# Algorithms for Stochastic Mixed-integer Nonlinear Programming and Long Term Optimization of Electric Power Systems

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Department of Chemical Engineering

Can Li (李灿)
B.S., Chemical Engineering, Tsinghua University

Carnegie Mellon University
Pittsburgh, PA
May, 2021

# Acknowledgments

First of all, I would like to thank my advisor, Prof. Ignacio E. Grossmann, the best advisor that I could dream of. His commitment to academic excellence, hard work, and passion for new knowledge have always inspired and motivated me. I am very grateful for his encouragement and guidance over the last five years. The research discussions with him have been intellectually stimulating which motivated me to think out-of-the-box and pursue high-risk high-reward ideas. In addition to his extraordinary professional accomplishments, he is an exceptional human being with high ethical standards, humility, and a great sense of humor. He really cares about his students and has shaped both my professional and personal life. I am grateful and honored to have him as my mentor.

I would also like to thank Prof. Yushan Zhu (朱玉山) who was my mentor at Tsinghua University. He led me into the field of process systems engineering and encouraged me to work with the best people in the world for my PhD*. He is a true scholar who always aims high and commits to academic excellence. He believes "cherry only has to blossom once for only a short time but do so vibrantly and beautifully" from Japanese culture, which has an influence on me.

I am honored to have Prof. Suvrajeet Sen, Prof. Gerald Cornuejols, Prof. Nick Sahinidis, and Prof. Chrysanthos Gounaris serve on my thesis committee who have provided insightful advice on my work. They are all world leaders in their fields and are my role models.

I would like to acknowledge the financial support by U.S. Department of Energy, Office of Fossil Energy through the Institute for the Design of Advanced Energy Systems (IDAES); EQT Corporation; ExxonMobil Corporation, and the Center for Advanced Process Decision-making (CAPD).

During my PhD, I was fortunate to collaborate with a number of outstanding researchers from different institutions. I would like to thank Prof. Antonio

*"做科研就是要去关公面前耍大刀"

Conejo for his guidance on the power systems projects. Dr. Cristiana Lara developed the expansion planning model and provided me with great advice. I would also like to thank Dr. Dimitri Papageorgiou, Dr. Peng Liu, Dr. John Siirola, Dr. Benjamin Omell, and Dr. David Miller for their constructive feedback and suggestions on the power systems projects. During my internship at ExxonMobil, Dr. Kevin Furman proposed a collaboration opportunity that made the sample average approximation project possible. The preliminary work of the SAAOA paper was conducted by Dr. Jing Wei, Dr. Marco Duran, and Dr. Kevin Furman at ExxonMobil. The collaboration with Dr. Kevin Furman and David Bernal is an enjoyable experience for me. Prof. Nick Sahinidis provided us with a special version of BARON that provides convex relaxations for nonconvex problems. I am grateful to Dr. Markus Drouven and Dr. John Eason for their support on the shale gas project with EQT. Prof. Paul Barton and Dr. Rohit Kannan provided the code of their GOSSIP library that helped to test our algorithms. I also had the opportunity to collaborate with a number of my fellow graduate students, David Bernal, Cristiana Lara, Hector Perez, Dr. Christian Hubbs, Zedong Peng, on their research projects.

I want to thank all my friends in Pittsburgh and other parts of the world for their friendship. The good memories of graduate school: the beer time during happy hour; the graduate student gangster for getting free food; the brainstorms in front of the whiteboard will always be near and dear to my heart.

Finally, I would like to thank my family, especially my parents, Feng Li (李峰) and Haixia Xue (薛海霞), for their continuous and unconditional love and support.

# Abstract

This thesis addresses two challenging problems. The first part is focused on developing algorithms and software for two-stage stochastic mixed-integer nonlinear programming problems (SMINLPs). The second part is on the long-term planning of power systems.

Stochastic programming, also known as stochastic optimization, is a mathematical framework to model decision-making under uncertainty that has been widely applied in process systems engineering (PSE). Two-stage stochastic mixed-integer programming (SMIPs) is a special case of stochastic programming that considers first and second stage decisions made sequentially with both discrete and continuous variables. Although there have been algorithmic advances in linear SMIPs, the decomposition algorithms to address the nonlinear counterpart, stochastic mixed-integer nonlinear programs (SMINLPs), are few. In Part I of this thesis, we propose four decomposition algorithms for different classes of SMINLPs. For SMINLPs with convex nonlinear constraints, mixed-binary first and second variables, and discrete probability distributions, we propose an improved L-shaped algorithm that combines strengthened Benders cuts and Lagrangean cuts. This algorithm has no guarantee of global optimality. To close the optimality gap, we propose a generalized Benders decomposition-based branch-and-bound algorithm where the stage two problems are convexified sequentially by performing basic steps. For SMINLPs with nonconvex constraints, mixed-binary first and second variables, and discrete probability distributions, we propose a generalized Benders decomposition-based branch-and-cut algorithm where we combine the Benders cuts derived from convexification of the stage two problems and the Lagrangean cuts. A spatial branch-and-cut algorithm is performed to guarantee convergence to global optimality. Last but not least, a sample-average approximation-based outer approximation algorithm is proposed for nonconvex SMINLPs with continuous probability distributions. This algo-

rithm does internal sampling within an outer approximation framework using confidence interval estimates.

Part II addresses generation expansion planning of power systems under high penetration of renewables. We propose a mixed-integer linear programming (MILP) model that incorporates both the investment decisions on the generating units, storage units, and transmission lines, and short-term unit commitment decisions to capture the variations of the renewables. To make the large-scale MILP model tractable, we propose several spatial and temporal aggregation schemes and adapt the Benders decomposition algorithm and the nested Benders decomposition algorithm to solve the problem efficiently. We also investigate several different algorithms to select the representative days. Case studies of the ERCOT region are provided to demonstrate the capabilities of our approaches.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

This thesis has two major parts. The first part describes algorithms and software for stochastic mixed-integer nonlinear programming (SMINLP) problems. The focus of the first part is to propose novel decomposition algorithms that can solve two-stage stochastic mixed-integer nonlinear programs. The second part deals with capacity expansion planning of powers systems. The focus of the second part is proposing modeling frameworks, reasonable simplifications, and solution techniques for long term planning of power systems.

The detailed list of objectives for the two classes of problems is as follows:

I Algorithms and software for stochastic mixed-integer nonlinear programming (SMINLP):

- Propose an improved L-shaped algorithm for two-stage SMINLP with convex nonlinear constraints and mixed-binary first and second stage variables.
- Propose a generalized Benders decomposition-based algorithm for two-stage SMINLP with convex nonlinear constraints and mixed-binary first and second stage variables with finite $\epsilon$ convergence.
- Propose a generalized Benders decomposition-based branch and cut algorithm for two-stage SMINLP with nonconvex nonlinear constraints and mixed-binary first and second stage variables.

- Propose an outer approximation-based sample average approximation algorithm for two-stage SMINLP with continuous probability distribution and nonconvex nonlinear constraints.
- Formulate a two-stage stochastic programming model for shale gas pad development under the uncertainty of natural gas price.

II Long term planning of power systems:

- Propose a mixed-integer linear programming formulation and algorithms for generation and transmission expansion planning of power systems under high penetration of renewables.
- Propose computational methods for selecting representative days for capacity expansion planning problems and analyze the theoretical properties of these methods.

This chapter contains a review of the basic concepts that are dealt with in the rest of the thesis. Section 1.1 provides an introduction to stochastic programming. The detailed introduction to stochastic programming and its applications in process systems can be found in chapter 2. Section 1.2 provides an introduction to the energy systems applications addressed in this thesis, including shale gas development in section 1.2.1 and capacity expansion planning of power systems in section 1.2.2.

## 1.1 Introduction to stochastic programming

### 1.1.1 Stochastic programming and applications in process systems engineering

Stochastic programming, also known as stochastic optimization (Birge and Louveaux, 2011), is a mathematical framework to model decision-making under uncertainty. The origin of stochastic programming dates back to the 1950s when George B. Dantzig, commonly

recognized as the father of linear programming, wrote the pioneer paper "Linear Programming under Uncertainty" (Dantzig, 1955). In this pioneering paper, Dantzig described one of the motivations of developing the stochastic programming modeling framework as "to include the case of uncertain demands for the problem of optimal allocation of a carrier fleet to airline routes to meet an anticipated demand distribution". Another early work on stochastic programming can be found in Beale (1955). From then on, stochastic programming has evolved into a major area of research for the mathematical programming and operations research community. A significant number of theoretical and algorithmic developments have been made by the mathematicians, which are summarized in the classical textbooks (Birge and Louveaux, 2011; Shapiro et al., 2014). With the increase in the maturity of algorithmic and computational methods, stochastic programming has been applied to a broad spectrum of problems (Wallace and Ziemba, 2005) including financial planning, electricity generation, supply chain management, mitigation of climate change, and pollution control, among many others.

Process systems engineering (PSE) is an area of chemical engineering that focuses on the development and application of modeling and computational methods to simulate, design, control, and optimize processes (Sargent, 2005). Uncertainties are prevalent in the optimization of process systems, such as prices and purity of raw materials, customer demands, yields of pilot reactors, etc. The marriage of stochastic programming with PSE seems to be a natural alliance. However, the first application of stochastic programming in PSE took place a decade after Dantzig wrote his first paper. The earliest paper that we can find is the paper by Kittrell and Watson (1966) where the authors applied stochastic programming to the optimal design of proces equipment under uncertain parameters. The reason that prohibited researchers in PSE to apply stochastic programming is that computational resources were limited in the early days as stochastic programming models are much more difficult to solve than their deterministic counterparts. After the 1990s, with the improvement of commercial mathematical programming software, e.g, solvers like CPLEX (Lima, 2010), and computer hardware, there is an increasing interest to apply stochastic programming to process systems

applications. In Figure 1.1, we survey the number of papers with stochastic programming as the main topic published in four mainstream PSE journals and conferences, Computers & Chemical Engineering, Computer Aided Chemical Engineering, Industrial & Engineering Chemistry Research, and AIChE Journal, from 1990 to September 1st, 2020. There is a significant growth in the number of papers in this surveyed time horizon, with around 30 papers per year after the 2010s.

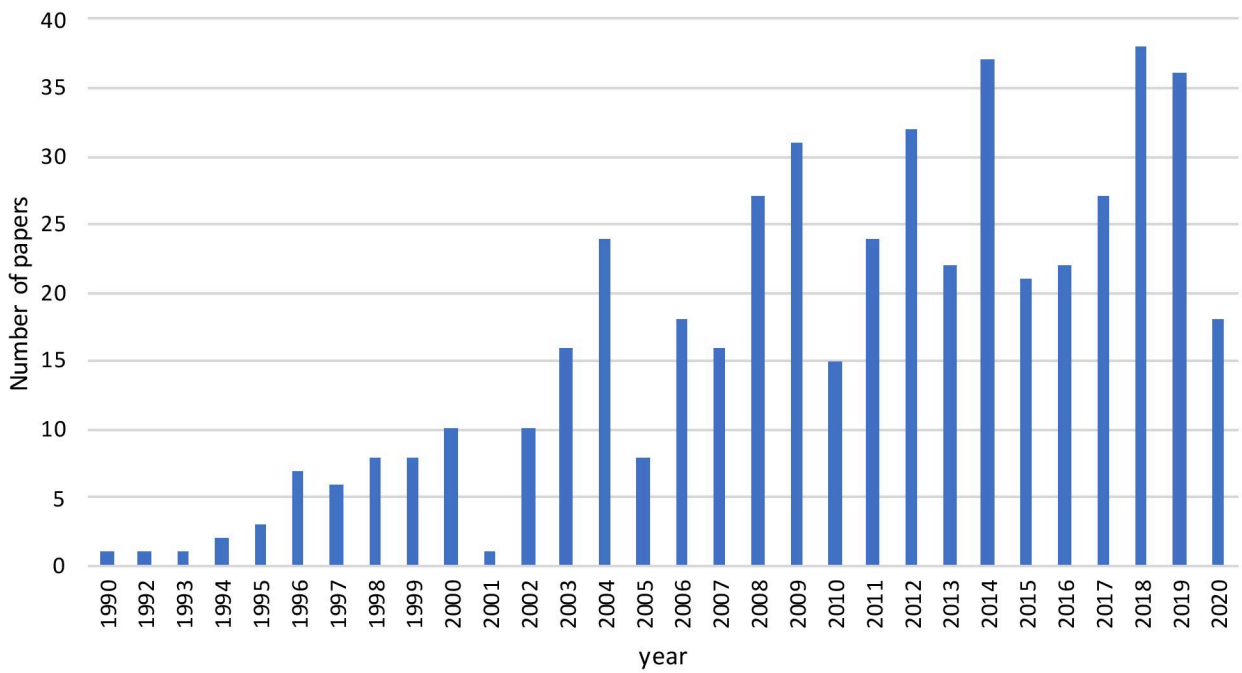


Figure 1.1: The number of papers published in journals including Computers & Chemical Engineering, Computer Aided Chemical Engineering, Industrial & Engineering Chemistry Research, and AIChE Journal from 1990 to September 1st 2020. (Data obtained from Web of Science)

The applications of stochastic programming are also widespread in the PSE community. In Table 1.1, some highly-cited papers from the four PSE-related journals that apply stochastic programming are reported. The applications have a very broad temporal scale, ranging from long-term design and planning problems to short-term scheduling and control problems. In terms of industrial sectors, the listed papers in Table 1.1 have both traditional industrial sectors, such as petroleum, natural gas, pharmaceutical, chemical, etc., and new

Table 1.1: Summary of representative works that use stochastic programming for PSE applications

| work | application | sources of uncertainty |
|---|---|---|
| Gupta and Maranas (2003) | supply chain planning | demand |
| Kim et al. (2011) | biomass supply chain network | supply, demands, prices, processing technologies |
| Guillén-Gosálbez and Grossmann (2009) | chemical supply chain | life cycle inventory |
| Gebreslassie et al. (2012) | hydrocarbon biorefinery supply chains | supply, demand |
| Liu and Sahinidis (1996) | process planning | supply, demand |
| Pistikopoulos and Ierapetritou (1995) | process design | supply, demand |
| Acevedo and Pistikopoulos (1998) | process synthesis | supply, demand |
| Goel and Grossmann (2004) | offshore gas field developments | gas reserves |
| Zeballos et al. (2014) | design and planning of supply chain | supply, demand |
| Levis and Papageorgiou (2004) | capacity planning in pharmaceutical industry | clinical trial outcomes |
| Karuppiah and Grossmann (2008) | synthesis of water networks | contaminants concentration, removal efficiency |
| Colvin and Maravelias (2008) | clinical trial planning | clinical trial outcomes |
| Li et al. (2011b) | natural gas production network design and operation | quality of natural gas |
| Sand and Engell (2004) | real-time scheduling of batch plant | processing time, yields, capacity, demand |
| Liu et al. (2010) | polygeneration energy systems design | price, demand |
| Paules IV and Floudas (1992) | synthesis of heat-integrated distillation | feed composition and flowrate |
| Chu and You (2013) | integrated scheduling and dynamic optimization | process uncertainty, e.g., kinetic parameters |
| Ye et al. (2014) | production scheduling of steelmaking | demand |
| Zhang et al. (2016) | scheduling of power-intensive processes | electricity price |
| Legg et al. (2012) | gas detector placement | leak locations, weather conditions |
| Han and Lee (2012) | carbon capture and storage infrastructure design | CO2 emissions, product prices, operating costs |
| Zavala (2014) | control of natural gas networks | demand |
| Zeng and Cremaschi (2018) | shale gas infrastructure planning | production |

sectors, such as biofuels, carbon capture, etc. The uncertainties that are considered in those applications include prices, supply, production profiles, and concentration of raw materials, demands and prices of final products, process technologies, clinical trial outcomes.

### 1.1.2 Optimization under Uncertainty

The decision-making process is normally modeled as an optimization problem. A generic optimization problem is represented in the following succinct form in equation (1.1) where we have some continuous variables $x$, to represent the decisions being made (e.g., sizing decision of a reactor), $0-1$ variables $y$ to represent the discrete choices (e.g., select a given reactor or not), an objective function $f$ to minimize or maximize (e.g., to minimize the total cost), and some constraints $g$, $h$, that the variables have to satisfy (e.g., product specifications, mass

balances).

$$\begin{aligned} \min_{x,y} \quad & f(x,y;\theta) \\ \text{s.t.} \quad & g(x,y;\theta) \leq 0 \\ & h(x,y;\theta) = 0 \\ & x \in \mathbb{R}^{n^x}, y \in \{0,1\}^{n^y} \end{aligned} \tag{1.1}$$

Variables $x$ are continuous variables with dimension $n^x$, which can take real values. Variables $y$ are binary variables with dimension $n^y$, which can only take values 0 or 1. Binary variables are usually used to represent logic relations or choices, e.g., whether to install a given chemical plant or not. Vector $\theta$ represents parameters involved in the optimization problem, such as product demand, unit costs of some processes. If we assume that those parameters are known with certainty, problem (1.1) is a deterministic optimization problem. A deterministic optimization problem can be classified into several categories depending on the forms of $f$, $g$, $h$, $x$, $y$.

- some of $f$, $g$, $h$, are nonlinear functions. Problem (1.1) is a mixed-integer nonlinear program (MINLP).

- $f$, $g$, $h$ are all linear functions. Problem (1.1) becomes a mixed-integer linear program (MILP).

- some of $f$, $g$, $h$, are nonlinear functions and there is no $y$ variable, i.e., $n^y = 0$. Problem (1.1) becomes a nonlinear program (NLP).

- $f$, $g$, $h$, are linear functions and there is no $y$ variable, i.e., $n^y = 0$. Problem (1.1) becomes a linear program (LP).

The four different mathematical programs, MINLP, MILP, NLP, LP, are chosen depending on the nature of the problem. For problems in chemical engineering, nonlinear equations are often used to describe thermodynamic or kinetic behavior. Integer variables can be used to describe process synthesis problems, e.g., a binary variable can describe whether a

given distillation column exists or not in a chemical flowsheet. For a detailed treatment of deterministic optimization methods, we refer to the textbook by Grossmann (2021).

In deterministic optimization models, the parameters $\theta$ are assumed to be known. However, in practice, uncertainties are prevalent in process systems due to inaccurate measurement, forecast error, or lack of information. For example, uncertainties in supply chain management can arise from future customer demand, potential network disruption, or even the spread of a pandemic. Failing to consider uncertainties in the decision-making process may lead to suboptimal or even infeasible solutions. To hedge against the uncertainties in process systems, several mathematical frameworks have been used by the PSE community including stochastic programming, chance-constrained programming (Li et al., 2008), and robust optimization (Lappas and Gounaris, 2016). The three approaches are different in their degrees of risk aversion and ways of characterizing uncertainties. Stochastic programming (SP) is a risk-neutral approach, which seeks to optimize the expected outcome over a known probability distribution. Chance-constrained programming can be seen as solving a stochastic program with some probabilistic constraints, which specify that some constraints with uncertain parameters are satisfied with a given level of probability. For example, a chance constraint can specify that a budget/cost that should not exceed a certain threshold. Chance constrained programming offers modeling flexibility to deal with reliability issues and have important connections with risk management. Robust optimization is another risk-averse approach, which seeks to ensure feasibility and optimize the "worst-case" over a pre-defined uncertainty set. Robust optimization problems typically involve min-max type of operators. A summary of the three approaches is shown in Table 1.2.

Table 1.2: Summary of stochastic programming, chance-constrained programming, and robust optimization

| method | degree of risk aversion | characterization of uncertainty |
|---|---|---|
| stochastic programming | risk neutral | probability distribution |
| chance-constrained programming | risk averse | probability distribution, risk level |
| robust optimization | worst case scenario | uncertainty set |

Besides these three approaches, there are a number of mathematical frameworks to model

decision-making under uncertainty, such as Markov Decision Process (MDP). Powell (2019) unifies 15 communities in optimization under uncertainty in a single framework. Reviewing all the 15 approaches is not within the scope of this thesis. Interested readers can refer to the relevant papers (Powell, 2019, 2016).

### 1.1.3 Two-stage stochastic mixed-integer programming

Two-stage stochastic mixed-integer programs (SMIPs) is a framework to model decision-making uncertainty that involves discrete decisions. In this section, we provide a literature review on the history and recent advances of two-stage SMIPs. The detailed mathematical formulations and algorithms are reviewed in chapter 2.

Traditional L-shaped method (Van Slyke and Wets, 1969) can only be applied to linear SMIPs with continuous recourse, but cannot be applied to solve SMIPs with (mixed)-integer recourse. Applications for SMIPs with integer recourse include stochastic server location problem (Ntaimo and Sen, 2005), dynamic capacity acquisition and allocation (Ahmed and Garcia, 2003), stochastic unit commitment problem (Ryan et al., 2013), etc. In the past two decades, there have been algorithmic developments for solving linear SMIPs with integer recourse.

A summary of decomposition algorithms for solving SMIPs with different assumptions on the types of variables and constraints both stage 1 and stage 2 of two-stage stochastic programs is shown in Table 1.3. The types of variables can be binary variables $\mathbb{B}$, continuous variables $\mathbb{R}$, and integer variables $\mathbb{Z}$. The functions that define the constraints can be linear **L**, convex nonlinear **C**, and nonconvex nonlinear **NC**. Since $\mathbb{B} \subset \mathbb{Z}$, if one algorithm can solve problems with integer variables in a given stage, only the box for integer variables is checked in Table 1.3.

The pioneering work is reported by Laporte and Louveaux (1993) who propose integer cuts for two-stage SMIPs with pure binary first stage variables. Benders decomposition algorithms that are based parametric cutting plane algorithms have also been proposed where the parametric cuts, such as disjunctive cuts (Angulo et al., 2016; Qi and Sen, 2017;

Ntaimo, 2010; Ntaimo and Tanner, 2008; Sen and Sherali, 2006), RLT (Sherali and Zhu, 2006; Sherali and Fraticelli, 2002), Gomory mixed-integer cuts (Gade et al., 2014; Zhang and Küçükyavuz, 2014), are used to convexify the (mixed)-integer subproblems. Carøe and Schultz (1999) propose a dual decomposition-based branch and bound algorithm. For a review of recent advances in SMIPs, we refer to the tutorial by Küçükyavuz and Sen (2017).

Although there have been algorithmic advances in linear SMIPs, the decomposition algorithms to address the nonlinear counterpart, stochastic mixed-integer nonlinear programs (SMINLPs), are few. For convex SMINLPs (the relaxed nonlinear functions in SMINLPs are all convex), Mijangos (2015) proposes an algorithm based on Branch-and-Fix Coordination method (Alonso-Ayuso et al., 2003) for convex problems with 0-1 mixed-integer variables in the first stage and only continuous variables in the second stage. Atakan and Sen (2018) propose a progressive hedging-based branch-and-bound algorithm for convex SMINLP, which works well in the case of pure binary first stage variables and continuous recourse. For nonconvex SMINLPs (the nonlinear functions in SMINLPs can be nonconvex), the nonconvexities in stage 2 are two-fold: first, we have nonlinear nonconvex functions in stage 2; second, some of the stage 2 variables need to satisfy integrality constraints. Li et al. (2011c) propose a nonconvex generalized Benders decomposition (NGBD) algorithm for problems with pure binary first stage variables where they relax the nonconvex functions in stage 2 by their convex relaxations in order to derive valid Benders-like cuts. The convergence of NGBD relies on using the integer cuts similar to the one proposed by Laporte and Louveaux (1993). NGBD and its variations have been applied to stochastic pooling problems (Li et al., 2011a, 2012b), integrated process design and operation problems (Li et al., 2012a), etc.

