2745–2766.

[86] Wicaksono D. S, Karimi I. Piecewise MILP under-and overestimators for global optimization of bilinear programs. *AIChE Journal.* 2008;54:991–1008.

[87] Gounaris C. E, Misener R, Floudas C. A. Computational comparison of piecewise-linear relaxations for pooling problems. *Industrial & Engineering Chemistry Research.* 2009;48:5742–5766.

[88] Misener R, Thompson J. P, Floudas C. A. APOGEE: Global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. *Computers & Chemical Engineering.* 2011;35:876–892.

[89] Vielma J. P, Nemhauser G. L. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming.* 2011;128:49–72.

[90] Kolodziej S, Castro P. M, Grossmann I. E. Global optimization of bilinear programs with a multiparametric disaggregation technique. *Journal of Global Optimization.* 2013;57:1039–1063.

[91] Teles J. P, Castro P. M, Matos H. A. Multi-parametric disaggregation technique for global optimization of polynomial programming problems. *Journal of Global Optimization.* 2013;55:227–251.

[92] Kesavan P, Barton P. I. Generalized branch-and-cut framework for mixed-integer nonlinear optimization problems. *Computers & Chemical Engineering.* 2000;24:1361–1366.

[93] Li J, Misener R, Floudas C. A. Continuous-time modeling and global optimization approach for scheduling of crude oil operations. *AIChE Journal.* 2012;58:205–226.

[94] Lee H, Pinto J. M, Grossmann I. E, Park S. Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management. *Industrial & Engineering Chemistry Research.* 1996;35:1630–1641.

[95] Shah N. Mathematical programming techniques for crude oil scheduling. *Computers & Chemical Engineering.* 1996;20:S1227–S1232.

[96] Shah N. K, Ierapetritou M. G. Short-term scheduling of a large-scale oil-refinery operations: Incorporating logistics details. *AIChE Journal.* 2011;57:1570–1584.

[97] Méndez C. A, Grossmann I. E, Harjunkoski I, Kaboré P. A simultaneous optimization approach for off-line blending and scheduling of oil-refinery operations. *Computers & Chemical Engineering.* 2006;30:614–634.

[98] Jia Z, Ierapetritou M. Mixed-integer linear programming model for gasoline blending and distribution scheduling. *Industrial & Engineering Chemistry Research.* 2003;42:825–835.

[99] Mouret S, Grossmann I. E, Pestiaux P. A novel priority-slot based continuous-time formulation for crude-oil scheduling problems. *Industrial & Engineering Chemistry Research.* 2009;48:8515–8528.

[100] Castro P. M, Grossmann I. E. Generalized disjunctive programming as a systematic modeling framework to derive scheduling formulations. *Industrial & Engineering Chemistry Research.* 2012;51:5781–5792.

[101] Moro L. F, Pinto J. M. Mixed-integer programming approach for short-term crude oil scheduling. *Industrial & engineering chemistry research.* 2004;43:85–94.

[102] Reddy P. C. P, Karimi I, Srinivasan R. A new continuous-time formulation for scheduling crude oil operations. *Chemical Engineering Science.* 2004;59:1325–1341.

[103] Achterberg T. *SCIP-a framework to integrate constraint and mixed integer programming*. Konrad-Zuse-Zentrum für Informationstechnik Berlin 2004.

[104] Alfaki M, Haugland D. Strong formulations for the pooling problem. *Journal of Global Optimization.* 2013;56:897–916.

[105] Gupte A, Ahmed S, Dey S. S, Cheon M. S. Pooling problems: relaxations and discretizations. *School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA. and ExxonMobil Research and Engineering Company, Annandale, NJ.* 2013.

[106] Hart W. E, Watson J.-P, Woodruff D. L. Pyomo: modeling and solving mathematical programs in Python *Mathematical Programming Computation.* 2011;3:219–260.

[107] Vecchietti A, Grossmann I. E. LOGMIP: a disjunctive 0–1 non-linear optimizer for process system models. *Computers & Chemical Engineering.* 1999;23:555–565.

[108] Lodi A, Martello S, Monaci M. Two-dimensional packing problems: A survey. *European Journal of Operational Research.* 2002;141:241–252.

[109] Wäscher G, Haußner H, Schumann H. An improved typology of cutting and packing problems. *European Journal of Operational Research.* 2007;183:1109–1130.

[110] Lodi A, Martello S, Vigo D. Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems. *INFORMS Journal on Computing.* 1999;11:345–357.

[111] Martello S, Monaci M, Vigo D. An exact approach to the strip-packing problem. *INFORMS Journal on Computing.* 2003;15:310–319.

[112] Alvarez-Valdes R, Parreno F, Tamarit J. A branch and bound algorithm for the strip packing problem. *OR spectrum.* 2009;31:431–459.

[113] Westerlund J, Papageorgiou L. G, Westerlund T. A MILP model for N-dimensional allocation. *Computers & Chemical Engineering.* 2007;31:1702–1714.

[114] Castro P. M, Oliveira J. F. Scheduling inspired models for two-dimensional packing problems. *European Journal of Operational Research.* 2011;215:45–56.

[115] Optimization G, others . Gurobi optimizer reference manual. *URL: http://www. gurobi. com.* 2012.

[116] Ravemark E. *Optimization models for design and operation of chemical batch processes*. PhD thesisETH Zurich 1995.