However, if the first stage variables are mixed-integer, the integer cuts used in Li et al. (2011c) are no longer applicable. For the more general case where the first stage variables can be mixed-integer, Ogbe and Li (2018) propose a joint decomposition algorithm where they reformulate the two-stage problem so that all the nonconvexities are in the first stage. Therefore, the Benders subproblems in Ogbe and Li (2018) are continuous and convex. The authors also add Lagrangean cuts to the Benders master problem to tighten the master

Table 1.3: Assumptions made in the SMIP literature cited. $\mathbb{B}$: binary variables , $\mathbb{R}$: continuous variables, $\mathbb{Z}$: integer variables, linear: **L**, convex nonlinear: **C**, nonconvex nonlinear: **NC**

| | First stage | | | | | | Second stage | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathbb{B}$ | $\mathbb{R}$ | $\mathbb{Z}$ | **L** | **C** | **NC** | $\mathbb{B}$ | $\mathbb{R}$ | $\mathbb{Z}$ | **L** | **C** | **NC** |
| Van Slyke and Wets (1969) | | ✓ | | ✓ | | | | ✓ | | ✓ | | |
| Laporte and Louveaux (1993) | ✓ | | | ✓ | | | ✓ | ✓ | | ✓ | | |
| Angulo et al. (2016) | ✓ | | | ✓ | | | ✓ | ✓ | | ✓ | | |
| Qi and Sen (2017) | | ✓ | ✓ | ✓ | | | | ✓ | ✓ | ✓ | | |
| Ntaimo (2010) | ✓ | ✓ | | ✓ | | | ✓ | ✓ | | ✓ | | |
| Ntaimo and Tanner (2008) | ✓ | ✓ | | ✓ | | | ✓ | ✓ | | ✓ | | |
| Sen and Sherali (2006) | ✓ | | | ✓ | | | ✓ | ✓ | | ✓ | | |
| Sherali and Zhu (2006) | ✓ | ✓ | | ✓ | | | ✓ | ✓ | | ✓ | | |
| Sherali and Fraticelli (2002) | ✓ | | | ✓ | | | ✓ | ✓ | | ✓ | | |
| Gade et al. (2014) | ✓ | | | ✓ | | | | | ✓ | ✓ | | |
| Zhang and Küçükyavuz (2014) | | | ✓ | ✓ | | | | | ✓ | ✓ | | |
| Carøe and Schultz (1999) | | ✓ | ✓ | ✓ | | | | ✓ | ✓ | ✓ | | |
| Mijangos (2015) | ✓ | ✓ | | | ✓ | | | ✓ | | | ✓ | |
| Atakan and Sen (2018) | ✓ | | | | ✓ | | | ✓ | | | ✓ | |
| Li and Grossmann (2018a) | ✓ | ✓ | | | ✓ | | ✓ | ✓ | | | ✓ | |
| Li and Grossmann (2019b) | ✓ | ✓ | | | ✓ | | ✓ | ✓ | | | ✓ | |
| Li et al. (2011c) | ✓ | | | | | ✓ | ✓ | ✓ | | | | ✓ |
| Ogbe and Li (2018) | ✓ | ✓ | | | | ✓ | ✓ | ✓ | | | | ✓ |
| Cao and Zavala (2017) | ✓ | ✓ | | | | ✓ | ✓ | ✓ | | | | ✓ |
| Kannan (2018) | ✓ | ✓ | | | | ✓ | ✓ | ✓ | | | | ✓ |

problem. The advantage of Ogbe and Li (2018) is that explicit spatial branch and bound can be avoided. However, a large-scale master problem may need to be solved by global solvers when the number of scenarios is large. Cao and Zavala (2017) propose a perfect information-based branch and bound algorithm that solves nonconvex SMINLPs to global optimality. The authors relax the nonanticipativity constraints (NACs) of the two-stage problem to derive a valid bound at each node of the spatial branch and bound process where they only need to branch on the first stage variables. Kannan (2018) propose a modified Lagrangean relaxation-based (MLR) branch and bound algorithm, which is similar to the algorithm proposed by Carøe and Schultz (1999) except that MLR only dualizes the NACs corresponding to the continuous first stage variables. Therefore, the Lagrangean subproblems still have NACs corresponding to integer variables, which are solved by NGBD proposed by Li et al. (2011c). The authors prove that MLR has finite $\epsilon$-convergence.

## 1.2 Energy systems

### 1.2.1 Shale gas pad development

According to the Annual Energy Outlook 2019 published by the EIA (U.S. Energy Information Administration, 2019a), the production of natural gas is expected to grow by 50% in the next 30 years. The growth in natural gas production supports increasing domestic consumption, particularly in the industrial and electric power sectors, and higher levels of natural gas exports. In Figure 1.2, the prediction of the consumption and the production of dry natural gas in the U.S. is shown under different sensitivity cases. We can observe that there will likely be an increase in both the production and the consumption of dry natural gas even in the case of low economic growth and low oil price.

The increase in natural gas production is driven by continued development of lower-cost shale gas resources. The Annual Energy Outlook 2019 (U.S. Energy Information Administration, 2019a) shows that dry natural gas production from shale gas and tight oil continues to grow in both share and absolute volume because of the sheer size of the associated resources,

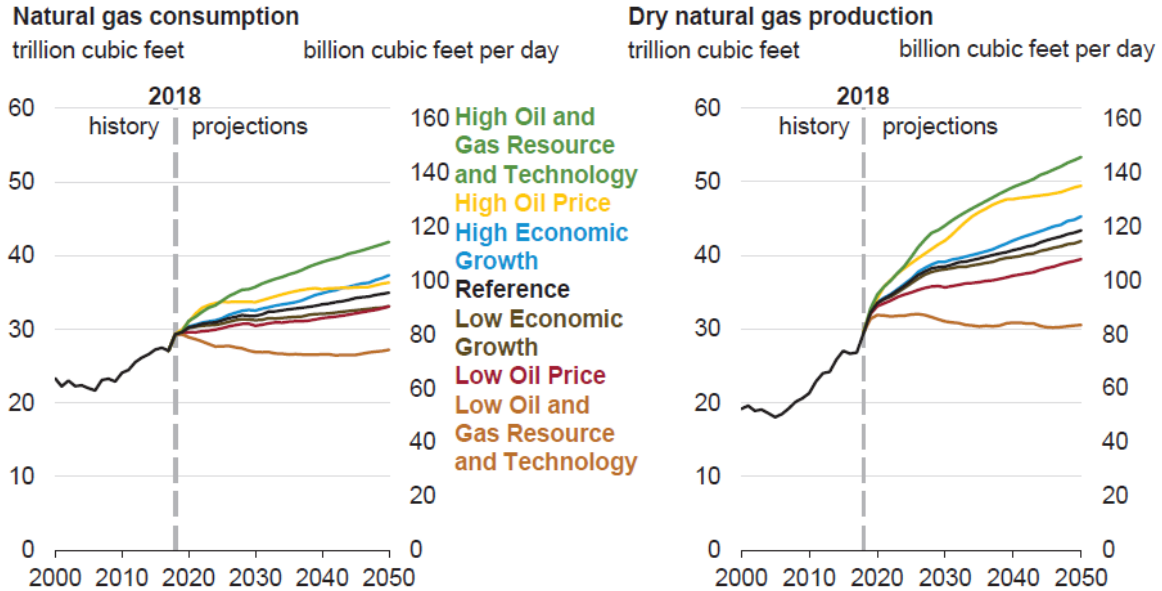


Figure 1.2: The prediction of natural gas production and consumption in different sensitivity cases

which extend over nearly 500,000 square miles, and because of improvements in technology that allow for the development of these resources at lower costs. Shale gas is expected to account for 90% of U.S. dry natural gas production in 2050.

Despite the significant role that shale gas plays in the energy industry, the shale gas industry is still young (A.Morris, 2015). Recently, the optimization of shale gas systems has drawn increasing attention in academia. Many shale gas companies (Markus G. Drouven, 2018) also start to apply optimization models to make better use of their resources. A common problem that arises in the shale gas industry is the operation of a single shale gas wellpad as it affects the whole natural gas supply chain. Recently, Ondeck et al. (2019) proposed an MILP (mixed-integer linear programming) model that considers the economical development of a shale gas pad and the optimal production of shale gas.

To develop a well for production (Figure 1.3), four operations must take place in the following order: 1) top setting (TS), 2) horizontal drilling (HZ), 3) hydraulic fracturing (FRAC), and 4) turning in line (TIL). The first operation, top setting, is the process of drilling a well down to the selected shale gas formation and properly encasing the well to prevent the release of gas and other chemicals into the ground surrounding the well bore.

Once the vertical part of the well has been developed, the next step is horizontal drilling (HZ), which is followed by fracturing (FRAC). Fracturing refers to the injection of a fracturing fluid into a geologically tight formation under high pressure of up to 70 MPa. Once the fracturing is complete, the well can be turned in line to release the gas. Based on the desired production, the entire well or sections of the well can be turned in line. From start to finish, the process of completing a well can take anywhere from a few weeks to two months, based on the geology of the subsurface, the length of the well, and the availability of resources.

Ondeck et al. (2019) point out that traditionally, upstream operators complete one operation for all the wells in a single wellpad before moving to the next operation to reduce the mobilization of their crew and equipment. However, there are several drawbacks for this type of operating strategy. First, since all the wells are turned in line around the same time, there is a dramatic increase of natural gas production in this single pad. Therefore, pipelines with large capacities are needed in order to deliver the natural gas to the customers. Second, since the wells cannot be turned in line until all the first three operations are completed for all the wells in the pad, the lateness of turning in line incurs a loss in the net present value (NPV) compared with the strategy in which some of the wells are turned in line before all the first three operations are completed for all the three wells. Third, the production of a typical shale gas well decreases sharply after the first year of turning in line. Therefore, if all the wells are turned in line around the same time, there will be a period with little gas production.

From our analysis on the conventional strategy of single pad operations, there is a tradeoff between avoiding the mobilization costs, and turning the well in line earlier to maximize the production NPV. In the paper by Ondeck et al. (2019), the authors perform a sensitivity analysis by varying the mobilization costs. Considering mobilization costs tends to affect the schedule of the single pad development.

We review the literature related to the optimization of shale gas systems. We first review the deterministic models. The models that consider uncertainty are relatively few and are reviewed at the end of this section. The deterministic optimization models for shale gas

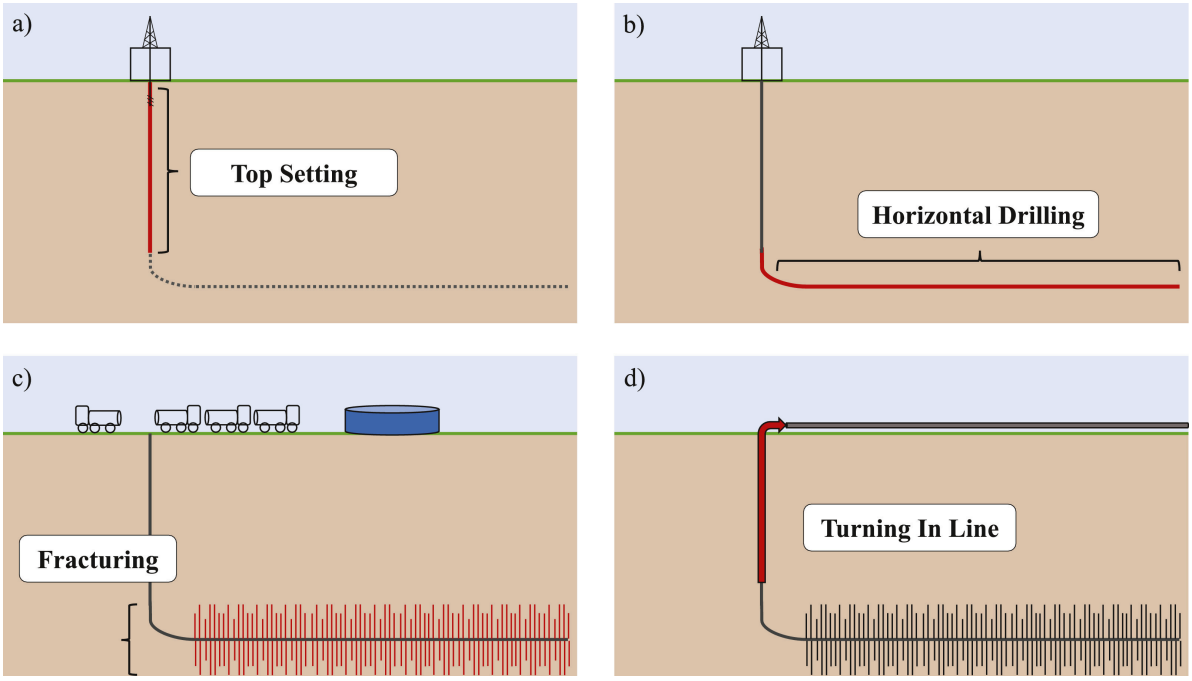


Figure 1.3: The four operations necessary to develop a well to completion: (a) top setting, (b) horizontal drilling, (c) hydraulic fracturing, and (d) turning in line.

development can be grouped into three levels based on the time scale: design/planning, scheduling, and operating.

The design/planning models correspond to long term decisions, usually for more than a decade. The decisions involved in these long-term models usually include the design of shale gas network, such as the layout of pipelines, and planning decisions such as the number of wells to drill. The constraints usually include mass/resource balance constraints, network design constraints, etc. The objective is usually to maximize NPV. Cafaro and Grossmann (2014) present a mixed-integer nonlinear programming (MINLP) model for the strategic planning, design, and development of the shale gas supply chain network where they determine planning decisions such as the number of wells to drill at every location, and design decisions, such as the size of gas processing plants, the length and diameter of the pipelines so as to maximize the net present value of the project. Following the work of Cafaro and Grossmann (2014), Drouven and Grossmann (2016) propose an MINLP model that involves planning, design, and strategic decisions such as where, when, and how many shale gas wells

to drill, where to lay out gathering pipelines, as well as which delivery agreements to arrange. Guerra et al. (2016) propose an optimization framework for the integration of water management and shale gas supply chain design. Gao and You (2015b) propose an MINLP model that addresses the life cycle economic and environmental optimization of shale gas supply chain network design and operations. Arredondo-Ramírez et al. (2016) use a disjunctive programming-based approach to account for complex logical relationships in the optimal planning of shale gas exploitation and infrastructure development.

The optimal scheduling of operations for shale gas development considers shorter time period than in planning/design. The constraints in the scheduling models usually include sequence of operations, resource availability constraints. The objective can be maximizing NPV or minimizing environmental impact. Knudsen et al. (2014) formulate an MILP model, which is solved using Lagrangean decomposition, for shut-in scheduling in large, multi-well shale-gas systems. Cafaro et al. (2016) propose a continuous time MINLP model and a discrete time MILP model for planning shale gas well refracture treatments. Cafaro et al. (2018) propose a continuous time optimization model for planning multiple refracture treatments over the lifespan of a shale gas well. Yang et al. (2014) propose an MILP model that addresses source water acquisition, waste water production, reuse and recycle, and subsequent transportation, storage, and disposal in shale gas production. Ondeck et al. (2019) consider the scheduling of the four operations in a single wellpad.

At the operational level, decisions are made on a daily to weekly time scale. Forouzanfar and Reynolds (2014) formulate a continuous optimization model to simultaneously optimize the number of wells, their locations and controls. Wilson and Durlofsky (2013) develop a surrogate model to accurately model complex shale gas reservoirs and further use the model for shale gas field development optimization. Others (Ma et al., 2013; Yu and Sepehrnoori, 2013) have studied the placement of hydraulic fracture stages.

The models for shale gas development that consider uncertainty are relatively few. Drouven et al. (2017) apply two-stage stochastic programming in a moving horizon approach for optimal shale well development and refracturing planning under exogenous gas price un-

certainty and endogenous well performance uncertainty. Gao and You (2015a) develop a two-stage stochastic mixed integer linear fractional programming (SMILFP) model to optimize the levelized cost of energy generated from shale gas under uncertainty of estimated ultimate recovery (EUR). Guerra et al. (2019) develop a two-stage stochastic model embedded in a moving horizon strategy to dynamically solve the planning of shale well development and refracturing. Zeng and Cremaschi (2017) propose stochastic programming models to the artificial lift infrastructure planning for shale gas producing wells.

### 1.2.2 Generation and transmission expansion planning of power systems

Generation expansion planning (GEP) of power systems involves determining the optimal size, location, and construction time of new power generation plants, while minimizing the total cost over a long-term planning horizon (Conejo et al., 2016b; Koltsaklis and Dagoumas, 2018). There is a growing interest to use mathematical programming models to solve generation expansion planning problems (Lara et al., 2018; Sadeghi et al., 2017; Oree et al., 2017). Conventional power units are dispatchable thermal power plants that can provide stable power output. Due to computational tractability concerns, generation expansion models can ignore short-term operating decisions. However, with the increased penetration of renewable generation technologies, such as solar and wind, power systems nowadays need to be more flexible so as to adjust to the volatile power generation from renewables. In this case, operations decisions, such as unit commitment, ramping decisions, become important to assess system feasibility (Ding and Somani, 2010; Koltsaklis and Georgiadis, 2015; Pina et al., 2013; Poncelet et al., 2014; Shortt and O'Malley, 2010; Flores-Quiroz et al., 2016; Palmintier and Webster, 2011; Lara et al., 2018; Lohmann and Rebennack, 2017). Due to the incorporation of short-term operating constraints into the long-term planning problem, the integrated model is computationally challenging. In order to solve such multi-scale problem efficiently, Lara et al. (2018) use nested Benders decomposition to solve a GEP model with unit commitment. Lohmann and Rebennack (Lohmann and Rebennack, 2017) develop

a tailored Generalized Benders Decomposition algorithm.

Transmission expansion planning (TEP) refers to installing new transmission lines or expanding the capacities of existing transmission lines in a power system. Bahiense et al. (2001) propose a mixed integer disjunctive model for transmission network expansion. (Alguacil et al., 2003) propose an MILP model that considers losses and guarantees convergence to optimality for the TEP. Zhang et al. (2013) propose an improved model that includes a linear representation of reactive power, off-nominal bus voltage magnitudes and network losses. For a more detailed review of of TEP models and algorithms, we refer the readers to the review papers (Hemmati et al., 2013; Ude et al., 2019).

GEP and TEP are generally solved as two independent optimization problems since the market agents addressing these two problems are different. GEP pertains to producers, while TEP pertains to a regulated planner. However, the significant penetration of renewables into power systems may lead to their concentration in remote areas not well connected to the load demand (Koltsaklis and Dagoumas, 2018). Therefore, installing renewables in those remote areas could compromise transmission expansion. The recognition of transmission's interaction with generation expansion has motivated the development of co-optimization methods to consider the tradeoffs between generation and transmission expansion (Krishnan et al., 2016). Several works have been reported to simultaneously optimize generation and transmission expansion planning (GTEP) (Pozo et al., 2012; Aghaei et al., 2014). We refer to Table 1 of the review paper (Koltsaklis and Dagoumas, 2018) for a long list of works. See also the review paper (Gacitua et al., 2018).

A number of related works consider uncertainties in the planning problem using two-stage or multistage stochastic programming (Lara et al., 2019; O'Neill et al., 2013; Liu et al., 2017b), robust optimization (Mejía-Giraldo and McCalley, 2013; Baringo and Baringo, 2017). (Le Cadre et al., 2015; Pozo et al., 2012) apply game theory or multi-level optimization to characterize the interaction of the participants in the markets.

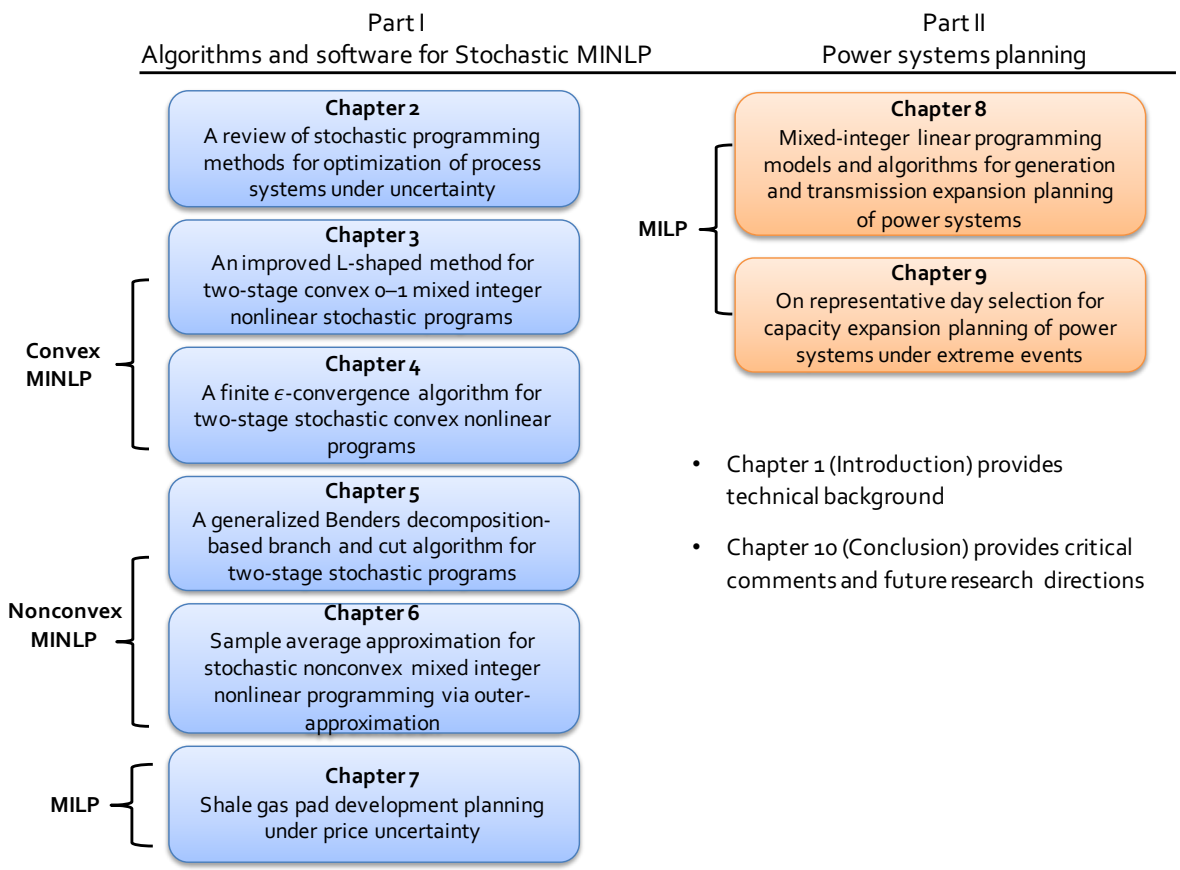


Figure 1.4: Overview of this thesis

## 1.3 Overview of the thesis

Figure 1.4 shows a graphic overview of the topics in Chapters 1-10. The following subsections provide an abstract of all the chapters.

### Chapter 2

In chapter 2, we review the basic concepts and recent advances of a risk-neutral mathematical framework called "stochastic programming" and its applications in solving process systems engineering problems under uncertainty. This review intends to provide both a tutorial for beginners without prior experience and a high-level overview of the current state-of-the-art developments for experts in process systems engineering and stochastic programming.

The mathematical formulations and algorithms for two-stage and multistage stochastic programming are reviewed with illustrative examples from process industries. The differences between stochastic programming under exogenous uncertainty and endogenous uncertainties are discussed. The concepts and several data-driven methods for generating scenario trees are also reviewed.

## Chapter 3

In chapter 3, we propose an improved L-shaped method to solve large-scale two-stage convex 0-1 mixed-integer nonlinear stochastic programs with mixed-integer variables in both first and second stage decisions and with relatively complete recourse. To address the difficulties in solving large problems, we propose a Benders-like decomposition algorithm that includes both (strengthened) Benders cuts and Lagrangean cuts in the Benders master problem. The proposed algorithm is applied to solve a batch plant design problem under demand uncertainty, and a planning problem under demand and price uncertainty. It is shown that the proposed algorithm outperforms the commercial solvers, DICOPT, SBB, Alpha-ECP, and BARON, when the deterministic equivalent problems involve a large number of scenarios. Also, although the proposed algorithm cannot close the duality gap, it is proved that it can yield a lower bound that is at least as tight as the one from Lagrangean decomposition.

## Chapter 4

In chapter 4, we propose a generalized Benders decomposition-based branch and bound algorithm, GBDBAB, to solve two-stage convex mixed-binary nonlinear stochastic programs with mixed-binary variables in both first and second-stage decisions. In order to construct the convex hull of the MINLP subproblem for each scenario in closed-form, we first represent each MINLP subproblem as a generalized disjunctive program (GDP) in conjunctive normal form (CNF). Second, we apply basic steps to convert the CNF of the MINLP subproblem into disjunctive normal form (DNF) to obtain the convex hull of the MINLP subproblem. We prove that GBD is able to converge for the problems with pure binary variables given that the

convex hull of each subproblem is constructed in closed-form. However, for problems with mixed-binary first and second-stage variables, we propose an algorithm, GBDBAB, where we may have to branch and bound on the continuous first-stage variables to obtain an optimal solution. We prove that the algorithm GBDBAB can converge to $\epsilon$-optimality in a finite number of steps. Since constructing the convex hull can be expensive, we propose a sequential convexification scheme that progressively applies basic steps to the CNF. Computational results on a problem with quadratic constraints, a constrained layout problem, and a planning problem, demonstrate the effectiveness of the algorithm.

## Chapter 5

In chapter 5, we propose a generalized Benders decomposition-based branch and cut algorithm for solving two-stage stochastic mixed-integer nonlinear programs (SMINLPs) with mixed binary first and second stage variables. At a high level, the proposed decomposition algorithm performs spatial branch and bound search on the first stage variables. Each node in the branch and bound search is solved with a Benders-like decomposition algorithm where both Lagrangean cuts and Benders cuts are included in the Benders master problem. The Lagrangean cuts are derived from Lagrangean decomposition. The Benders cuts are derived from the Benders subproblems, which are convexified by cutting planes, such as rank-one lift-and-project cuts. We prove that the proposed algorithm converges in the limit. We apply the proposed algorithm to a stochastic pooling problem, a crude selection problem, and a storage design problem. The performance of the proposed algorithm is compared with a Lagrangean decomposition-based branch and bound algorithm and solving the corresponding deterministic equivalent with the solvers including BARON, ANTIGONE, and SCIP.

## Chapter 6

In chapter 6, we propose a sample average approximation-based outer-approximation algorithm (SAAOA) that can address nonconvex two-stage stochastic programs (SP) with any continuous or discrete probability distributions. Previous work has considered this ap-

proach for convex two-stage SP (Wei and Realff, 2004). The SAAOA algorithm does internal sampling within a nonconvex outer-approximation algorithm where we iterate between a mixed-integer linear programming (MILP) master problem and a nonconvex nonlinear programming (NLP) subproblem. We prove that the optimal solutions and optimal value obtained by the SAAOA algorithm converge to the optimal solutions and the optimal value of the true SP problem as the sample size goes to infinity. The convergence rate is also given to estimate the sample size. Since the theoretical sample size estimate is too conservative in practice, we propose an SAAOA algorithm with confidence intervals for the upper bound and the lower bound at each iteration of the SAAOA algorithm. Two policies are proposed to update the sample sizes dynamically within the SAAOA algorithm with confidence intervals. The proposed algorithm works well for the special case of pure binary first stage variables and continuous stage two variables since in this case the nonconvex NLPs can be solved for each scenario independently. The proposed algorithm is tested with a stochastic pooling problem and is shown to outperform the external sampling approach where large scale MINLPs need to be solved.

## Chapter 7

In chapter 7, we study shale gas pad development under natural gas price uncertainty. We optimize the sequence of operations, gas curtailment and storage on a single pad to maximize the net present value (NPV). The optimization problem is formulated as a mixed-integer linear programming (MILP) model, which is similar to the one proposed by Ondeck et al. Ondeck et al. (2019). We investigate how natural gas price uncertainty affects the operation strategy in the pad development. Both two-stage and multi-stage stochastic programming are used as the mathematical framework to hedge against uncertainty. Our case study shows that there is value in using stochastic programming when the price variance is high. However, when the variance of the price is low, solving the stochastic programming problems does not create additional value compared with solving the deterministic problem.

## Chapter 8

Chapter 8 is focused on generation transmission expansion planning of power systems. With the increasing penetration of renewable generating units, especially in remote areas not well connected with load demand, there is growing interest to co-optimize generation and transmission expansion planning (GTEP) in power systems. Due to the volatility in renewable generation, a planner needs to include the operating decisions into the planning model to guarantee feasibility. However, solving the GTEP problem with hourly operating decisions throughout the planning horizon is computationally intractable. Therefore, we propose several spatial and temporal simplifications to the problem. Built on the generation expansion planning (GEP) formulation of Lara et al. (2018), we propose a mixed-integer linear programming formulation for the GTEP problem. Three different formulations, i.e., a big-M formulation, a hull formulation, and an alternative big-M formulation, are reported for transmission expansion. We theoretically compare the tightness of the LP relaxations of the three formulations. The proposed MILP GTEP model typically involves millions or tens of millions of variables, which makes the model not directly solvable by the commercial solvers. To address this computational challenge, we propose a nested Benders decomposition algorithm and a tailored Benders decomposition algorithm that exploit the structure of the GTEP problem. Using a case study from Electric Reliability Council of Texas (ERCOT), we are able to show that the proposed tailored Benders decomposition outperforms the nested Benders decomposition. The coordination in the optimal generation and transmission expansion decisions from the ERCOT study implies that there is additional value in solving GEP and TEP simultaneously.

## Chapter 9

Capacity expansion planning (CEP) of power systems determines the optimal future generation mix and/or transmission lines. Due to the increasing penetration of renewables, CEP has to capture the hourly variations of renewable generator outputs and load demand. Since CEP problems typically involve planning horizons of several years, solving the fullspace

models where the operating decisions corresponding to all the days is intractable. Therefore, some "representative days" are selected as a surrogate to the fullspace model. We present an input-based and a cost-based approach in combination with the $k$-means and the $k$-medoids clustering algorithms for representative day selection. The mathematical properties of the proposed algorithms are analyzed, including an approach to calculate the "optimality gap" of the investment decisions obtained from the representative day model to the fullspace model, and the relationship between the clustering error and the optimality gap. To capture the extreme events, two novel approaches, i.e., a "load shedding cost" approach and a "highest cost" approach, are proposed to identify the "extreme days". We conclude with a case study based on the Electric Reliability Council of Texas (ERCOT) region, which compares the different approaches and the effects of adding the extreme days.

## Chapter 10

Chapter 10 provides a critical review of the work in this thesis, along with a summary of its contributions and suggestions for future work.

# Part I

# Algorithms and Software for Stochastic Mixed-integer Nonlinear Programming

# Chapter 2

# A Review of Stochastic Programming Methods for Optimization of Process Systems under Uncertainty

## 2.1 Two-stage stochastic programming

Two-stage stochastic programming is a special case of stochastic programming. In this section, we describe the mathematical formulations, algorithms and illustrative examples for two-stage stochastic programming.

### 2.1.1 Two-stage stochastic mixed integer linear programs

For simplicity of presentation, we fist consider stochastic MILP problems.

#### 2.1.1.1 Mathematical formulation

In stochastic programming, it is assumed that the probability distributions of the uncertain parameters are known *a priori*. The uncertainties are usually characterized by some discrete realizations of the uncertain parameters as an approximation to the real probability

distribution. For example, the realizations of the demand for a product can have three different values which represent high, medium, and low demand, respectively. Each realization is defined as a scenario. The objective of stochastic programming is to optimize the expected value of an objective function (e.g., the expected cost) over all the scenarios.

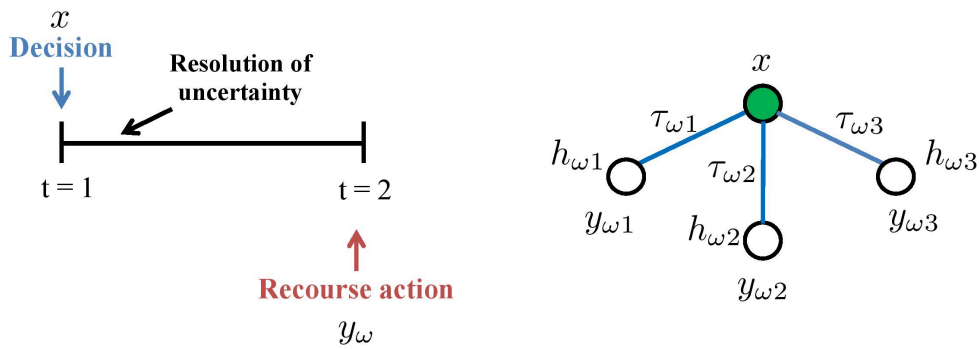


Figure 2.1: Two-stage problem: conceptual representation (left); scenario tree (right) where $x$ represents the first stage decisions, $y_\omega$ represents the stage two decisions for scenario $\omega$. $\tau_\omega$, $h_\omega$ represent the probability and the uncertain right hand side of scenario $\omega$, respectively.

A special case of stochastic programming is two-stage stochastic programming (Figure 2.1). Specifically, stage one decisions are made 'here and now' at the beginning of the period, and are then followed by the resolution of uncertainty. Stage two decisions, or recourse decisions, are taken 'wait and see' as corrective action at the end of the period. One common type of two-stage stochastic program is mixed-integer linear program presented in (2.1). $\Omega$ is the set of scenarios. $\tau_\omega$ is the probability of scenario $\omega$. $x$ represent the first-stage decisions. $y_\omega$ represent the second-stage decisions in scenario $\omega$. The uncertainties are reflected in the matrices (vectors), $W_\omega$, $h_\omega$, $T_\omega$ shown in Equation (2.1). In the literature, $W_\omega$ is called the "recourse matrix"; $T_\omega$ is called the "technology matrix". On the right of Figure 2.1, an example of a "scenario tree" that has three scenarios with uncertain $h_\omega$ is used to represent the realizations of uncertainties on the right hand side.

For problem (2.1), both the first and the second stage decisions are mixed-binary. Let $I = \{1, 2, \cdots, n\}$ be the index set of all the first stage variables. $I_1 \subseteq I$ is the subset for indices of the binary first stage variables. Let $J = \{1, 2, \cdots, m\}$ be the index set of all the second stage variables. $J_1 \subseteq J$ is the subset for the indices of the binary second stage

variables. $x^{ub}$ is a vector that represents the upper bound of all the first stage variables. $y_\omega^{ub}$ is a vector that represents the upper bound of all the second stage variables. The problem described by (2.1) is a two-stage stochastic mixed-integer linear program (TS-MILP). If both $J_1$ and $I_1$ are empty sets, (2.1) reduces to a two-stage stochastic linear programming problem (TS-LP). (2.1) is often referred to as the *deterministic equivalent* or the *extensive form* of the two-stage stochastic program since (2.1) can be solved in the same way as if we were solving a deterministic optimization problem.

$$\begin{aligned}
\min \quad & c^\top x + \sum_{\omega\in\Omega} \tau_\omega d_\omega^\top y_\omega \\
\text{s.t.} \quad & Ax \le b \\
& W_\omega y_\omega \le h_\omega - T_\omega x \quad \forall \omega \in \Omega \\
& x \in X, \quad X = \left\{x : x_i \in \{0,1\}, \forall i \in I_1,\ 0 \le x \le x^{ub}\right\} \\
& y_\omega \in Y_\omega \quad \forall \omega \in \Omega, \quad Y_\omega = \left\{y_\omega : y_{\omega j} \in \{0,1\}, \forall j \in J_1,\ 0 \le y_\omega \le y_\omega^{ub}\right\}
\end{aligned} \tag{2.1}$$

#### 2.1.1.2 Process network problem

To show how two-stage stochastic programming can be applied to a process systems engineering problem, we provide a process network design problem under demand uncertainty. Through this example, we also aim to show that the solutions obtained from a stochastic program can be different from solving a deterministic problem where the uncertain parameters are fixed at their expected value.

Consider producing a chemical C which can be manufactured with either process 2 or process 3, both of which use chemical B as raw material. B can be purchased from another company and/or manufactured with process 1 which uses A as a raw material. The demand for chemical C, denoted as $d$, is the source of uncertainty. The superstructure of the process network is shown in Figure 2.2, which outlines all the possible alternatives to install this chemical plant. The alternatives include (1) All three processes are selected. (2) A true subset of the three processes are selected. (3) None of the three processes are selected.

Following the time realization framework described in Figure 2.1, the problem is formulated as a two-stage stochastic program. The chemical plant has to be first installed

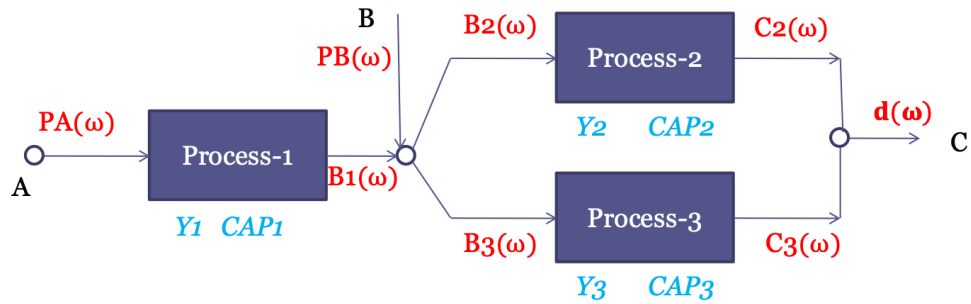


Figure 2.2: Superstructure for the process network problem.

before the demand for the product is realized and the plant starts production. Therefore, the first-stage decisions are investment decisions on the three processes, which include binary variables $Y_i$ to denote whether process $i$ is selected and continuous variables $CAP_i$ to denote the capacity of process $i$ for $i = 1, 2, 3$. We assume that after the plant is installed, the demand for the product is realized. Based on the realizations of the demand, different recourse actions on how to operate the installed plant can be taken, i.e., the second-stage decisions are the material flows. We denote the scenarios as $\omega$ and explicitly state the dependency of the second-stage decision on them by presenting them as functions of $\omega$.

Variables $PA(\omega)$, $PB(\omega)$ represent the purchase amount of chemical A and B, respectively. Other material flows for chemical B and C are shown in the superstructure in Figure 2.2. The MILP formulation is shown as follows.

$$\begin{aligned}\max \quad & -\big(10Y_1 + 15Y_2 + 20Y_3 + CAP_1 + 1.5CAP_2 + 2CAP_3\big) \\ & + \mathop{\mathbb{E}}_{d(\omega)\sim\mathbb{P}}\Big[-4.5PA(\omega) - 9.5PB(\omega) - 0.5PA(\omega) - 0.5B_2(\omega) - 0.5B_3(\omega) + 25C_2(\omega) + 25C_3(\omega)\Big]\end{aligned} \tag{2.2a}$$

$$\text{s.t.} \quad CAP_1 \le U \cdot Y_1, \quad CAP_2 \le U \cdot Y_2, \quad CAP_3 \le U \cdot Y_3, \quad Y_2 + Y_3 \le 1 \tag{2.2b}$$

$$PA(\omega) \le CAP_1, \quad B_2(\omega) \le CAP_2, \quad B_3(\omega) \le CAP_3 \quad \forall\omega \tag{2.2c}$$

$$B_1(\omega) = 0.9PA(\omega), \quad C_2(\omega) = 0.82B_2(\omega), \quad C_3(\omega) = 0.95B_3(\omega) \quad \forall\omega \tag{2.2d}$$

$$B_1(\omega) + PB(\omega) = B_2(\omega) + B_3(\omega) \quad \forall\omega \tag{2.2e}$$

$$C_2(\omega) + C_3(\omega) \leq d(\omega) \quad \forall \omega \tag{2.2f}$$

$$Y_i \in \{0, 1\} \quad \forall i \in \{1, 2, 3\} \tag{2.2g}$$

The objective is maximizing the expected profit, which includes the expected income obtained from selling the final product $(25C_2(\omega) + 25C_3(\omega))$, minus the total cost that includes the fixed $(10Y_1 + 15Y_2 + 20Y_3)$ and variable $(CAP_1 + 1.5CAP_2 + 2CAP_3)$ investment costs in stage one, the expected cost to purchase chemical A and B in different scenarios $(4.5PA(\omega) + 9.5PB(\omega))$, the expected operating cost $(0.5PA(\omega) + 0.5B_2(\omega) + 0.5B_3)$, which is proportional to the amount of input to each process, i.e., $PA(\omega)$, $B_2(\omega)$, $B_3(\omega)$.

Suppose we have a 3-scenario problem where the the demands $d(\omega)$ take values $d(\omega_1) = 8$, $d(\omega_2) = 10$, $d(\omega_3) = 12$, with probabilities $\tau(\omega_1) = 0.25$, $\tau(\omega_2) = 0.5$, $\tau(\omega_3) = 0.25$, respectively. The optimal first-stage decisions are to select processes 1 and 3 with capacities 11.70 and 12.63, respectively, which is shown in Figure 2.3(I). Note that the first-stage decisions are made "here-and-now" and thus are the same for all three scenarios. However, different second-stage decisions are taken for different scenarios as shown in Figure 2.3(II). When the demand is low $d(\omega_1) = 8$, processes 1 and 3 are not operating at their full capacity. For $d(\omega_2) = 10$, process 1 is operating at full capacity but process 3 is not. For $d(\omega_3) = 12$, both installed processes are operating at full capacity. The chemical A produced by process 1 is not able to satisfy the requirement of process 3. Therefore, additional chemical B needs to be purchased from other vendors when the demand is high. The expected profit of the stochastic program is 117.22. This optimal value of the stochastic program is called the value of the recourse problem (RP) in the literature (Birge and Louveaux, 2011) (RP=117.22).

Other than using stochastic programming, an alternative approach is to solve the deterministic model where the demand is fixed at its mean value, i.e., set $d = 10$. The optimal solution for this deterministic model is selecting processes 1 and 3 with capacities being 11.70, and 10.52, respectively as shown in Figure 2.3(III). The only difference from the stochastic solution is that the capacity of process 3 becomes lower. The reason is that the deterministic model is "unaware" of the high demand scenario and therefore makes the capacity of process

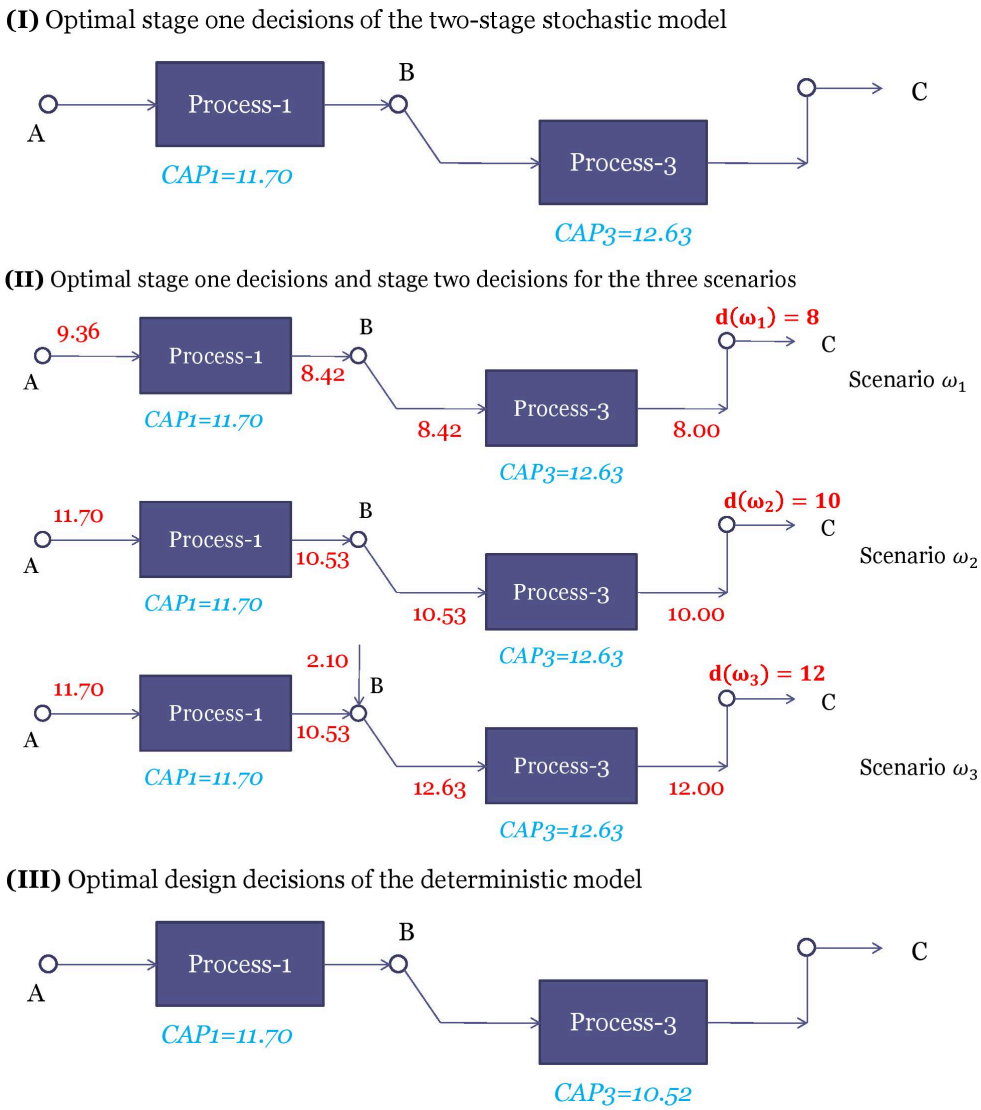


Figure 2.3: **(I)** represents the optimal first stage decisions of the two stage stochastic program. **(II)** describes the optimal stage one decisions and the stage two decisions under scenarios $\omega_1$, $\omega_2$, $\omega_3$. **(III)** represents the optimal design decisions of the deterministic model.

3 to be just enough to satisfy $d = 10$. However, if we use the deterministic solution for $d = 12$, it will result in lost sales. We can fix the first stage solutions to the optimal solutions and evaluate how it performs in the three scenarios by solving each stage two problem separately. An expected profit of 114.20 is obtained. This value is called the expected result of using the expected solution (EEV).

One quantitative metric to evaluate the additional value created by stochastic programming compared with solving the deterministic model at mean value is a concept called *the value of the stochastic solution (VSS)* (Birge and Louveaux, 2011). If the problem is a maximization problem, VSS is defined as,

$$VSS = RP - EEV \tag{2.3}$$

Therefore, the value of the stochastic solution is $117.22 - 114.20 = 3.02$ for the process network problem.

#### 2.1.1.3 Classical decomposition algorithms

One option to solve the stochastic MILP problems described by (2.1) is to solve the deterministic equivalent problem (2.1) directly using commercial solvers like CPLEX , GUROBI. However, solving (2.1) directly can be prohibitive when the number of scenarios is large because the computational time can grow exponentially with the number of scenarios. Due to the difficulties in solving the deterministic equivalent problem, decomposition algorithms such as Lagrangean decomposition (Guignard, 2003; Oliveira et al., 2013) and Benders decomposition (Laporte and Louveaux, 1993; Van Slyke and Wets, 1969) can be applied to solve problem (2.1) more effectively.

A high-level view of the ideas behind Benders decomposition and Lagrangean decomposition is shown in Figure 2.4. The figure shows the structures of the constraint matrices where the columns correspond to the variables and the rows correspond to the constraints. Both decomposition algorithms take advantage of the "almost" block-diagonal structure shown in Figure 2.4.

Benders decomposition (Laporte and Louveaux, 1993; Van Slyke and Wets, 1969), also referred to as L-shaped method in stochastic programming literature, views the first stage variables as "complicating variables" in the sense that if the first stage variables $x$ are fixed the rest of problem (2.1) has a block-diagonal structure that can be decomposed by scenario and solved independently. A schematic view of the "complicating variable" idea behind Benders decomposition is shown in Figure 2.4. Mathematically, the Benders decomposition

algorithm starts by defining a master problem with only the first-stage decisions. After the master problem is solved, the first-stage decisions are fixed at the optimal solution of the master problem. Then the deterministic equivalent problem can be decomposed into $|\Omega|$ subproblems where $|\Omega|$ denotes the cardinality of the set of scenarios. The subproblems can be solved in parallel and valid inequalities of $x$ can be derived and added to the Benders master problem. The master problem is solved again and the algorithm iterates until the upper bound and the lower bound converge. The lower bound is the optimal value of the Benders master problem. The upper bound is obtained by updating the best feasible solution. Benders decomposition is able to converge in a finite number of steps when the second-stage decisions are all continuous and the second stage constraints are linear.

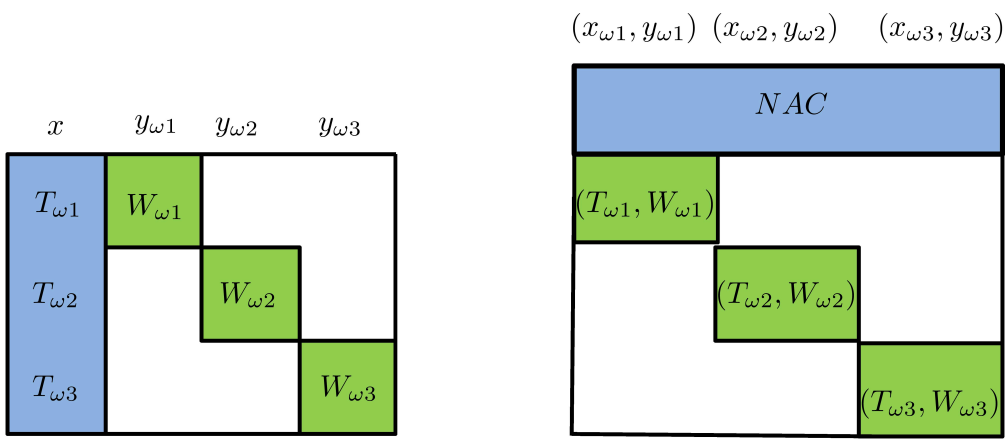


Figure 2.4: Schematic view of Benders decomposition (left; variables $x$ are "complicating variables"), Lagrangean decomposition (right; the NACs are "complicating constraints").

Lagrangean decomposition reformulates problem (2.1) by making a copy of the first-stage decisions for each scenario and adding non-anticipativity constraints ($NACs$) to ensure that the first-stage decisions made for all the scenarios are the same. For example, in a three-scenario problem, three copies of the $x$ variables, $x_{\omega_1}$, $x_{\omega_2}$, $x_{\omega_3}$, are made. NACs including $x_{\omega_1} = x_{\omega_2}$, $x_{\omega_1} = x_{\omega_3}$ are added to guarantee the first-stage decisions are the same for all three scenarios. A schematic view of Lagrangean decomposition is also shown in Figure 2.4. After this reformulation, the $NACs$ are then dualized so that the deterministic equivalent problem is decomposed into scenarios that can be solved in parallel. The Lagrangean multipliers can

be updated using the subgradient method or the cutting plane method (Oliveira et al., 2013). A lower bound can be obtained at each iteration of Lagrangean decomposition which is the summation of the optimal value of the Lagrangean subproblems. However, the upper bound procedure of Lagrangean decomposition is in general a heuristic. Therefore, Lagrangean decomposition is not guaranteed to converge even for the problems with continuous recourse. There is usually a "duality gap" between the upper and the lower bound.

Although neither Benders nor Lagrangean decomposition is able to solve (2.1) with integer recourse variables to optimality with their classical form, recent developments have been made that can solve (2.1) to optimality by extending these two methods. A discussion of these recent advances can be too technical and we refer interested readers to the review papers, Torres et al. (2019); Küçükyavuz and Sen (2017), for details.

As for software implementations, the CPLEX solver has an implementation of the Benders decomposition algorithm Bonami et al. (2020a), which can be accessed through most modeling platforms, such as GAMS, C, C++, Pyomo (Hart et al., 2017), Python, etc. DSP (Kim and Zavala, 2018) is a Lagrangean decomposition-based solver for two-stage stochastic mixed-integer programs, which provides interfaces that can read models expressed in C code and the SMPS format (Gassmann and Schweitzer, 2001). DSP can also read models expressed in JuMP (Lubin and Dunning, 2015).

### 2.1.2 Two-stage stochastic mixed-integer nonlinear programs

Most SP algorithms and applications investigated by the Operations Research (OR) community are (mixed-integer) linear problems as in Equation (2.1). However, chemical engineering applications can involve significant nonlinearities. Examples include the mass balance equations for the splitters and mixers in a flowsheet, the MESH equations in distillation column design, the Arrhenius equation in reactor modeling. Therefore, developing stochastic programming models and algorithms for (mixed-integer) nonlinear problems is of great interest to chemical engineers doing PSE research. As a matter of fact, the PSE

community has been the main driver of developing algorithms to efficiently solve stochastic MINLPs (Li et al., 2011c; Cao and Zavala, 2019; Li and Grossmann, 2019a).

#### 2.1.2.1 Mathematical formulation

The mathematical formulation of a general two-stage stochastic mixed-integer nonlinear program is shown in (2.4).

$$\begin{aligned}
\min \quad & f_0(x) + \sum_{\omega \in \Omega} \tau_\omega f_1(x, y_\omega; \theta_\omega) \\
\text{s.t.} \quad & g_0(x) \leq 0 \\
& g_1(x, y_\omega; \theta_\omega) \leq 0 \quad \forall \omega \in \Omega \\
& x \in X, \quad X = \left\{x : x_i \in \{0,1\}, \forall i \in I_1, \ 0 \leq x \leq x^{ub}\right\} \\
& y_\omega \in Y_\omega, \quad \forall \omega \in \Omega, \quad Y_\omega = \left\{y_\omega : y_{\omega j} \in \{0,1\}, \forall j \in J_1, \ 0 \leq y_\omega \leq y_\omega^{ub}\right\}
\end{aligned} \tag{2.4}$$

Problem (2.4) is an extension of (2.1) by considering nonlinear objective and nonlinear constraints. Functions $f_0$, $f_1$, $g_0$, $g_1$ in (2.4) are nonlinear functions. $f_1$ and $g_1$ are functions of the first-stage decisions $x$, the second-stage decisions $y_\omega$ and the uncertain parameters $\theta_\omega$ for scenario $\omega$. Due to the nonlinear objective and the nonlinear constraints, (2.4) is more difficult to solve than (2.1). We will review the algorithms to solve (2.4) in subsection 2.1.2.3.

#### 2.1.2.2 Stochastic pooling problem example

We provide an example of the stochastic pooling problem presented in Li and Grossmann (2019a). Pooling problem arises in applications include wastewater treatment (Karuppiah and Grossmann, 2008), crude oil refinery planning (Yang and Barton, 2016), and natural gas networks planning (Li et al., 2011b). Solving the pooling problem to optimality is of great interest due to potential savings in the order of tens of millions of dollars. In a pooling problem, flow streams from different sources are mixed in some intermediate tanks (pools) and blended again in the terminal points. At the pools and the terminals, the quality of a

mixture is given as the weighted average of the qualities of the flow streams that go into them.

The pooling problem involves both design decisions and operating decisions. The design decisions are which feed $i$ and which pool $l$ to select from the superstructure shown in Figure 2.5**(A)**, and the sizing decisions for the selected feed tanks and pools. The second-stage decisions are the mass flow rates of different streams, and the split fractions. The operator has to operate the feeds and the pools to satisfy the quality specifications at the product terminals $j$ (see Figure 2.5**(A)** ). In this problem, we also consider purchasing the feeds using three different types of contracts, i.e., fixed price, discount after a certain amount, and bulk discount.

Before the design decisions are made, several sources of uncertainties can arise at the operating level including quality of the feed streams, prices of the feeds and products, demands for the products. For this illustrative example, we only consider uncertainty in the demands of the products. The demands for the three products can take low, medium, and high values and are assumed to vary simultaneously, i.e., there are three scenarios in this problem with probabilities 0.3, 0.4, 0.3, respectively. The actual delivered products have to be less than or equal to the demands. The objective is to maximize the expected total profit. The mathematical formulation of this problem has been reported in Li and Grossmann (2019a). To make this chapter self-contained, we include the mathematical formulation in the supplementary material.

The deterministic equivalent of the stochastic pooling problem with three scenarios is solved to optimality. The optimal first-stage decisions are to select feeds $i_1$, $i_2$, $i_5$, pools $l_1$, and $l_4$. The optimal capacities of the selected feeds and pools are shown in Figure 2.5(**B**). Recall that in two-stage stochastic programming, after the first-stage decisions are made, the uncertain demands are realized. In different scenarios, different recourse actions can be taken depending on the realization of the demand. The mass flow rates of all the streams in the high and the low demand scenarios can be seen in Figure 2.5**(C)**. When the demand is low, the feed tanks and the pools are not operating at their full capacities. When the

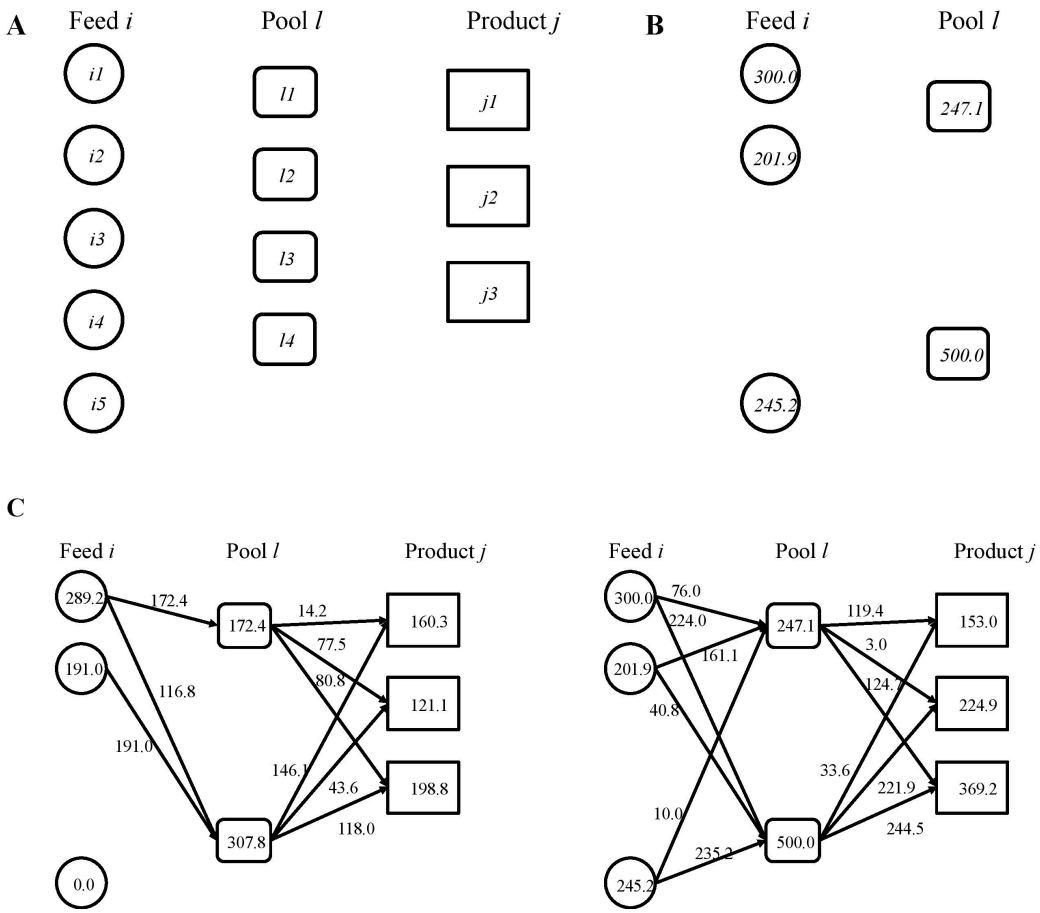


Figure 2.5: Stochastic pooling problem. **(A)** is the superstructure of the pooling problem. **(B)** is the optimal first-stage decisions for the pooling example. **(C)** represents the optimal mass flow rates in the low-demand scenario (left) and the high-demand scenario (right).

demand is high, the feed tanks and pools are operating at their full capacities. Recall that the production of each product can be less than or equal to the demand. In the high-demand scenario, the production of product $j_1$ is even lower than that in the low-demand scenario. However, the productions of $j_2$ and $j_3$ are higher in the high-demand scenario than those in the low-demand scenario. The results also indicate that the optimal capacity is not enough to meet the high demand. The reason is that if one were to fully satisfy the high demand, a great part of the capacity would be left idle in the low-demand scenario. On the other hand, if the capacity were designed to be just enough to satisfy the low demand, there would be significant lost sales in the high-demand and the medium-demand scenarios. Stochastic

programming is a "risk-neutral" approach to achieve the highest expected profit, which balances the trade-off between lost sales and idle capacity.

#### 2.1.2.3 Algorithms for solving stochastic MINLP

As we discussed in subsection 2.1.1.3, the classical decomposition algorithms, such as Benders decomposition and Lagrangean decomposition, cannot be readily applied to solve stochastic MINLPs. The challenges to solve stochastic MINLP problems are two-fold: the nonlinear functions in stage two can be nonconvex; second, some of the stage two variables need to satisfy integrality constraints, which also makes the stage two problem nonconvex. Due to its wide application in process systems, researchers from the PSE community have been developing algorithms for solving problem (2.4). Most of the algorithms are based on the classical decomposition algorithms including Lagrangean decomposition, and generalized Benders decomposition (GBD) (Geoffrion, 1972) which is an extension of the Benders decomposition (BD) algorithm to convex nonlinear problems.

For convex stochastic MINLP, where the nonlinear feasible region of the continuous relaxation is convex, Li and Grossmann (2018b) propose an improved L-shaped method where the Benders subproblems are convexified by rank-one lift-and-project, and Lagrangean cuts are added to tighten the Benders master problem. Li and Grossmann (2019b) further propose a generalized Benders decomposition-based branch and bound algorithm with finite $\epsilon$-convergence for convex stochastic MINLPs with mixed-binary first and second stage variables.

For nonconvex stochastic MINLP, where the nonlinear functions in the stochastic MINLPs can be nonconvex, the pioneering work is done by Li et al. (2011c) who propose a nonconvex generalized Benders decomposition algorithm, which can solve two-stage nonconvex MINLPs with pure binary variables in a finite number of iterations. For the more general case where the first stage variables can be mixed-integer, Ogbe and Li (2018) propose a joint decomposition algorithm. A perfect information-based branch and bound algorithm that solves non-

separable nonconvex stochastic MINLPs to global optimality is proposed by Cao and Zavala (2019). Kannan (2018) propose a modified Lagrangean relaxation-based (MLR) branch and bound algorithm, and they prove that MLR has finite $\epsilon$-convergence. A generalized Benders decomposition-based branch and cut algorithm for nonconvex stochastic MINLPs with mixed-binary first and second stage variables is proposed by Li and Grossmann (2019a). Li et al. (2020a) propose a sample average approximation based outer approximation algorithm for stochastic MINLPs with continuous probability distributions.

## 2.2 Multistage stochastic programming

In two-stage stochastic programming, it is assumed that the uncertainties are realized only once after the first-stage decisions are made. However, most practical problems involve a sequence of decisions reacting to outcomes that evolve over time. A generalization of two-stage stochastic programming to the sequential realization of uncertainties is called "multistage stochastic programming". The time horizon is discretized into "stages" where each stage has the realizations of uncertainties at the current stage.

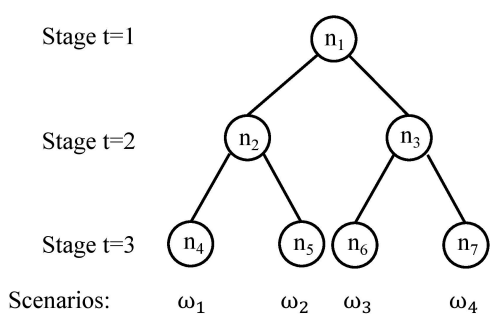


Figure 2.6: Illustration of a scenario tree with 3 stages, 4 scenarios, 7 nodes.

A scenario tree similar to Figure 2.1 for a multistage problem is shown in Figure 2.6. The scenario tree has 3 stages. In stage two, there are two different realizations of uncertain parameters and therefore two nodes. Each of the two nodes can have different stage two decisions depending on the realization of information until stage two. After the stage two decisions are made, there are two different realizations of uncertain parameters at stage

three for each node in stage two. Therefore, there are four nodes in total at stage three. The stage three decisions depend both on stage one, stage two decisions and the history of the uncertain parameters. A scenario in the multistage setting is a "path" from the root node of the scenario tree to the leaf node of the scenario tree, which corresponds to a full history of realization of uncertainty parameters until stage three. It is easy to see that there are 4 scenarios in Figure 2.6, represented using $\omega_1$, $\omega_2$, $\omega_3$, $\omega_4$. The 7 nodes in the scenario tree are numbered sequentially with notation $n_1$-$n_7$.

It is worth pointing out that in some PSE applications, there are two types of decisions, i.e., the strategic decisions, e.g., the installation of chemical processes and the operational decisions, e.g., material flow rates. The uncertainties can arise from both strategic and operational decision-making processes. A scenario tree that combines the two types of decisions and uncertainties can be found in Escudero and Monge (2018).

### 2.2.1 Mathematical formulation

The general multistage stochastic programming formulation is given as follows (Birge and Louveaux, 2011):

$$\min \quad c_1^\top x_1 + \mathbb{E}_{\boldsymbol{\xi}_{[\mathbf{2},\boldsymbol{H}]}|\xi_{[1,1]}}[\min c_2^\top(\xi_2)x_2(\xi_2) + ... + \mathbb{E}_{\boldsymbol{\xi}_{[\boldsymbol{H}:\boldsymbol{H}]}|\xi_{[1:H-1]}}[\min c_H^\top(\xi_H)x_H(\xi_H)]] \tag{2.5a}$$

$$\text{s.t.} \quad W_1 x_1 \leq h_1 \tag{2.5b}$$

$$T_2(\xi_2)x_1 + W_2 x_2(\xi_2) \leq h_2(\xi_2) \tag{2.5c}$$

$$... \tag{2.5d}$$

$$T_H(\xi_H)x_{H-1}(\xi_{H-1}) + W_H x_H(\xi_H) \leq h_H(\xi_H) \tag{2.5e}$$

$$x_1 \in X_1; x_t(\xi_t) \in X_t, t = 2, ..., H; \tag{2.5f}$$

where we have $H$ stages. For simplicity, we assume all the constraints and the objective function are linear. Suppose the data $(\boldsymbol{\xi_2}, \ldots, \boldsymbol{\xi_H})$ is uncertain and evolves according to a known stochastic process. We use $\boldsymbol{\xi_t}$ to denote the random data vector in stage t and $\xi_t$ to denote a specific realization. The parameters $c_t$, $T_t$, $W_t$, $h_t$ are functions of $\xi_t$. Similarly, we use $\boldsymbol{\xi}_{[\boldsymbol{t},\boldsymbol{t'}]}$ to denote the sequence of random data vectors corresponding to stages $t$ through

$t'$ and $\xi_{[t,t']}$ to denote a specific realization of this sequence of random vectors. The decision dynamics is as follows: in stage $t$, we first observe the data realization $\xi_t$ and then take an action $x_t$ depending on the previous stage decision $x_{t-1}$ and the observed data $\xi_t$ to optimize the expected future cost. $\mathbb{E}_{\boldsymbol{\xi_{[t,H]}}|\boldsymbol{\xi_{[1,t-1]}}}$ denotes the expectation operation in stage $t$ with respect to the conditional distribution of $\boldsymbol{\xi_{[t,H]}}$ given realization $\xi_{[1,t-1]}$ in stage $t-1$. The constraints at stage $t$ are defined for all the possible realizations of the stochastic process $(\boldsymbol{\xi_2}, \ldots, \boldsymbol{\xi_t})$ until stage $t$. Set $X_t$ denotes the domain of variables $x_t$, which can be continuous or (mixed)-integer. Equation (2.5) can represent the deterministic equivalent of any multi-stage mixed-integer linear stochastic programs with discrete or continuous distributions.

For problems with discrete distributions, we can use the following node-based formulation,

$$\min_{x_n} \Big\{ \sum_{n \in \mathcal{N}} \tau_n c_n^\top x_n : \quad T_n x_{a(n)} + W_n x_n \leq h_n, x_n \in X_n, \forall n \in \mathcal{N} \Big\} \tag{2.6}$$

where set $\mathcal{N}$ denotes the set of nodes in the scenario tree. The decisions at node $n$ is denoted as $x_n$. We use $a(n)$ to denote the ancestor nodes of node $n$. For example, in the scenario tree shown in Figure 2.6, the ancestor nodes of $n_4$ are $n_2$ and $n_1$, i.e., $a(n_4) = \{n_2, n_1\}$. $\tau_n$ is the probability of node $n$. The probability of a non-leaf node equals to the summation of the probabilities of its child nodes. For example, in Figure 2.6, $\tau_{n_2} = \tau_{n_4} + \tau_{n_5}$. The decisions at a given node $n$, $x_n$, is dependent on the decision made at its ancestor nodes $x_{a(n)}$ and the realizations of the uncertain parameters until node $n$, which is reflected in $T_n$, $W_n$, and $h_n$.

An alternative way to write the deterministic equivalent of the node-based formulation is the following recursive formulations where set $\mathcal{C}(n)$ denotes all the child nodes of a node $n$. Parameter $q_{nm}$ denotes the transition probability from node $n$ to its child node $m$. For example, in Figure 2.6, if nodes $n_4$, $n_5$ have equal probability, the transition probabilities from node $n_2$ to $n_4$ and $n_5$ are both 0.5.

$$\min_{x \in X_1} \left\{ c_1^\top x_1 + \sum_{m \in \mathcal{C}(1)} q_{1m} Q_m(x_1) : W_1 x_1 \leq h_1 \right\} \tag{2.7}$$

where for each node $n \in \mathcal{N}\backslash\{1\}$, the cost-to-go function $Q_n$ is defined as

$$Q_n(x_{a(n)}) = \left\{ \min_{x_n \in X_n} \; c_n^\top x_n + \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(x_n) : W_n x_n \leq h_n - T_n x_{a(n)} \right\} \tag{2.8}$$

For the terminal condition $t = H$, the set $\mathcal{C}(n)$ is be empty.

### 2.2.2 Multistage process network problem

To provide an illustrative example for multistage stochastic programming, the two-stage process network design problem in subsection 2.1.1.2 is extended to a three-stage problem. Stage one decisions are the same as those in 2.1.1.2, i.e., the selection of the three processes and the capacities of the selected processes. After the design decisions at stage one are made, the demand in stage two are realized. Besides the operating decisions on the installed plant, the decisions at stage two include capacity expansion decisions on the installed processes. We assume that the expanded capacity will be available at stage three. After the stage two decisions are made, the demand at stage three is realized. The stage three decisions only include the operating decisions, i.e., the material flow rates. The objective is to maximize the expected total profit over all the scenarios.

The scenario tree is assumed to have two realizations per stage, which has the same structure as the scenario tree in Figure 2.6. We use $d_n$ to denote the realization of demand at node $n$. The demand at stage two is assumed be take two different values, $d_{n_2} = 8$, $d_{n_3} = 12$. The two child nodes, $n_4$ and $n_5$, of node $n_2$, are assumed to take values $d_{n_5} = 8$, $d_{n_6} = 10$, whereas, the two child nodes, $n_6$ and $n_7$, of node $n_3$, are assumed to take higher values, i.e., $d_{n_6} = 12$, $d_{n_7} = 14$. The four scenarios are assumed to have equal probabilities.

The optimal solutions of multistage stochastic process network design problem corresponding to nodes $n_2, \ldots, n_7$ are shown in Figure 2.7. The material flow rates at each node are shown in red. The installed capacities at the previous stage are shown in blue italics. At stage one, process 1 and 3 are selected and can be used in stage two. At stage two, when the demand is low, i.e., $d_{n_2} = 8$, processes 1 and 3 are not operating at full capacity. When the demand is high i.e., $d_{n_3} = 12$, both process 1 and process 3 are operating at full capac-

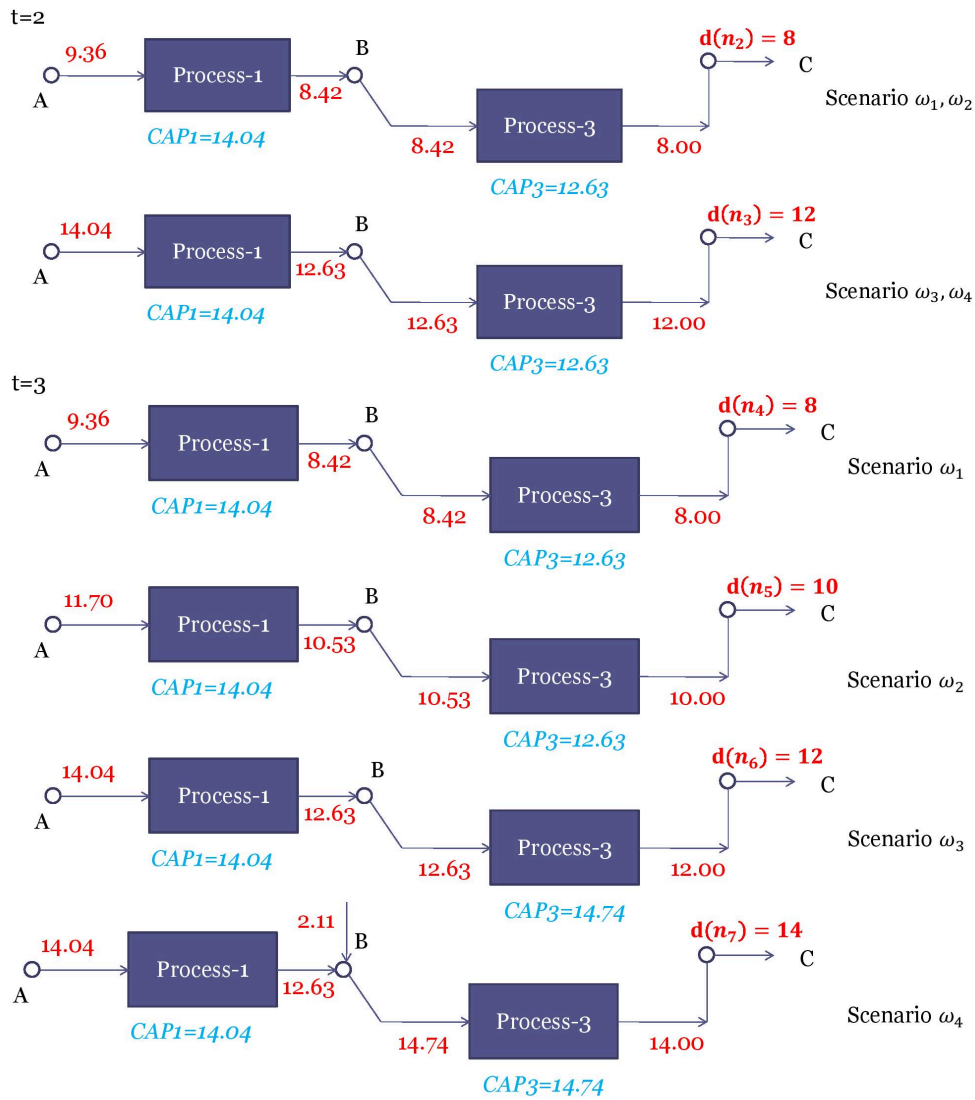


Figure 2.7: Optimal solutions of the stochastic process network design problem corresponding to nodes $n_2, \ldots, n_7$. The material flow rates at each node are shown in red. The installed capacities are shown in blue italics.

ity. Process 3 is expanded at stage two when the high demand is observed. The expanded capacity can be used in stage three at nodes $n_6$ and $n_7$. The stage three decisions are the material flow rates of different chemicals. At node $n_4$, $n_5$, and $n_6$, the installed process 1 and 3 are able to satisfy the demand. At node $n_7$, due to the limited capacity of process 1, additional chemical B needs to be purchased.

### 2.2.3 Algorithms for multistage stochastic (mixed-integer) linear programs

The number of scenarios of a multistage stochastic program grows exponentially with the number of stages. For example, suppose we have 3 different realizations of uncertain parameters at each stage, the number of scenarios for a problem with $H$ stages is $3^{H-1}$. Therefore, special decomposition algorithms are usually applied to solve multistage stochastic programming problems. Most of the algorithms for multistage stochastic programs are based on classical Benders and Lagrangean decomposition algorithms.

An extension of Benders decomposition to multistage stochastic linear programs is called "nested Benders decomposition" (Birge, 1985b). The idea is to apply Benders decomposition recursively to the multistage problem. Nested Benders decomposition has a "forward pass" step and a "backward pass" step. In the forward pass step, feasible solutions are generated by sequentially solving the root node of the scenario tree all the way to the leaf nodes of the scenario tree. Problem (2.8) for node $n$ is solved "locally" in the forward pass with the value of $x_{a(n)}$ fixed from the solution of its parent node and the expected cost to go function $Q_m(x_n)$ approximated by some cutting planes generated from the backward pass. In the backward pass, the problems are solved sequentially from the leaf nodes to the root node. Cutting planes are generated in the backward pass to refine the approximations for the cost-to-go functions $Q_m(x_n)$.

Although the nested Benders decomposition algorithm can decompose the fullspace problem by nodes, it still has to solve an exponential number of leaf nodes. An algorithm that avoids solving an exponential number of nodes is the stochastic dual dynamical programming (SDDP) algorithm (Pereira and Pinto, 1991). One caveat to apply SDDP is that the underlying stochastic process has to be stagewise-independent, i.e., the realizations of $\boldsymbol{\xi_t}$ do not dependent on the history $\xi_{[1,t-1]}$. The idea of SDDP is that the cutting plane can be shared among the nodes under the stagewise-independence assumption and thus avoiding solving an exponential number of nodes. An extension of SDDP that can solve multistage stochastic mixed-integer linear program is the stochastic dynamic dual integer programming

(SDDiP) algorithm proposed by Zou et al. (2019). Besides the standard Benders cuts, the authors propose to use other families of cutting planes including Lagrangean cuts, integer cuts, and strengthened Benders cuts. Applications of SDDiP have been reported in Escudero et al. (2020); Zou et al. (2018b); Lara et al. (2019).

Lagrangean decomposition can be adapted to solve multistage stochastic programs in a similar way to two-stage by duplicating the variables such that each scenario has its own copy of variables at each stage and adding nonanticipativity constraints (NACs). The scenario tree in Figure 2.6 can be expressed in an alternative form shown in Figure 2.8 (Ruszczyński, 1997). The node in the first stage in Figure 2.6 is duplicated for each scenario $\omega$ and NACs (shown in red in Figure 2.8) are added to guarantee that the first stages decisions are the same for all the scenarios. Each node in stage two in Figure 2.6 is duplicated once and NACs are added to guarantee that the scenarios with the same parent node in Figure 2.6 have the same second-stage decisions. With the alternative scenario tree, Lagrangean decomposition dualizes the NACs and solves each scenario independently.

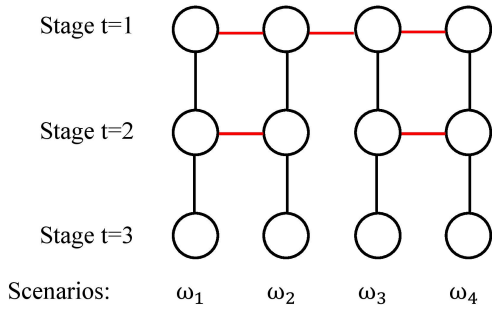


Figure 2.8: An alternative form of the scenario tree with 3 stages, 4 scenarios

One major difference between nested Benders decomposition and Lagrangean decomposition is that nested Benders decomposition decomposes the fullspace problem by nodes, whereas Lagrangean decomposition decompose the fullspace problem by scenarios. A discussion of the trade-offs are not within the scope of this thesis and can be found in Torres et al. (2019). Moreover, a recent review on main types of decomposition algorithms for two-stage and multistage stochastic mixed integer linear optimization can be found in Escudero et al. (2017).

As for software packages for multi-stage stochastic programs, SDDP.jl (Dowson and Kapelevich, 2020) is an implementation of the SDDP algorithm in Julia that is built on the JuMP modeling tools (Lubin and Dunning, 2015). MSPPy (Ding et al., 2019) is an implementation of the SDDP/SDDiP algorithm in Python. PySP (Watson et al., 2012) is an implementation of the progressive hedging algorithm (Rockafellar and Wets, 1991), an algorithm that can be seen as an augmented Lagrangean decomposition, in Pyomo (Hart et al., 2017).

## 2.3 Multistage stochastic programming under endogenous uncertainty

In the stochastic programming models that we have discussed in the last few sections, the underlying stochastic processes are *independent* of the decisions. This type of uncertainty is called "exogenous" uncertainty. Another type of uncertainty that is decision-*dependent* is called "endogenous" uncertainties. There two distinct types of endogenous uncertainties commonly denoted as Type 1 and Type 2 in the literature (Goel and Grossmann, 2006).

In the case of Type 1 endogenous uncertainties, decisions influence the parameter realizations by altering the underlying probability distributions for the uncertain parameters. A simple example of this may be an oil company's decision to flood the market in order to force a competitor out of business. Here the uncertainty is no longer strictly exogenous, as the decision will make lower oil-price realizations more probable. Type 1 endogenous has not been widely studied.

In the case of Type 2 endogenous uncertainties, decisions influence the parameter realizations by affecting the time at which we observe these realizations. This refers specifically to technical parameters, such as oilfield size, for which the true values cannot be determined until a particular investment decision is made. The modeling framework and algorithm for stochastic programming with Type 2 endogenous uncertainty are first proposed by Goel and Grossmann (2006). Since then, this approach has been adopted by the PSE community

to address problems in a wide range of applications including oil and gas field planning (Goel and Grossmann, 2004), synthesis of process networks (Tarhan and Grossmann, 2008), pharmaceutical industry (Colvin and Maravelias, 2008; Christian and Cremaschi, 2015), etc.

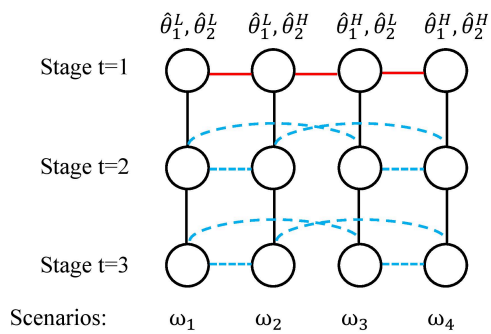


Figure 2.9: Illustration of the scenario tree of a stochastic programming problem under endogenous uncertainty with 3 stages, 4 scenarios, 2 uncertain parameters $\theta_1$, $\theta_2$.

Here, we describe the modeling framework proposed by Goel and Grossmann (2006). Figure 2.9 shows a scenario tree for a three-stage problem with two endogenous uncertain parameters, $\theta_1$, $\theta_2$. Both parameters are assumed to have two realizations, i.e., $\theta_1 \in \{\hat{\theta}_1^L, \hat{\theta}_1^H\}$, $\theta_2 \in \{\hat{\theta}_2^L, \hat{\theta}_2^H\}$. We consider the combinations of the two realizations for $\theta_1$ and $\theta_2$. There are four scenarios in total with realizations, $\{\hat{\theta}_1^L, \hat{\theta}_2^L\}$, $\{\hat{\theta}_1^L, \hat{\theta}_2^H\}$, $\{\hat{\theta}_1^H, \hat{\theta}_2^L\}$, $\{\hat{\theta}_1^H, \hat{\theta}_2^H\}$, respectively. The scenario tree in Figure 2.9 is a scenario-based representation, i.e., the decisions are duplicated for each scenario and NACs are added. However, the way that the NACs are added in Figure 2.9 is different than that in Figure 2.8 because the realizations of the uncertain parameters are decision-dependent. In stage one, since the decisions have not been made to reveal the values of the uncertain parameters, "initial NACs" are added to guarantee that the first-stage decisions are the same for all the scenarios. The initial NACs are represented using solid red lines in Figure 2.9. For stage two and three, the NACs are added conditionally, depending on whether the corresponding decisions are made in order to reveal the uncertain parameters. These NACs are called "conditional NACs", which are shown in Figure 2.9 as dashed blue lines.

To provide a concrete example, suppose $\theta_1$ and $\theta_2$ are the sizes of two undeveloped oil fields, $o_1$ and $o_2$, respectively, the decisions to reveal the sizes are the drilling of oil field 1

and 2. We use binary variable $b_{ot}$ to denote whether oil field $o$ is drilled at stage $t$. At $t = 1$, since no oil fields have been drilled before $t = 1$ and no information regarding the size of the oil fields has been obtained, the decisions for all the four scenarios are the same, i.e., we need to add the initial NACs. At $t = 1$, if the operator decides to drill $o_1$, then the realization of $\theta_1$ will be known before stage 2. At stage two, the scenarios with $\theta_1 = \hat{\theta}_1^L$ and the scenarios with $\theta_1 = \hat{\theta}_1^H$ can be distinguished. Conditional NACs will only be added for the scenario pairs $(\omega_1, \omega_2)$, and $(\omega_3, \omega_4)$. On the other hand, if the operator decides not to drill any oil fields at $t = 1$, the four scenarios are still indistinguishable at the beginning of stage two, all the conditional NACs at stage two have to be enforced.

### 2.3.1 Mathematical formulation

The mathematical formulation for the multistage stochastic programming problem under endogenous uncertainty is,

$$\min \sum_{\omega \in \Omega} \tau_\omega \sum_{t \in \mathcal{T}} c_{t,\omega}^\top x_{t,\omega} \tag{2.9a}$$

$$\text{s.t.} \quad \sum_{t'=1}^{t} A_{t',t,\omega} x_{t',\omega} \leq b_{t,\omega} \quad \forall t \in \mathcal{T}, \omega \in \Omega \tag{2.9b}$$

$$x_{1,\omega} = x_{1,\omega'}, \quad \forall (\omega, \omega') \in \mathcal{SP}_{\mathcal{F}} \tag{2.9c}$$

$$\begin{bmatrix} Y_{t,\omega,\omega'} \\ x_{t+1,\omega} = x_{t+1,\omega'} \end{bmatrix} \vee \left[\neg Y_{t,\omega,\omega'}\right] \quad \forall (\omega, \omega') \in \mathcal{SP}_{\mathcal{N}}, t \in \mathcal{T} \tag{2.9d}$$

$$Y_{t,\omega,\omega'} \Leftrightarrow F(x_{1,\omega}, \ldots, x_{t,\omega}) \quad \forall (\omega, \omega') \in \mathcal{SP}_{\mathcal{N}}, t \in \mathcal{T} \tag{2.9e}$$

$$x_{t,\omega} \in X_t, \quad \forall t \in \mathcal{T}, \omega \in \Omega \tag{2.9f}$$

$$Y_{t,\omega,\omega'} \in \{True, False\} \quad \forall (\omega, \omega') \in \mathcal{SP}_{\mathcal{N}}, t \in \mathcal{T} \tag{2.9g}$$

where $x_{t,\omega}$ denotes the decisions made at stage $t$ in scenario $\omega$. $Y_{t,\omega,\omega'}$ is a boolean variable that equals to $True$ if the two scenarios $\omega$ and $\omega'$ are indistinguishable at stage $t$. The objective function (2.9a) is to minimize the expected cost. Equation (2.9b) represents constraints that

govern decisions $x_{t,\omega}$ and link decisions across time periods. The initial NACs are expressed in equation (2.9c), where set $\mathcal{SP}_{\mathcal{F}}$ represents the set of scenario pairs to enforce the initial NACs. The scenario pairs for the initial NACs can be written similar to the NACs in the exogenous case. Equation (2.9d) represents the conditional NACs where set $\mathcal{SP}_{\mathcal{N}}$ represents the scenario pairs that differ in the realization of only one endogenous parameter (Goel and Grossmann, 2004). If the two scenarios $\omega$ and $\omega'$ are indistinguishable at stage $t$, i.e., $Y_{t,\omega,\omega'} = True$, NACs are added for the decisions in the next time period. Otherwise, no NACs are added. The value of the boolean variable is determined by the decisions made at and prior to time $t$, which is expressed in a general form in equation (2.9e). In the oil field example, the expression $F(x_{1,\omega}, \ldots, x_{t,\omega})$ is to determine whether a given oil field has been developed based on the drilling decisions. Equations (2.9f) and (2.9g) specify the domain of variables $x_{t,\omega}$ and $Y_{t,\omega,\omega'}$.

We acknowledge that the mathematical formulation in (2.9) is a succinct form for ease of explanation. For a more detailed and rigorous formulation, we refer the readers to the paper by Apap and Grossmann (2017). See also Hellemo et al. (2018) for a nonlinear modelling approach.

### 2.3.2 Multistage process network design under endogenous uncertainty

To provide an illustrative example for multistage stochastic programming under endogenous uncertainty, the two-stage process network design problem in subsection 2.1.1.2 is modified to a three-stage problem where the uncertainty is assumed to come from the yield of process 1 and the demand is assumed to be deterministic. In practice, process 1 can be a new technology that has never been installed before. Once process 1 is installed, its yield will be realized in the next stage. Therefore, the time of realization is decision-dependent. We assume that there are two realizations of the yield of process 1, i.e., two scenarios, being 0.9, and 0.8 with equal probability, which is denoted as $\text{Yield}(\omega_1) = 0.9$, $\text{Yield}(\omega_2) = 0.8$, $\tau(\omega_1) = 0.5$, $\tau(\omega_2) = 0.5$

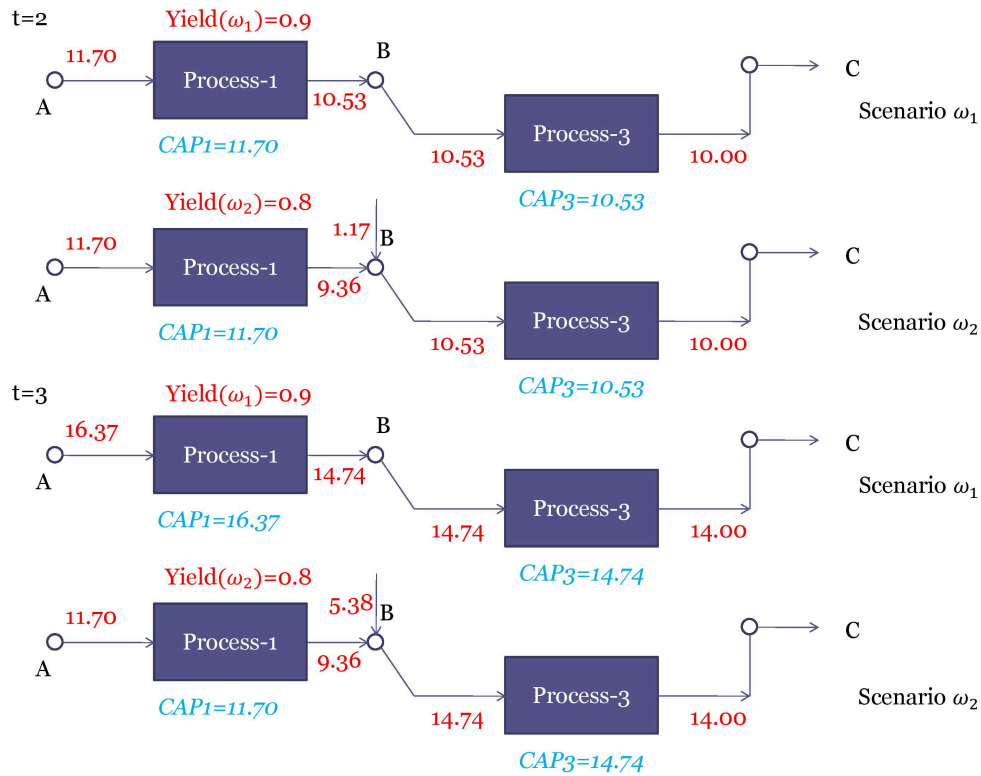


Figure 2.10: Optimal solution of the three-stage process network problem under endogenous uncertainty.

For this problem, stage one decisions are the same as those in subsection 2.1.1.2, i.e., the selection of the three processes and the capacities of the selected processes. At stage two, besides the operating decisions on the installed plant, the capacity of the processes that have been installed at stage one can be expanded; the processes that have not been installed at stage one can also be selected and installed at stage two. The installed and expanded processes will be available at stage three. The stage three decisions only include the operating decisions, i.e., the material flow rates. The objective is to maximize the expected total profit over all the scenarios.

Three cases can happen for the realization of the yield uncertainty:

1. Process 1 is selected at stage one. The uncertainty of the yield is realized at stage two. Only the initial NACs are active.

2. Process 1 is selected at stage two. The uncertainty of the yield is realized at stage three. The initial NACs and the NACs for stage two decisions are active.

3. Process 1 is never selected. The uncertainty of the yield is never realized. The initial NACs and the NACs for stage two and stage three decisions are active.

The optimal solution is shown in Figure 2.10. Processes 1 and 3 are selected at stage one. The yield of process 1 is realized at stage two. The capacity of process 1 installed at stage one is chosen such that it is only able to produce enough chemical B if the yield is high. On the other hand, if the yield of process 1 is low, additional chemical B needs to be purchased at stage two. Since the two scenarios can be distinguished at stage two, additional capacity for process 1 is added in the high-yield scenario such that enough chemical B can be produced by process 1 when the demand is increased to 14 at stage three. For the low-yield scenario, it is not profitable to produce chemical B from process 1. Therefore, no capacity for process 1 is added in stage two. The inadequacy of chemical B is resolved by purchasing from other suppliers.

In order to see the value of the stochastic solution, we solve a deterministic problem with the yield fixed at mean value, i.e., 0.85. The optimal solution of the deterministic model is to select processes 1 and 3 as well. However, the capacity of process 1 at stage one is 12.38 instead of 11.70 as in the stochastic model. This capacity is chosen such that enough chemical B can be produced at stage two when the yield is 0.85. This capacity is suboptimal if the actual yield can be 0.8 or 0.9.

### 2.3.3 Algorithms for solving stochastic programming under endogenous uncertainty

Solving stochastic programs under endogenous uncertainties can be even more challenging than solving the stochastic programs under exogenous uncertainties because of the additional binary variables involved in reformulating the disjunctions in the conditional NACs (2.9d). To reduce the number of redundant NACs, scenario pair reduction techniques have been

proposed Goel and Grossmann (2006); Boland et al. (2016); Apap and Grossmann (2017). However, even with a reduced number of scenario pairs, the deterministic equivalent (2.9) of problems with a large number of scenarios is still intractable. Goel and Grossmann (2006) propose a Lagrangean decomposition-based branch and bound algorithm that dualizes the initial NACs and relax the conditional NACs. To satisfy the NACs, the authors propose to branch on the logical variables $Y_{t,\omega,\omega'}$, and the variables involved in the constraints that are dualized. The Lagrangean decomposition-based algorithm is able to provide both feasible solutions and a lower bound for the stochastic programs. Colvin and Maravelias (2008) propose a branch and cut algorithm where necessary non-anticipativity constraints that are unlikely to be active are removed from the initial formulation and only added if they are violated within the search tree.

Others have proposed heuristic algorithms that can quickly obtain feasible solutions to (2.9). Christian and Cremaschi (2015) apply a knapsack decomposition algorithm (KDA) to the pharmaceutical R&D pipeline management problem. Zeng et al. (2018) extend the KDA algorithm to a generalized knapsack-problem based decomposition algorithm (GKDA). Apap and Grossmann (2017) propose a sequential decomposition algorithm for stochastic programs under both exogenous and endogenous uncertainties.

## 2.4 Data driven scenario tree generation

The crucial assumption in stochastic programming is that a scenario tree is given to characterize the probability distribution of the underlying stochastic process. The values and probabilities in the scenario tree affect the optimal solutions obtained from the stochastic programs. A poor scenario-generation method can spoil the result of the whole optimization model. In this section, we review the concepts and algorithms of scenario tree generation. Note that the notations used here are for exogenous uncertainty. The concepts and methods can be easily extended to problems with endogenous uncertainty.

The sources of data for generating scenarios can come from historical data, time-series model, or expert knowledge (Kaut, 2011). Historical data represents reliable past infor-

mation but can generalize poorly in the future. Statistical or time-series models such as autoregressive integrated moving average (ARIMA), hidden Markov model (HMM), might generalize well in the future but usually need to be fit by data. Expert knowledge is useful in certain applications. For example, the prediction of oil price needs expertise in geopolitics. In practice, usually, a combination of all the three approaches is used, i.e., estimate the distribution from historical data, then use a mathematical model and/or expert knowledge to adjust the distribution to the current situation.

A good scenario tree should capture the distributions of the random variables at each time period and the inter-temporal dependencies. The distributions at each time period can be characterized by the marginal distributions of all the variables, or more succinctly by their means and higher-order moments. Furthermore, the dependence of the variables is usually measured by their correlations. The inter-temporal dependencies describe how the realizations in the previous time periods affect the distributions at the current period. These dependencies are typically modeled by time-series models.

The time series models are inherently continuous probability distributions of the random variables. A scenario tree can be seen as a discrete approximation of a time-series model. The quality of a scenario tree generation method can be evaluated by the stability and the error of this approximation. The stability measures the deviation of the optimal solutions if the generation method is applied multiple times to generate the scenario trees. The error measures the suboptimality in the objective value resulted from the discrete approximation. The quantitative descriptions of the stability and error in scenario tree generation can be found in Kaut and Wallace (2003).

### 2.4.1 Sampling-based scenario generation methods

A common way to generate scenario trees is to generate i.i.d. (independent and identically distributed) samples from the "true" distribution through a simulation. For example, in the case of two-stage stochastic programming, we represent "true" stochastic programming

problem succinctly as,

$$z^* = \min_x \mathop{\mathbb{E}}_{\theta\sim\mathbb{P}} f(x;\theta) \tag{2.10}$$

where $x$ is the first stage variables, $\mathbb{P}$ denotes the "true" distribution, $\theta$ denotes the random variables. The stage two problem is contained implicitly in function $f$. Solving (2.10) with a continuous distribution directly is usually computationally intractable due to the integration over the continuous distribution. Suppose we can generate $N$ i.i.d. samples, $\theta_1, \ldots, \theta_N$ from $\mathbb{P}$, (2.10) can be approximated by the following sample average approximation (SAA) problem (Shapiro et al., 2014),

$$z^N = \min_x \frac{1}{N} \sum_{i=1}^{N} f(x;\theta_i) \tag{2.11}$$

where the true distribution $\mathbb{P}$ is approximated by the empirical distribution of the $N$ i.i.d. samples. It can be proved that as $N \to \infty$, the optimal value and solutions of (2.11) converge to the optimal value and solutions of (2.10) under some mild assumptions (Shapiro et al., 2014). However, generating an infinite number of samples is impractical. Sample size estimates for finite error have been developed (Shapiro et al., 2014; Kleywegt et al., 2002) for different types of two-stage stochastic programs. These estimates are usually too conservative for real-world problems. A practical implementation of the sample average approximation is proposed by Kleywegt et al. (2002).

There are several advantages to the SAA method. It is easy to implement if it is possible to sample from the true distribution. It also has very good asymptotic and finite convergence properties. However, SAA can have poor performance and stability if a small number of samples are used, especially for probability distributions with large variances. More importantly, we have to know the "true" distribution to sample from.

### 2.4.2 Property matching methods

The idea behind the property matching methods is to construct the scenario trees in such a way that some given properties of the "true distribution" are matched. The properties are, for example, the moments of the marginal distributions and covariances/correlations.

Høyland and Wallace (2001) propose a nonlinear programming approach to matching the moments of the scenario tree to those estimated from a dataset or a continuous distribution. The unknowns in the nonlinear programming problem are the probabilities and values of the scenarios with the number of scenarios fixed.

However, with the moments alone, the matching problem is usually under-specified, especially if the number of realizations in the scenario tree is large, i.e., the scenario tree is able to match the moments perfectly and there are still some degrees of freedom left. To address the under-specification issue, Calfa et al. (2014) propose to match both the moments and the cumulative distribution function (CDF) of the empirical distribution that are derived from real-world data. For example, in Figure 2.11, the blue dots represent the empirical CDF constructed from some production yield data. Since the empirical distribution is discontinuous and difficult to match in an optimization problem, Calfa et al. (2014) propose to fit a Generalized Logistic Function (GLF) with the data points of the empirical distribution.

$$GLF(x) = \beta_0 + \frac{\beta_1 - \beta_0}{\left(1 + \beta_2 e^{-\beta_3 x}\right)^{1/\beta_4}} \tag{2.12}$$

When fitting the GLF to the empirical CDF data, the GLF can be simplified by setting $\beta_0 = 0$ and $\beta_1 = 1$ as these parameters correspond to the lower and upper asymptotes, respectively. The empirical CDF in Figure 2.11 can be approximated by the red GLF curve with $\beta_2 = 9.5587 \times 10^7$, $\beta_3 = 22.2792$, $\beta_4 = 2.3810$. The NLP formulation that aims to

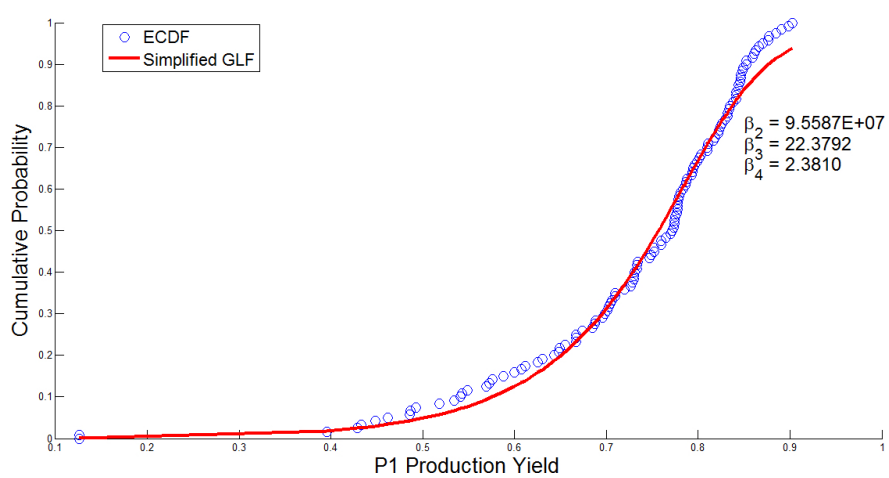


Figure 2.11: Distribution of the historical data for the production yield of facility P1 (Calfa et al., 2014).

match the moments, the covariances, and the distribution calculated from data is,

$$\min_{x,\,p} \quad z^{L^2}_{\mathrm{DMP}} = \sum_{i\in I}\sum_{k\in K} \mathrm{w}_{i,k}(m_{i,k} - \mathrm{M}_{i,k})^2 + \sum_{\substack{(i,\,i')\in I\\ i<i'}} \mathrm{w}_{i,i'}(c_{i,i'} - \mathrm{C}_{i,i'})^2 + \sum_{i\in I}\sum_{j=1}^{\mathrm{N}} \omega_{i,j}\delta^2_{i,j} \tag{2.13a}$$

$$\text{s.t.} \sum_{j=1}^{\mathrm{N}} p_j = 1 \tag{2.13b}$$

$$m_{i,1} = \sum_{j=1}^{\mathrm{N}} x_{i,j}p_j \qquad \forall\, i \in I \tag{2.13c}$$

$$m_{i,k} = \sum_{j=1}^{\mathrm{N}} (x_{i,j} - m_{i,1})^k p_j \qquad \forall\, i \in I,\ k > 1 \tag{2.13d}$$

$$c_{i,i'} = \sum_{j=1}^{\mathrm{N}} (x_{i,j} - m_{i,1})(x_{i',j} - m_{i',1})p_j \qquad \forall\, (i,\ i') \in I,\ i < i' \tag{2.13e}$$

$$x_{i,j} \in [\mathrm{x}^{\mathrm{LB}}_{i,j},\ \mathrm{x}^{\mathrm{UB}}_{i,j}] \qquad \forall\, i \in I,\ j = 1,\ \ldots,\ \mathrm{N} \tag{2.13f}$$

$$p_j \in [0,\ 1] \qquad \forall\, j = 1,\ \ldots,\ \mathrm{N} \tag{2.13g}$$

$$\widehat{ECDF}(x_{i,j}) - \sum_{j'=1}^{j} p_{j'} = \delta_{i,j} \qquad \forall\, i \in I,\ j = 1,\ \ldots,\ \mathrm{N} \tag{2.13h}$$

$$x_{i,j} \le x_{i,j+1} \qquad \forall\, i \in I,\ j = 1,\ \ldots,\ \mathrm{N}-1 \tag{2.13i}$$

where the entries of the uncertain parameters are indexed by $i \in I$. For example, if the uncertainties are the yields of some processes, $I$ will be the set of these processes. N denotes the number of outcomes per node at the second stage, $j \in J = \{1,\ 2,\ \ldots,\ \mathrm{N}\}$ denotes the branches (outcomes) from the root node, and $k \in K = \{1,\ 2,\ 3,\ 4\}$ is the index of the first four moments. The decision variables are the uncertain parameters of the stochastic programming problem, $x_{i,j}$, and their corresponding probabilities, $p_j$. The moments calculated from the tree are denoted by variables $m_{i,k}$ and the ones calculated from the data are denoted by parameters $\mathrm{M}_{i,k}$. The covariances calculated between entity $i$ and $i'$ from the tree and the data are denoted by $c_{i,i'}$ and $\mathrm{C}_{i,i'}$, respectively. Variables $\delta_{i,j}$ represent the deviations with respect to the empirical CDF data, which in turn are approximated by, for example, the GLF and is represented by the expression $\widehat{ECDF}(x_{i,j})$. In addition to minimizing the weighted square errors from matching (co-)moments, the sum of squares of the deviations $\delta_{i,j}$ is also minimized with given weights $\omega_{i,j}$ that can be chosen relative to the weights for

the term involving the moments.

Problem (2.13) minimizes the weighted Euclidean distance of the moments, covariances, and distribution from the scenario tree to those calculated from the data. Problem (2.13) is a nonconvex NLP because the variables $x$, $m$, and $p$ are involved in the bilinear and trilinear terms in Equations (2.13d) and (2.13e). If the empirical CDF is not considered, problem (2.13) reduces to the standard moment matching NLP problem proposed by Høyland and Wallace (2001).

## 2.5 Conclusions

We have provided an overview of stochastic programming in process systems engineering. The applications of stochastic programming are widespread, from the traditional petrochemical, pharmaceutical industry to carbon capture, energy storage. There are two major types of uncertainties that can be modeled using stochastic programming, exogenous uncertainty, which is the most commonly one considered, and endogenous uncertainty where realizations of the uncertainties depend on the decisions taken. Depending on the time discretization and the realizations of uncertainties, two-stage or multistage stochastic programming can be used to solve the problem under exogenous uncertainty. We have provided the mathematical formulations and algorithms for solving two-stage and multistage stochastic mixed-integer linear/nonlinear programming problems. Endogenous uncertainty is decision-dependent. Therefore, the scenario tree representation and the mathematical formulation of multistage stochastic programming under endogenous uncertainty differ from those of stochastic programming under exogenous uncertainty. To provide an intuitive view of the modeling framework, some process network design problems and a pooling problem are used as illustrative examples. Finally, we discussed how the scenario trees in stochastic programming can be generated from time-series models or historical data.

Despite the advances in applying stochastic programming in process systems, several challenges exist for its broader applications. First, it is often difficult to convince practitioners without any mathematical programming background to adopt stochastic programming

methods in their problems. To that end, more tutorials and textbooks for practitioners rather than mathematicians need to be developed. Also, simulation can be used to verify the advantages of using stochastic programming. Second, the types of problems that can be solved efficiently are still limited. A *utopia problem* would be multistage stochastic mixed-integer nonlinear programming under both exogenous and endogenous uncertainty with an arbitrary probability distribution that can be stagewise dependent. The current algorithmic development and computational resources are still far from solving the *utopia problem* efficiently. The major challenge for algorithm development remains to be handling nonconvexity. Third, risk measures can be better integrated into stochastic programming, such as the ones used in chance-constrained programming. Other risk measures, such as Expected CVaR (conditional value at risk) (Pflug and Pichler, 2016), stochastic dominance (SD) (Ruszczyński, 2010), can also be used. Fourth, the connections of stochastic programming with the other 14 communities of decision-making under uncertainty (Powell, 2016) can be better established. Last but not least, generating a scenario tree that has a low error in practice requires high fidelity historical data and better forecast methods. The recent popularity and advances in data science and machine learning bring opportunities towards this direction.

# Chapter 3

# An Improved L-shaped Method for Two-stage Convex 0-1 Mixed Integer Nonlinear Stochastic Programs

## 3.1 The improved L-shaped algorithm

In problem (2.1), only linear constraints are included. In this chapter, we extend (2.1) by considering convex nonlinear constraints and 0-1 mixed-integer variables in both first and second stage. The deterministic equivalent of the two-stage stochastic program we address in this chapter is defined as $(P)$ given by equation (3.1a)-(3.1f).

$$(P): \quad \min \quad z = c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{3.1a}$$

$$s.t. \quad A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{3.1b}$$

$$A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{3.1c}$$

$$g_{2,\omega}(y_\omega) \leq b_{2,\omega} \quad \forall \omega \in \Omega \tag{3.1d}$$

$$x \in X, \quad X = \left\{ x : x_i \in \{0,1\}, \forall i \in I_1, \ 0 \leq x \leq x^{ub} \right\} \tag{3.1e}$$

$$y_\omega \in Y \quad \forall \omega \in \Omega, \quad Y = \left\{ y : y_j \in \{0,1\}, \forall j \in J_1,\ 0 \le y \le y^{ub} \right\} \tag{3.1f}$$

Here, $x$ represents the first stage decisions. $y_\omega$ represents the second stage decisions in scenario $\omega$. $g_0$, $g_{1,\omega}$, $g_{2,\omega}$, are convex functions. Both the first and the second stage decisions are mixed-integer. Let $I = \{1, \cdots, n\}$ be the index set of all the first stage variables. $I_1 \subseteq I$ is the subset for indices of the binary first stage variables. Let $J = \{1, \cdots, m\}$ be the index set of all the second stage variables. $J_1 \subseteq J$ is the subset for the indices of the binary second stage variables. $x^{ub}$ is a vector that represents the upper bound of all the first stage variables. $y^{ub}$ is a vector that represents the upper bound of all the second stage variables. In this chapter, we assume problem $(P)$ has relatively complete recourse. Solving $(P)$ with a large number of scenarios directly using mixed-integer nonlinear programming (MINLP) solvers is prohibitive since the computational time grows exponentially with the number of scenarios. In order to solve problem $(P)$ more efficiently, we propose an improved L-shaped method that is based on Benders decomposition (Van Slyke and Wets, 1969). As discussed in the introduction section, the basic idea of Benders decomposition is to decompose $(P)$ into a Benders master problem which only contains first stage decisions and Benders subproblems each of which only contains second stage decisions for a given scenario. In this chapter, we add both (strengtened) Benders cuts and Lagrangean cuts to the Benders master problem to tighten the relaxation.

We first define the Benders master problem $(MB^k)$, with both Lagrangean cuts (3.2c), Benders cuts (3.2d), and strengthened Benders cuts (3.2e) for a given set of iterations $k \in K$. $z^k_{MB}$ is the objective of the Benders master problem at iteration $k$. $z^{*k}_{SL,\omega}$ is the optimal value of the Lagrangean subproblem $(SL^k_\omega)$. $z^{*k}_{SB,\omega}$ and $z^{*k+}_{SSB,\omega}$ are the optimal values of the Benders subproblem $(SB^k_\omega)$ and strengthened Benders subproblem $(SSB^{k+}_\omega)$. All the subproblems are defined next. Note that (3.2e) and (3.2d) are the same type of cuts, i.e., (3.2e) is a strengthened version of (3.2d). Only one of (3.2e) and (3.2d) are included in the Benders master problem. $\tilde{x}^k$ represents the optimal solution to the Bender master problem

in iteration $k$.

$$(MB^k): \quad \min \quad z^k_{MB} = \sum_{\omega} \eta_\omega \tag{3.2a}$$

$$s.t. \quad A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{3.2b}$$

$$\eta_\omega \geq z^{*k}_{SL,\omega} - \mu^k_\omega x \quad \forall \omega \in \Omega, k \in K \tag{3.2c}$$

$$\eta_\omega \geq z^{*k}_{SB,\omega} + (\lambda^k_\omega)^T (x - \tilde{x}^k) + \tau_\omega c^T x \quad \forall \omega \in \Omega, k \in K \tag{3.2d}$$

$$\eta_\omega \geq z^{*k+}_{SSB,\omega} + (\lambda^k_\omega)^T (x - \tilde{x}^k) + \tau_\omega c^T x \quad \forall \omega \in \Omega, k \in K \tag{3.2e}$$

$$x \in X \tag{3.2f}$$

Next, we show how the valid inequalities (3.2c), (3.2d) and (3.2e) can be derived.

***Lagrangean cuts.*** The deterministic equivalent problem $(P)$ can be reformulated by duplicating the first stage decisions $x$ for each scenario $\omega$ and adding nonanticipativity constraints $(NACs)$, $x_{\omega 1} = x_{\omega 2}, x_{\omega 1} = x_{\omega 3}, \cdots, x_{\omega 1} = x_{\omega |\Omega|}$, to guarantee that all the first stage decisions made for all the scenarios are the same. The $NACs$ can be dualized by multiplying $\pi_\omega$ to the constraints, $x_{\omega 1} = x_{\omega+1}$, $\omega = \omega_1, \omega_2, \cdots, \omega_{|\Omega|-1}$. Then, the deterministic equivalent problem can be decomposed into $|\Omega|$ scenarios. Each subproblem is defined as a Lagrangean subproblem $(SL^k_\omega)$ for iteration $k$:

$$(SL^k_\omega): \quad \min \quad z^k_{SL,\omega} = \tau_\omega (c^T x_\omega + d^T_\omega y_\omega) + \mu^k_\omega x_\omega \tag{3.3a}$$

$$s.t. \quad A_0 x_\omega \geq b_0, \quad g_0(x_\omega) \leq 0 \tag{3.3b}$$

$$A_{1,\omega} x_\omega + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \tag{3.3c}$$

$$g_{2,\omega}(y_\omega) \leq b_{2,\omega} \tag{3.3d}$$

$$x_\omega \in X \tag{3.3e}$$

$$y_\omega \in Y \tag{3.3f}$$

where $\mu^k_{\omega 1} = \sum_{\omega=\omega 1}^{\omega_{|\Omega|-1}} \pi^k_\omega$, $\mu^k_{\omega+1} = -\pi^k_\omega$, $\omega = \omega_1, \omega_2, \cdots, \omega_{|\Omega|-1}$. Each Lagrangean subproblem $(SL^k_\omega)$ is solved to optimality. The optimal value of the subproblem $\omega$ at iteration $k$ is defined as $z^{*k}_{SL,\omega}$. $\eta_\omega \geq z^{*k}_{SL,\omega} - \mu^k_\omega x$ is a valid inequality for the Benders master problem defined as a Lagrangean cut. The proof of the validity of the Lagrangean cuts is shown in proposition 1 in Appendix A. A similar proof can be found in Ogbe (2016). After solving the Lagrangean subproblems at each iteration $k$, the multipliers of the Lagrangean subproblems can be updated using the subgradient method (Oliveira et al., 2013), which is described in Appendix B. After adding the Lagrangean cuts to the Benders master problem, the lower bound obtained by solving the Benders master problem is at least as tight as the lower bound that can be obtained by using Lagrangean decomposition. The proof is shown in proposition 2 in Appendix A.

***Benders cuts.*** Assume the solution to the Benders master problem at iteration $k$ is $\tilde{x}^k$, $x$ is fixed at $\tilde{x}^k$ for the relaxation of the deterministic equivalent problem $(RP)$. $(RP)$ can be decomposed into $|\Omega|$ subproblems. Each subproblem is defined as the Benders subproblem $(SB^k_\omega)$:

$$(SB^k_\omega): \quad \min \quad z^k_{SB,\omega} = \tau_\omega d^T_\omega y_\omega \tag{3.4a}$$

$$s.t. \quad x = \tilde{x}^k \tag{3.4b}$$

$$A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq 0 \tag{3.4c}$$

$$g_{2,\omega}(y_\omega) \leq b_{2,\omega} \tag{3.4d}$$

$$0 \leq y_\omega \leq y^{ub} \tag{3.4e}$$

According to Geoffrion (1972), a Benders cut can be generated at iteration $k$:

$$\begin{aligned} \eta_\omega \geq\ & \tau_\omega d^T_\omega \tilde{y}^k_\omega + (\lambda^k_\omega)^T (x - \tilde{x}^k) + (\chi^k_\omega)^T (g_{1,\omega}(\tilde{y}^k_\omega) - b_{1,\omega} + A_{1,\omega}\tilde{x}^k) \\ & + (\nu^k_\omega)^T (g_{2,\omega}(\tilde{y}^k_\omega) - b_{2,\omega}) + \psi^k_{\omega l}(0 - \tilde{y}^k_\omega) + \psi^k_{\omega u}(\tilde{y}^k_\omega - y^{ub}) + \tau_\omega c^T x \\ & = z^{*k}_{SB,\omega} + (\lambda^k_\omega)^T (x - \tilde{x}^k) + \tau_\omega c^T x \end{aligned} \tag{3.5a}$$

where $\lambda^k_\omega, \chi^k_\omega, \nu^k_\omega, \psi^k_{\omega l}, \psi^k_{\omega u}$, are the optimal dual multipliers for constraints (3.4b)-(3.4e) at iteration $k$. $z^{*k}_{SB,\omega}$ is the optimal value of $(SB^k_\omega)$. $\tilde{y}^k_\omega$ is the optimal solution of $(SB^k_\omega)$.

***Strengthened Benders cuts.*** The strengthened Benders subproblem is defined as follows:

$$(SSB_\omega^{k-}): \quad \min \quad z_{SSB,\omega}^k = \tau_\omega d_\omega^T y_\omega \tag{3.6a}$$

$$s.t. \quad x = \tilde{x}^k \tag{3.6b}$$

$$A_{1,\omega}x + g_{1,\omega}(y_\omega) \leq 0 \tag{3.6c}$$

$$g_{2,\omega}(y_\omega) \leq b_{2,\omega} \tag{3.6d}$$

$$0 \leq y_\omega \leq y^{ub} \tag{3.6e}$$

$$(\alpha_{1\omega}^{jk'})^T x + (\alpha_{2\omega}^{jk'})^T y_\omega \leq \beta_\omega^{jk'} \quad \forall j \in J_{1\omega}^{k'}, k' \leq k-1 \tag{3.6f}$$

where (3.6f) are the valid inequalities that have been derived before the $k$th iteration. In the first iteration, $(SSB_\omega^{k-})$ reduces to $SB_\omega^k$. The minus sign in the notation of $(SSB_\omega^{k-})$ means that the valid inequalities for iteration $k$ have not been derived. Later on, we will denote the strengthened Benders subproblem in iteration $k$ that includes the valid inequalities of iteration $k$ as $(SSB_\omega^{k+})$. As $(SSB_\omega^{k-})$ is a relaxation of the convex MINLP, some $(\tilde{y}_\omega^k)_j$, $j \in J_1$, may be fractional. However, the second stage variables $(y_\omega)_j$, $j \in J_1$ have to satisfy the binary constraints. Therefore, the following disjunctions hold for each $j \in J_1$, which means that the binary second stage variables can be either 0 or 1.

$$\begin{bmatrix} A_0 x \geq b_0, \quad g_0(x) \leq 0 \\ A_{1,\omega}x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \\ g_{2,\omega}(y_\omega) \leq b_{2,\omega} \\ 0 \leq y_\omega \leq y^{ub} \\ (y_\omega)_j = 1 \end{bmatrix} \vee \begin{bmatrix} A_0 x \geq b_0, \quad g_0(x) \leq 0 \\ A_{1,\omega}x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \\ g_{2,\omega}(y_\omega) \leq b_{2,\omega} \\ 0 \leq y_\omega \leq y^{ub} \\ (y_\omega)_j = 0 \end{bmatrix} \quad \forall j \in J_1 \tag{3.7}$$

From the disjunctions, valid inequalities called lift-and-project cuts can be derived, which were first proposed by Balas et al. (1993) in the MILP setting and extended to convex MINLP by Stubbs and Mehrotra (1999). In order to derive the lift-and-project cuts, in each iteration $k$ for each scenario $\omega$, the minimum distance problem $(MD_\omega^{jk})$ is solved for all $j \in J_{1\omega}^k$ where

$J^k_{1\omega}$ is the index set that includes the indices of all the fractional $(\tilde{y}^k_\omega)_j$, $j \in J_1$. The objective function is any norm of the difference between $(x, y_\omega)$ and the fractional solution $(\tilde{x}^k, \tilde{y}^k_\omega)$. $(u^d, v^d_\omega)$ is the disaggregated variable for disjunct $d$. $\gamma_d$, $d = 1, 2$, are variables between 0 and 1 that represent the weight of each disjunct. From Ceria and Soares (1999), (3.8b)-(3.8i) is the convex hull representation of the disjunction that enforces that the binary variable $(y_\omega)_j$ should be either 0 or 1.

$$(MD^{jk}_\omega): \quad \min \quad ||(x, y_\omega) - (\tilde{x}^k, \tilde{y}^k_\omega)|| \tag{3.8a}$$

$$s.t. \quad x = \sum_{d=1,2} u^d, \quad y_\omega = \sum_{d=1,2} v^d_\omega \tag{3.8b}$$

$$\gamma_1 + \gamma_2 = 1, \quad \gamma_1, \gamma_2 \geq 0 \tag{3.8c}$$

$$0 \leq v^d_\omega \leq y^{ub}\gamma_d, \quad 0 \leq u^d \leq x^{ub}\gamma_d \quad d = 1, 2 \tag{3.8d}$$

$$A_0 u^d \geq b_0 \gamma_d, \quad \gamma_d g_0(u^d/\gamma_d) \leq 0 \quad d = 1, 2 \tag{3.8e}$$

$$A_{1,\omega} u^d + \gamma_d g_{1,\omega}(v^d_\omega/\gamma_d) \leq b_{1,\omega}\gamma_d \quad d = 1, 2 \tag{3.8f}$$

$$\gamma_d g_{2,\omega}(v^d_\omega/\gamma_d) \leq 0 \quad d = 1, 2 \tag{3.8g}$$

$$(v^1_\omega)_j = 0 \tag{3.8h}$$

$$(v^2_\omega)_j = \gamma_2 \tag{3.8i}$$

Let $(\bar{x}^{jk*}_\omega, \bar{y}^{jk*}_\omega)$ be the optimal solution of $(MD^{jk}_\omega)$. Stubbs and Mehrotra (1999) proved that $(\xi^{jk}_\omega)^T \begin{pmatrix} x - \bar{x}^{jk*}_\omega \\ y_\omega - \bar{y}^{jk*}_\omega \end{pmatrix} \geq 0$ is a valid inequality for the fullspace problem $(P)$ where $\xi^{jk}_\omega$ is one subgradient of the objective function evaluated at the optimal solution of $(MD^{jk}_\omega)$. The cut is able to cut off the fractional solution $(\tilde{x}^k, \tilde{y}^k_\omega)$, if the optimal distance is strictly greater than 0.

For the 2-norm, $\xi^{jk}_\omega = 2 \begin{pmatrix} \bar{x}^{jk*}_\omega - \tilde{x}^k \\ \bar{y}^{jk*}_\omega - \tilde{y}^k_\omega \end{pmatrix}$. The appropriate subgradient for the 1-norm and the $\infty-$norm can be found in the work of Stubbs and Mehrotra (1999).

The nonlinear lift-and-project cuts can be expensive since they require to solve an NLP for each fractional binary variable for each scenario in each iteration $k$. Inspired by the work of Zhu and Kuno (2006) and Kılınç et al. (2017) who outer-approximate the convex nonlinear functions to avoid solving the NLPs, we outer-approximate the nonlinear constraints and solve the cut generating linear program $(CGLP_\omega^{jk})$ instead of $(MD_\omega^{jk})$.

$$(CGLP_\omega^{jk}): \quad \min \quad ||(x, y_\omega) - (\tilde{x}^k, \tilde{y}_\omega^k)|| \tag{3.9a}$$

$$s.t. \quad x = \sum_{d=1,2} u^d, \quad y_\omega = \sum_{d=1,2} v_\omega^d \tag{3.9b}$$

$$\gamma_1 + \gamma_2 = 1 \tag{3.9c}$$

$$0 \leq v_\omega^d \leq y^{ub}\gamma_d, \quad 0 \leq u^d \leq x^{ub}\gamma_d \quad d = 1, 2 \tag{3.9d}$$

$$A_0 u^d \geq b_0 \gamma_d, \quad \gamma_d g_0(x^l) + \nabla g_0(x^l)^T \big(u^d - x^l \gamma_d\big) \leq 0 \quad d = 1, 2, l \in L \tag{3.9e}$$

$$A_{1,\omega} u^d + \gamma_d g_{1,\omega}(y_\omega^l) + \nabla g_{1,\omega}(y_\omega^l)^T \big(v_\omega^d - y_\omega^l \gamma_d\big) \leq b_{1,\omega}\gamma_d \quad d = 1, 2, l \in L \tag{3.9f}$$

$$\gamma_d g_{2,\omega}(y_\omega^l) + \nabla g_{2,\omega}(y_\omega^l)^T \big(v_\omega^d - y_\omega^l \gamma_d\big) \leq 0 \quad d = 1, 2, l \in L \tag{3.9g}$$

$$(v_\omega^1)_j = 0 \tag{3.9h}$$

$$(v_\omega^2)_j = \gamma_2 \tag{3.9i}$$

where $(x^l, y_\omega^l)$ are the points that we choose to outer-approximate the nonlinear constraints. $L$ is the index set of such points. Note that in order to solve an LP instead of an NLP for the cut generating process, we can only use the 1-norm or the $\infty$-norm in the objective of $(CGLP_\omega^{jk})$. The lift-and-project cuts derived by solving $(CGLP_\omega^{jk})$ are weaker than those derived by solving $(MD_\omega^{jk})$, since the feasible region of $(CGLP_\omega^{jk})$ is a relaxation of the feasible region of $(MD_\omega^{jk})$. So there is a tradeoff between computational efficiency and the tightness of the cuts that are derived. For the problems that we investigate, $(CGLP_\omega^{jk})$ is able to provide good computational results, which will be discussed in section 4. For the derivation of the lift-and-project cuts based the solution of $(CGLP_\omega^{jk})$, we refer to the work of

Kılınç et al. (2017). Here, we simply denote the lift-and-project cut derived from $(CGLP_\omega^{jk})$ as:

$$(\alpha_{1\omega}^{jk})^T x + (\alpha_{2\omega}^{jk})^T y_\omega \leq \beta_\omega^{jk} \tag{3.10}$$

The lift-and-project cuts that are obtained by solving $(MD_\omega^{jk})$ or $(CGLP_\omega^{jk})$ are added to the Benders subproblem $(SB_\omega^k)$. The strengthened Benders subproblem with the lift-and-project cuts that are derived in iteration $k$ is defined as $(SSB_\omega^{k+})$ where the plus sign simply means that the newly derived cuts are included:

$$(SSB_\omega^{k+}): \quad \min \quad z_{SSB,\omega}^k = \tau_\omega d_\omega^T y_\omega \tag{3.11a}$$

$$s.t. \quad x = \tilde{x}^k \tag{3.11b}$$

$$A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq 0 \tag{3.11c}$$

$$g_{2,\omega}(y_\omega) \leq b_{2,\omega} \tag{3.11d}$$

$$0 \leq y_\omega \leq y^{ub} \tag{3.11e}$$

$$(\alpha_{1\omega}^{jk'})^T x + (\alpha_{2\omega}^{jk'})^T y_\omega \leq \beta_\omega^{jk'} \quad \forall j \in J_{1\omega}^{k'}, k' \leq k \tag{3.11f}$$

where (3.11f) are the lift-and-project cuts that are derived until iteration $k$ for scenario $\omega$. A strengthened Benders cut can be derived by solving the strengthened Benders subproblem in the same way that we derive the Benders cuts:

$$\eta_\omega \geq z_{SSB,\omega}^{*k+} + (\lambda_\omega^k)^T (x - \tilde{x}^k) + \tau_\omega c^T x \tag{3.12}$$

Note that we slightly abuse notation here: $\lambda_\omega^k$ is used to represent the optimal dual variable for both the Benders subproblem $(SB_\omega^k)$ and the strengthened Benders subproblem $(SSB_\omega^{k+})$.

***Upper bounding procedure.*** After the Benders master problem is solved at iteration $k$, $x$ in problem $(P)$ is fixed at $\tilde{x}^k$. The rest of the problem can be decomposed into $|\Omega|$ scenarios. A feasible solution to the deterministic equivalent problem can be obtained by solving the upper bound subproblems $(UB_\omega^k)$ defined by (3.13a)-(3.13d). Let $z_{UB,\omega}^{*k}$ be the optimal value

of $(UB_\omega^k)$. An upper bound of $(P)$ can be obtained by calculating $c^T\tilde{x}^k + \sum_{\omega\in\Omega} z_{UB,\omega}^{*k}$.

$$(UB_\omega^k): \quad \min \quad z_{UB,\omega}^k = \tau_\omega d_\omega^T y_\omega \tag{3.13a}$$

$$s.t. \quad g_{1,\omega}(y_\omega) \leq b_{1,\omega} - A_{1,\omega}\tilde{x}^k \tag{3.13b}$$

$$g_{2,\omega}(y_\omega) \leq b_{2,\omega} \tag{3.13c}$$

$$y_\omega \in Y \tag{3.13d}$$

The steps of the improved L-shaped method are outlined in Figure 3.1. The lower bound is initialized to $-\infty$. The upper bound is initialized with the optimal value obtained by solving the stochastic program with the first stage decisions fixed at the optimal solution of the expected value problem $(EEV)$ (Birge and Louveaux, 2011). The Lagrangean multipliers are initialized to 0. At each iteration, the Lagrangean subproblems, the Benders master problems, the strengthened Benders subproblems, the cut generating linear program, the strengthened Benders subproblem with the newly generated cuts, and the upper bound subproblems are solved sequentially. The optimal solution of the Benders master problem is used to initialize the strengthened Benders subproblems, the cut generating linear programs, the strengthened Benders subproblems with the newly generated cuts and the upper bound subproblems. Valid inequalities for the Benders master problem can be derived by solving the Lagrangean subproblems, the strengthened Benders subproblems with the newly generated cuts. The algorithm iterates until the relative optimality gap is within a given tolerance $\epsilon$.

However, for problems with good NLP relaxations where the Benders cuts are tight, we can just solve the Benders subproblems without generating the lift-and-project cuts. Note that in each iteration, we can add either Benders cuts or strengthened Benders cuts to the master problem, but not both, because the strengthened Benders cuts are derived from the subproblems $(SSB_\omega^{k+})$ which have tighter relaxations than the Benders subproblems $SB_\omega^k$. Therefore, the strengthened Benders cuts dominate the Benders cuts.

**The proposed algorithm**

The steps of the improved L-shaped method are outlined below:

**1. Initialization**

Set $\mu_\omega^0 = 0$ for all $\omega \in \Omega$, $k = 0$, $LB = -\infty$, $UB = EEV$(expected result of using the expected value problem solution).

Go to Step 2.

**2. Lagrangean subproblem**

For fixed $\mu_\omega^k$, solve each Lagrangean subproblem $SL_\omega^k$ in parallel. The optimal solution is $\hat{x}_\omega^k, \hat{y}_\omega^k, z_{SL,\omega}^{*k}$. Add the Lagrangean cuts, $\eta_\omega \geq z_{SL,\omega}^{*k} - \mu_\omega^k x$, to the Benders master problem.

If $\sum_{\omega\in\Omega} z_{SL,\omega}^{*k} \geq LB$, set $LB = \sum_{\omega\in\Omega} z_{SL,\omega}^{*k}$.

Update $\mu_\omega^k \to \mu_\omega^{k+1}$ using the subgradient method shown in Appendix B.

Go to Step 3.

**3. Benders master problem**

Solve the Benders master problem. Obtain the optimal solution $\tilde{x}^k$ and $z_{MB}^{*k}$.

If $z_{MB}^{*k} \geq LB$, set $LB = z_{MB}^{*k}$.

Go to Step 4.

**4. Strengthened Benders subproblem**

Fix $x = \tilde{x}^k$. Solve each strengthened Benders subproblem $SSB_\omega^{k-}$ in parallel. Obtain the optimal primal solution $\tilde{y}_\omega^k, z_{SSB,\omega}^{*k-}$ and the optimal dual variable $\lambda_\omega^k$.

If there is any fractional variable $(\tilde{y}_\omega^k)_j$, $j \in J_1$, denote the index set of fractional variables as $J_{1\omega}^k$ for all $\omega \in \Omega$ and go to Step 5.

Otherwise, add the strengthened Benders cuts, $\eta_\omega \geq z_{SSB,\omega}^{*k-} + (\lambda_\omega^k)^T(x - \tilde{x}^k) + \tau_\omega c^T x$, to the Benders master problem and go to Step 7.

**5. Cut generating linear program**

Solve in parallel the cut generating linear programs $(CGLP_\omega^{jk})$ for all $\omega \in \Omega, j \in J_{1\omega}^k$. Obtain the lift-and-project cuts, $(\alpha_{1\omega}^{jk'})^T x + (\alpha_{2\omega}^{jk'})^T y_\omega \leq \beta_\omega^{jk'}$, and add them to the strengthened Benders subproblem $(SSB_\omega^{k-})$. The subproblem with the newly generated lift-and-project cuts is defined as $(SSB_\omega^{k+})$.

Go to Step 6.

**6. Strengthened Benders subproblem**

Solve the strengthened Benders subproblem ($SSB_\omega^{k+}$) in parallel. Obtain the optimal primal objective value $z_{SSB,\omega}^{*k+}$ and optimal dual variable $\lambda_\omega^k$.

Add the strengthened Benders cuts, $\eta_\omega \geq z_{SSB,\omega}^{*k+} + (\lambda_\omega^k)^T(x - \tilde{x}^k) + \tau_\omega c^T x$, to the Benders master problem and go to Step 7.

**7. Upper bound subproblem**

Fix $x = \tilde{x}^k$ in each upper bound subproblem and solve them in parallel. Obtain the optimal solution to the upper bound subproblem $\bar{y}_\omega^k, z_{UB,\omega}^{*k}$.

If $c^T\tilde{x}^k + \sum_{\omega\in\Omega} z_{UB,\omega}^{*k} \leq UB$, set $UB = c^T\tilde{x}^k + \sum_{\omega\in\Omega} z_{UB,\omega}^{*k}$.

**8. Optimality check**

If $UB \leq (1+\epsilon)LB$, stop.

Else set $k = k+1$ and include in $K$; go back to Step 2.

Although the algorithm can always yield a lower bound that is at least as tight as using Lagrangean decomposition, it still does not have theoretical guarantee of convergence. An intuitive explanation is that there is a duality gap if the Lagrangean cuts alone are added to the Benders master problem. While the strengthened Benders cuts are tighter than the Benders cuts as rank-one lift-and-project cuts are added to convexify the MINLP subproblems, the strengthened Benders subproblems are rank-one lift-and-project closure but not the convex hull of the original convex MINLP subproblems. The rank-one lift-and-project cuts can close a significant portion of the integrality gap for the original convex MINLP subproblems, but they do not have theoretical guarantee to close the integrality gap. Therefore, while combining Lagrangean cuts and strengthened Benders cuts can make the relative optimality gap smaller than both the duality gap and the integrality gap, the proposed algorithm does not have guarantee to close the optimality gap.

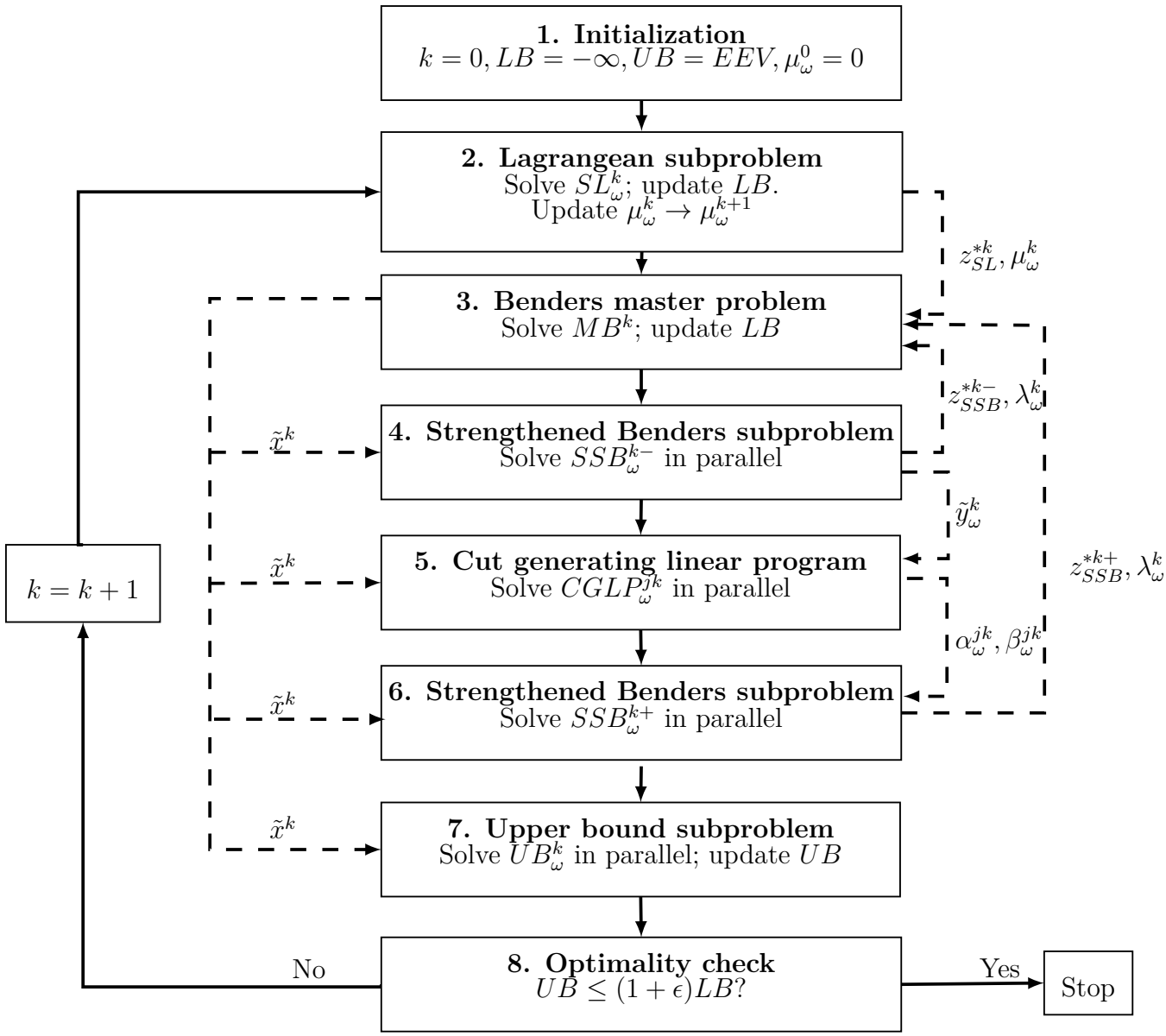


Figure 3.1: Algorithmic flow(solid line) and information flow(dotted line) of the improved L-shaped method

## 3.2 Motivating examples

### 3.2.1 Optimal Design of Multi-product Batch Plant under Demand Uncertainty

**Indices**

$i$ = products

$j$ = stages

$k$ = possible integer values for number of processing units

$\omega$ = scenarios

**Parameters**

$Q_{i\omega}$ = Demand of product $i$ in scenario $\omega$ $(kg)$

$S_{ij}$ = Size factor for product $i$ in processing stage $j$ $(L/kg)$

$t_{ij}$ = Processing time for product $i$ in processing stage $j$ $(hours)$

$H$ = Fixed production horizon time $(hours)$

$\alpha_i$ = Cost coefficient for purchasing unit in processing stage $j$ (\$)

$\beta_j$ = Cost exponent for the volume of processing unit in stage $j$

$\lambda_j$ = Fixed cost of using one unit in stage $j$

$\delta$ = Penalty cost for exceeding production horizon (\$/hour)

$p_\omega$ = Probability of scenario $\omega$

$V_j^L$ = minimum volume of the processing units in stage $j$

$V_j^U$ = maximum volume of the processing units in stage $j$

**First stage decisions**

$V_j$ = Size of processing unit s for stage $j$

$v_j$ = Nature logarithm of $V_j$

$N_j^F$ = Number of processing units purchased for stage $j$

$n_j^F$ = Natural logarithm of $N_j^F$

$Y_{kj}^F$ = Binary variable. Equals 1 if $N_j^F = k$

**Second stage decisions**

$B_{i\omega}$ = Batch size of product $i$ in scenario $\omega$

$b_{i\omega}$ = Natural logarithm of $B_{i\omega}$

$TL_{i\omega}$ = Cycle time of product $i$ in scenario $\omega$

$tl_{i\omega}$ = Natural logarithm of $TL_{i\omega}$

$N^S_{j\omega}$ = Number of processing units that are operating in stage $j$ in scenario $\omega$

$n^S_{j\omega}$ = Natural logarithm of $N^S_{j\omega}$

$Y^S_{kj\omega}$ = Binary variable. Equals 1 if $N^S_{j\omega} = k$

#### 3.2.1.1 Problem statement

We study a multi-product multi-stage batch plant (Figure 3.2) under demand uncertainty, which is an extension of the work of Grossmann and Sargent (1979). Every product has to be processed through all the stages. It is assumed that in each stage there can be multiple processing units with equal volume. The number and the volume of the processing units purchased have to be decided before the demand for each product is known. After the demand of each product is realized, operating decisions including the number of processing units actually used and the batch size of each product have to be made. We assume the batch sizes for any given product are the same. Not all the processing units purchased have to be used, i.e., processing units can be idle when the demand is low. There will be a fixed cost for using one processing unit, so one should try to use as few processing units as possible. One should also try to complete all the processing tasks within a given time horizon and there is a penalty cost for exceeding the time horizon to meet the demands. This problem can be formulated as a two-stage stochastic program where the first stage decisions are the number and volume of the processing units. The second stage decisions are the batch size of each product, the actual number of processing units used in each stage, and the cycle time of each product.

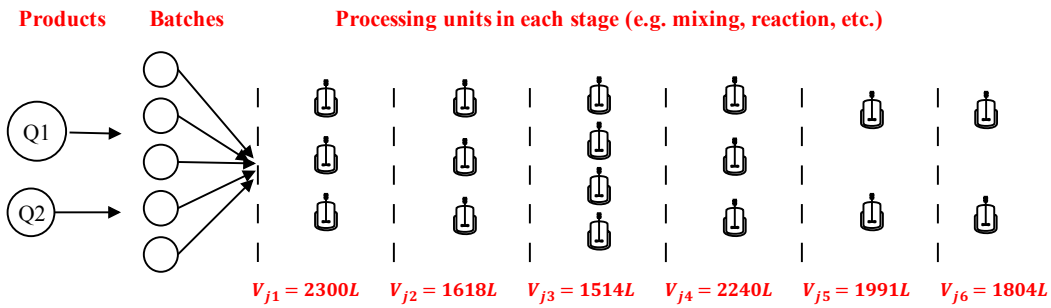


Figure 3.2: A multi-product multi-stage batch plant for two products with corresponding demands $Q_1$ and $Q_2$

#### 3.2.1.2 Mathematical formulation

Constraints (3.14)-(3.15) determine the number of processing units purchased. $Y^F_{kj}$ is the binary variable which equals to one when the number of processing units purchased at stage $j$, $N^F_j$, equals to $k$.

$$\sum_k Y^F_{kj} = 1 \quad \forall j \tag{3.14}$$

$$N^F_j = \sum_k k Y^F_{kj} \quad \forall j \tag{3.15}$$

The volume of any processing unit has to lie within $V^L$ and $V^U$.

$$V^L_j \leq V_j \leq V^U_j \quad \forall j \tag{3.16}$$

Constraint (3.17) enforces the batch size in any scenario times a size factor cannot exceed the volume of the processing unit purchased in stage $j$.

$$V_j \geq S_{ij} B_{i\omega} \quad \forall i, j, \omega \tag{3.17}$$

$N^S_{j\omega}$ denotes the number of processing units operating in stage $j$ in scenario $\omega$. The binary variable $Y^S_{kj\omega}$ is used to guarantee the integrality of $N^S_{j\omega}$ (constraint (3.18) and (3.19)). The number of units that is operating in any scenario $\omega$ should be less than or equal to the number of units purchased, which is enforced by constraint (3.20).

$$\sum_k Y^S_{kj\omega} = 1 \quad \forall j, \omega \tag{3.18}$$

$$N_{j\omega}^S = \sum_k kY_{kj\omega}^S \quad \forall j, \omega \tag{3.19}$$

$$N_{j\omega}^S \leq N_j^F \quad \forall j, \omega \tag{3.20}$$

The number of operating processing units times the cycle time should be greater or equal to any processing time $t_{ij}$.

$$N_{j\omega}^S TL_{i\omega} \geq t_{ij} \quad \forall i, j, \omega \tag{3.21}$$

The demand for product $i$ in scenario $\omega$ is given by $Q_{i\omega}$ and the batch size is $B_{i\omega}$. Therefore, the number of batches is given by $\frac{Q_{i\omega}}{B_{i\omega}}$. The time consumed on producing product $i$ equal the number of batches of product $i$ times its cycle time. The summation over the processing time of all products should be less than or equal to the given time horizon plus slack variable $L_\omega$ representing the lateness in scenario $\omega$.

$$\sum_i \frac{Q_{i\omega} TL_{i\omega}}{B_{i\omega}} \leq H + L_\omega \quad \forall \omega \tag{3.22}$$

The total cost includes the investment of processing units, the fixed and variable costs of operating the processing units, and the lateness penalty. As the variable cost of operating the processing units is a constant that is proportional to the demand, it is not included in the objective.

$$cost = \sum_j \alpha_j N_j^F V_j^{\beta_j} + \sum_\omega p_\omega \Big( \sum_j \lambda_j N_{j\omega}^S + \delta L_\omega \Big) \tag{3.23}$$

The formulation, which includes constraints (3.14)-(3.23), is a nonconvex MINLP. However, according to Kocis and Grossmann (1988), this model can be reformulated as a convex MINLP by considering exponential transformations of variables $V_j, N_j^F, B_{i\omega}, TL_{i\omega}$ and rewriting constraint (3.15)-(3.17), (3.19)-(3.23) as (3.24)-(3.31).

$$n_j^F = \sum_k \ln(k) Y_{kj}^F \quad \forall j \tag{3.24}$$

$$\ln(V^L) \leq v_j \leq \ln(V^U) \quad \forall j \tag{3.25}$$

$$v_j \geq \ln(S_{ij}) + b_{i\omega} \quad \forall i, j, \omega \tag{3.26}$$

$$n^S_{j\omega} = \sum_k \ln(k) Y^S_{kj\omega} \quad \forall j, \omega \tag{3.27}$$

$$n^S_{j\omega} \leq n^F_j \quad \forall j, \omega \tag{3.28}$$

$$n^S_{j\omega} + tl_{i\omega} \geq \ln(t_{ij}) \quad \forall i, j, \omega \tag{3.29}$$

$$\sum_i Q_{i,\omega} \exp(tl_{i\omega} - b_{i\omega}) \leq H + L_\omega \quad \forall \omega \tag{3.30}$$

This leads to the following convex MINLP:

$$\min \quad cost = \sum_j \alpha_j \exp(n^F_j + \beta_j v_j) + \sum_\omega p_\omega \Big( \sum_j \lambda_j \exp(n^S_{j\omega}) + \delta L_\omega \Big) \tag{3.31}$$

$$s.t. \qquad (3.14), (3.18), (3.24) - (3.30)$$

#### 3.2.1.3 Case study

We study a batch plant with 6 stages, 5 products. Each stage can have at most 4 processing units. We assume that there are 3 scenarios, where the demand of each product can be high, medium, or low with probability of 25%, 50%, 25% respectively. The high and low demand are +10% and -10% away from the mean. The problem is first solved in deterministic equivalent form in GAMS using DICOPT on the 12 processors of an Intel Xeon (2.67GHz) machine with 64 GB RAM. The model has 96 binary variables, 64 continuous variables, 316 constraints. The minimum cost is 423.5($10^3$\$). The NLP relaxation of this problem is 408.4($10^3$\$). The number and volume of each processing units purchased in each stage is shown in Figure 3.2. The number of processing units that are operating in each stage in each scenario is shown in Table 3.1, where $\omega 1, \omega 2, \omega 3$ correspond to high, medium and low demand respectively. One can see that the recourse decisions of each scenario are

different. When the demand is low, one fewer processing unit is used in stages $j2$ and $j4$.

Table 3.1: Optimal number of processing units in each stage in each scenario

| Stage | $j1$ | $j2$ | $j3$ | $j4$ | $j5$ | $j6$ |
|---|---|---|---|---|---|---|
| $N^S_{j\omega 1}$ | 3 | 3 | 4 | 3 | 2 | 2 |
| $N^S_{j\omega 2}$ | 3 | 2 | 4 | 3 | 2 | 2 |
| $N^S_{j\omega 3}$ | 3 | 2 | 4 | 2 | 2 | 2 |

However, if we solve the expected value problem with the demand of all products fixed at their mean values, the number of units purchased at each stage is 3, 2, 4, 3, 2, 2, respectively. Compared to the stochastic solution, the number of units at stage $j2$ decreases from 3 to 2. Therefore, with the expected value solution, the high demand cannot be satisfied without an extension of the time horizon. The expected value solution has an expected cost of 433.4($10^3$\$). The value of stochastic solution ($VSS$) (Birge and Louveaux, 2011) of this problem is 9.9($10^3$\$).

### 3.2.2 Planning under demand uncertainty

**Indices**

$i$ = process

$j$ = chemical

$s$ = production scheme

$r$ = supplier

$p$ = plant

$c$ = customer

$\omega$ = scenario

**Sets**

$I(j)$ = index set of processes that consume chemical $j$

$O(j)$ = index set of processes that produce chemical $j$

$R(j)$ = index set of suppliers that provide chemical $j$

$PS(i)$ = index set of production schemes for process $i$

$JM(i,s)$ = index set of main products for production scheme $s$ of process $i$

$L(i,s)$ = index set of chemicals involved in production scheme $s$ of process $i$ that are not main product for which the input-output relationship with the main product is linear

$\bar{L}(i,s)$ = index set of chemicals involved in production scheme $s$ of process $i$ that are not main product for which the input-output relationship with the main product is logarithmic

**Parameters**

$Q^U_{pi}$ = maximum capacity of process $i$ in plant $p$

$\mu_{ijs}$ = material balance coefficients for chemical $j$ for production scheme $s$ for process $i$

$\rho_{ijs}$ = relative maximum production rate of main product $j$, $j \in JM(i,s)$, for production scheme $s$ for continuous flexible process $i$ referenced to a base scheme $\bar{s}$

$H_i$ = maximum time for which process $i$ is available for production

$D_{cj\omega}$ = demand of chemical $j$ from customer $c$ in scenario $\omega$

$\alpha^C_i$ = fixed cost for installation of process $i$

$\beta^C_i$ = variable cost for installation of process $i$

$\beta^S_{rj}$ = price for purchase of chemical $j$ from supplier $r$

$\beta^{RP}_{rp}$ = variable cost for transporting chemical from supplier $r$ to plant $p$

$\alpha^{RP}_{rp}$ = fixed cost for transporting chemical from supplier $r$ to plant $p$

$\alpha^{PC}_{pc}$ = fixed cost for transporting chemical from plant $p$ to customer $c$

$\beta^{PC}_{pc}$ = variable cost for transporting chemical from plant $p$ to customer $c$

$\delta_{is}$ = unit operating cost for production scheme $s$ for process $i$

$\phi_{cj\omega}$ = penalty cost for not satisfying demand from customer $c$ for chemical $j$ in scenario $\omega$

$PU^U_{rpj}$ = maximum amount of chemical $j$ transported from supplier $r$ to plant $j$

$F^U_{pcj}$ = maximum amount of chemical $j$ transported from plant $j$ to customer $c$

$\tau_\omega$ = probability of scenario $\omega$

**First stage decisions**

$x_{pi}$ = binary variable (1 if process $i$ is installed in plant $p$)

$Q_{pi}$ = capacity of process $i$ in plant $p$

**Second stage decisions**

$PU_{rpj\omega}$ = purchase amount of chemical $j$ of plant $p$ from supplier $r$ in scenario $\omega$

$y^R_{rp\omega}$ = binary variable (1 if a chemical is transported from supplier $r$ to plant $p$ in scenario $\omega$)

$F_{pcj\omega}$ = amount of chemical $j$ transported from plant $p$ to customer $c$ in scenario $\omega$

$y^C_{pc\omega}$ = binary variable (1 if a chemical is transported from plant $p$ to customer $c$ in scenario $\omega$)

$T_{pijs\omega}$ = length of time assigned to the production of main product $j$ for production scheme $s$ of process $i$ in plant $p$ in scenario $\omega$

$\theta_{pijs\omega}$ = amount of main product $j$, $j \in JM(i,s)$, produced from production scheme $s$ of process $i$ in plant $p$ in scenario $\omega$. The mathematical definition is shown in constraint (3.36)

$W_{pijs\omega}$ = amount of chemical $j$ produced from production scheme $s$ of process $i$ in plant $p$ in scenario $\omega$

$S_{cj\omega}$ = slack variable for not satisfying the demand of chemical $j$ from customer $c$ in scenario $\omega$

#### 3.2.2.1 Problem statement

We study a supply chain where raw materials are purchased from suppliers to manufacture products and deliver them to customers as shown in Figure 3.3. Each plant consists of a processing network with several processes interconnected in a finite number of ways. The processes can be dedicated or flexible (Norton and Grossmann, 1994). A flexible process can have multiple production schemes where these schemes can be different in raw materials, products or combination of both. A dedicated process has fixed recipe of raw materials and products, i.e., only one scheme. We assume all the processes considered in this chapter are continuous. Therefore, process capacity can be expressed as production rate. There is a fixed and a variable cost associated with the installation of each process. The demands of

customers are regarded as uncertain parameters, which will be realized after the investment decisions are made. If the manufacturer fails to satisfy the customer demands, extra products can be purchased from other manufacturers and there is a penalty cost for the extra products purchased. This penalty cost can be regarded as the market prices of the products. In this problem, we assume that the prices of the products are another source of uncertainty, which will be realized after the first stage decisions are made. The second stage decisions include the purchase and transportation of raw materials, the production amount in each scheme of each process in each plant, the delivery of products to customers and the purchase of products from other manufacturers. Note that there is a fixed cost for the transportation link between a supplier and a plant or a plant and a customer. Therefore, both the first stage and second stage variables are mixed-integer.

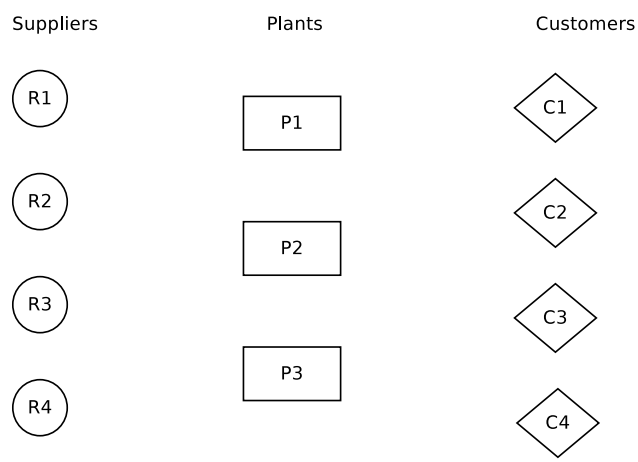


Figure 3.3: Suppliers, Plants, and Customers in Supply Chain

#### 3.2.2.2 Mathematical formulation

The constraints for the investment decisions for each process $i$ in each plant $p$ are as follows:

$$Q_{pi} \leq Q_{pi}^{U} x_{pi} \quad \forall p, i \tag{3.32}$$

where $x_{pi}$ are binary variables to denote whether process $i$ in plant $p$ is installed. Constraint (3.32) forces the capacity to zero if $x_{pi}$ is zero.

Constraint (3.33) represents the mass balance for each chemical $j$ in a given plant $p$. The sum of purchases of raw materials from all suppliers, plus the the sum of output of chemical $j$ from all processes, should be equal to the sum of delivery of $j$ to all potential customers, plus the sum of consumption of $j$ from all processes.

$$\sum_{r \in R(j)} PU_{rpj\omega} + \sum_{i \in O(j)} \sum_{s \in PS(i)} W_{pijs\omega} = \sum_{c} F_{pcj\omega} + \sum_{i \in I(j)} \sum_{s \in PS(i)} W_{pijs\omega} \quad \forall p, j, \omega \tag{3.33}$$

The maximum production rate of a main product $j$ for production scheme $s$ is proportional to the capacity of the plant for a reference scheme $\bar{s}$. For $\bar{s}$, $\rho_{ij\bar{s}}, j \in JM(i, \bar{s})$ equals to 1. The amount produced can be calculated by the production rate times the production time.

$$W_{pijs\omega} = \rho_{ijs} Q_{pi} T_{pijs\omega} \quad \forall p, i, s \in PS(i), j \in JM(i, s), \omega \tag{3.34}$$

For each process $i$, the total production time has to be less than the available time of that process.

$$\sum_{s \in PS(i)} \sum_{j \in JM(i,s)} T_{pijs\omega} \leq H_i \quad \forall p, i, \omega \tag{3.35}$$

Note that constraint (3.34) is nonlinear since it includes the bilinear term $Q_{pi} T_{pijs\omega}$. To linearize it, we define new variable $\theta_{pijs\omega}$ in (3.36) and rewrite (3.34)-(3.35) as (3.37)-(3.38).

$$\theta_{pijs\omega} = Q_{pi} T_{pijs\omega} \quad \forall p, i, j \in JM(i, s), s \in PS(i), \omega \tag{3.36}$$

$$\sum_{s \in PS(i)} \sum_{j \in JM(i,s)} \theta_{pijs\omega} \leq H_i Q_{pi} \quad \forall p, i, \omega \tag{3.37}$$

$$W_{pijs\omega} = \rho_{ijs} \theta_{pijs\omega} \quad \forall p, i, s, s \in PS(i), j \in JM(i, s), \omega \tag{3.38}$$

Constraint (3.39) and (3.40) specifies the input-output relationship between the main product and other chemicals involved in scheme $s$ of process $i$. Note that for some of the processing schemes, the input-output relationship is linear while in others it can be logarithmic. The physical meaning of the logarithmic relationship is that in some of the

processes, the larger the input material flowrate, the less efficient those schemes become to produce products.

$$W_{pijs\omega} = \mu_{ijs} W_{pij's\omega} \quad \forall p, i, j \in L(i,s), j' \in JM(i,s), s \in PS(i), \omega \tag{3.39}$$

$$\ln(1 + W_{pijs\omega}) = \mu_{ijs} W_{pij's\omega} \quad \forall p, i, j \in \bar{L}(i,s), j' \in JM(i,s), s \in PS(i), \omega \tag{3.40}$$

Note that constraint (3.40) is a nonlinear equality which is nonconvex. This equality is relaxed to constraint (3.41) which is convex as pointed out by Kocis and Grossmann (1987). In the optimal solution of the problem, constraint (3.41) is always active because the optimal solution will always tend to make as much product as possible.

$$\ln(1 + W_{pijs\omega}) \geq \mu_{ijs} W_{pij's\omega} \quad \forall p, i, j \in \bar{L}(i,s), j' \in JM(i,s), s \in PS(i), \omega \tag{3.41}$$

Binary variable $y^R_{rp\omega}$ denotes whether a transportation link is established between supplier $r$ and plant $p$, which will force the corresponding purchase amount to zero if the link is not established. Constraint (3.43) serves the same purpose for transportation between plants and customers.

$$PU_{rpj\omega} \leq PU^U_{rpj} y^R_{rp\omega} \quad \forall j, r, r \in R(j), p, \omega \tag{3.42}$$

$$F_{pcj\omega} \leq F^U_{pcj} y^C_{pc\omega} \quad \forall j, p, c, \omega \tag{3.43}$$

The demand of customer $c$ for chemical $j$, $D_{cj\omega}$, must be satisfied by the total flow from all the plants to customer $c$ plus the slack variable, $S_{cj\omega}$, which represents extra purchase of $j$ from other companies if we fail to produce enough to satisfy the demand.

$$\sum_p F_{pcj\omega} + S_{cj\omega} = D_{cj\omega} \quad \forall c, j, \omega \tag{3.44}$$

The objective is to minimize the expected cost, which includes the investment cost, transportation cost, production cost, cost of purchasing raw materials, and penalty cost for

not satisfying the demand.

$$
\begin{aligned}
\min \quad Cost = & \sum_p \sum_i (\beta_i^C Q_{pi} + \alpha_i^C x_{pi}) + \sum_\omega \tau_\omega \Big( \sum_p \sum_i \sum_{s \in PS(i)} \delta_{is} \sum_{j \in JM(i,s)} \rho_{ijs} \theta_{pijs\omega} \\
& + \sum_p \sum_j \sum_{r \in R(j)} (\beta_{rj}^S + \beta_{rp}^{RP}) PU_{rpj\omega} + \sum_r \sum_p \alpha_{rp}^{RP} y_{rp\omega}^R \\
& + \sum_p \sum_c \alpha_{pc}^{PC} y_{pc\omega}^C + \sum_p \sum_c \sum_j \beta_{pc}^{PC} F_{pcj\omega} \\
& + \sum_c \sum_j \phi_{cj\omega} S_{cj\omega} \Big) \\
s.t. \quad & (3.32) - (3.33), (3.37) - (3.39), (3.41) - (3.44)
\end{aligned}
\tag{3.45}
$$

#### 3.2.2.3 Case study

We study a supply chain where we have 4 suppliers, 3 plants, 4 customers as in Figure 3.3. Each plant has the same process network superstructure as shown in Figure 3.4, which is from Norton and Grossmann (1994). As an example, process I1 is a dedicated continuous process producing chemical J3 from chemicals J1 and J6. Process I2 is a flexible continuous process with two production schemes. Scheme S1 for process I2 produces chemical J3 from chemical J1, while scheme S2 produces chemical J4 from chemicals J1 and J6. The input-output relationships for processes I1, I2, I3 are linear. The input-output relationships between chemicals J3 and J5 for scheme S1 and between chemical J4 and chemical J5 for scheme S2 in process I4 are logarithmic. No processes are initially installed.

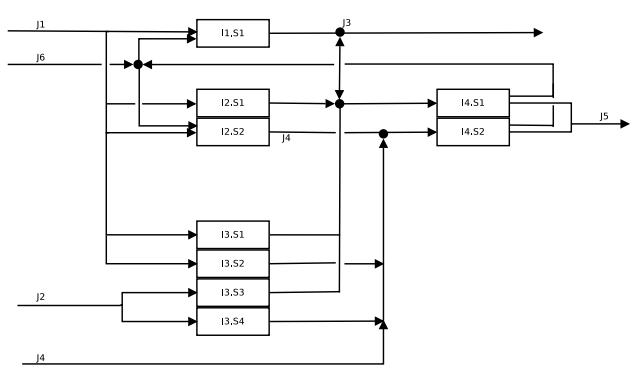


Figure 3.4: Process network for case study

We assume that there are 3 scenarios where the demands of each product can be high, medium or low with probabilities of 25%, 50%, 25%, respectively. The high and low demands are +30% and -30% away from the mean. The convex MINLP problem is solved in deterministic equivalent form in GAMS using DICOPT on the 12 processors of an Intel Xeon (2.67GHz) machine with 64 GB RAM. The model for 3 scenarios has 84 binary variables, 2158 continuous variables, 2167 constraints. The optimal solution yields a minimum cost of 1509.3($10^6$\$). The investment decisions for each of the three plants are shown in Table 3.2. The transportation links in three scenarios are shown in Figure 3.5. It is easy to see different recourse decisions are made in the three scenarios.

Table 3.2: Optimal capacity of each process in each plant

| $Q_{pi}$ (million lb/yr) | I1 | I2 | I3 | I4 |
|---|---|---|---|---|
| P1 | 150.0 | 0 | 0 | 4.9 |
| P2 | 95.4 | 95.6 | 0 | 5.9 |
| P3 | 57.5 | 36.1 | 0 | 5.5 |

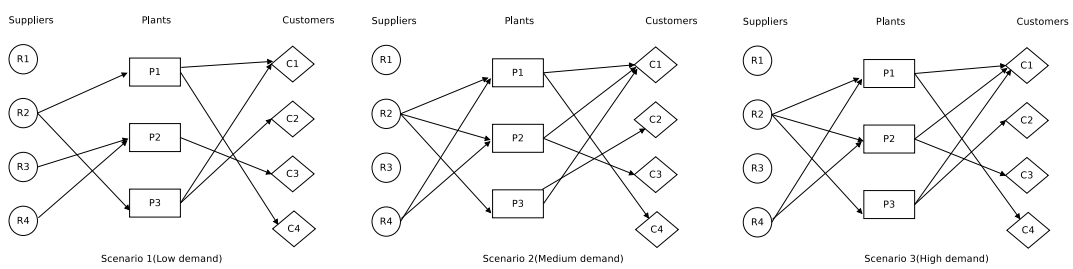


Figure 3.5: Optimal solution for transportation links

To illustrate the capacity and material flows of process networks, the optimal configuration and mass flows of plant 1 in scenario 1 is shown in Figure 3.6. In this process network, processes I1 and I4 are installed. Raw material J1 is purchased from the suppliers and used as feedstock for I1.S1. Chemical J3 is produced by I1.S1. Part of J3 is delivered to customers directly, while the rest is used as feedstock for I4.S1. Chemical J4 is purchased as raw material and used as feedstock for I4.S2. Chemical J5 is produced by I4.S1 and I4.S2 and

delivered to customers. Chemical J6 is produced by I4.S1 and I4.S2 and used as feedstock for I1.S1.

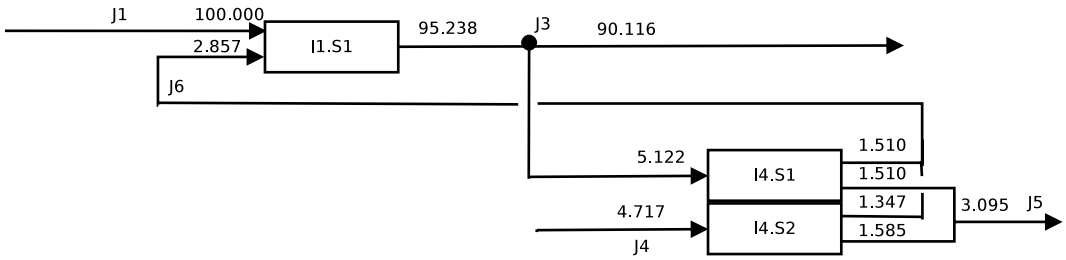


Figure 3.6: Optimal material flows for plant 1 in scenario 1

## 3.3 Computational results

In order to test the proposed algorithm, the two proposed examples are solved with increasing number of scenarios. For the batch plant design problem under demand uncertainty, the scenarios are constructed by assuming the demands of all the products can be high, medium or low. We also assume that some of the product demands are independent of all the other product demands while some of the products demands vary simultaneously. For example, in the 81-scenario case, we assume that the demands of product $i1$, $i2$, and $i3$ are independent of the demands of all the other products while the demands of product $i4$ and $i5$ vary simultaneously. So there are $3^4 = 81$ scenarios. For the planning problem under demand and price uncertainty, we have two products $C3$ and $C5$. In the 3-scenario problem, we assume that the demands of product $C3$ and $C5$ vary simultaneously and they can be high, medium, or low. In the problems with 9, 27, and 81 scenarios, we assume that the demands and the prices of all the products can be high, medium, or low. The demands are always assumed to be independent of the prices. The prices of the two products $C3$ and $C5$ vary simultaneously in the problem with 9 and 27 scenarios but they are assumed to be independent of each other in the 81-scenario problem. The demands of $C3$ and $C5$ are assumed to vary simultaneously in the 9-scenario problem but they are assumed to be independent of each other in the problems with 27 and 81 scenarios. Note that in practice

the discrete probability distribution can be constructed using historical data. If the computational resources are limited, the number of scenarios can be reduced using sample average approximation Kleywegt et al. (2002).

The deterministic equivalent of both the batch plant design problem and the planning problem with different number of scenarios are first solved using different convex MINLP solvers including DICOPT, AlphaECP, SBB, and BARON on the 12 processors of an Intel Xeon (2.67GHz) machine with 64 GB RAM. The time limit is set to 50,000 seconds. The size of the deterministic equivalent problems, the wall time that the solvers require to solve the problems, the upper bound, the lower bound, the expected result of using the expected value problem solution (EEV), and value of stochastic solution(VSS) are shown in Tables 3.3 and 3.4. It easy to see that the sizes of the problem can grow very quickly with the number of scenarios. SBB, Alpha-ECP, and DICOPT cannot obtain a feasible solution to the deterministic equivalent problems for the 243-scenario batch or the 81-scenario planning problem. Although BARON can obtain feasible solutions to the two large problems, the relative optimality gap is large, i.e., 5.9% and 22.3%, respectively.

Table 3.3: Computational statistics of the deterministic equivalent of the batch plant design problem

| Number of scenarios | 3 | 27 | 81 | 243 |
|---|---|---|---|---|
| Binary variables | 96 | 672 | 1,968 | 5,856 |
| Continuous variables | 64 | 472 | 1,390 | 4,144 |
| Constraints | 316 | 2,158 | 6,424 | 19,222 |
| Wall time(s) by BARON | 2 | 210 | 50,000 | $50,000^*$ |
| Wall time(s) by SBB | 4 | 99 | 9,602 | $50,000^*$ |
| Wall time(s) by Alpha-ECP | 7 | 1,551 | $50,000^*$ | $50,000^*$ |
| Wall time(s) by DICOPT | 1 | 35 | 503 | $50,000^*$ |
| NLP relaxation ($10^3$\$) | 408.4 | 394.5 | 391.7 | 391.0 |
| Best upper bound ($10^3$\$) | 423.5 | 415.9 | 413.6 | 435.0 |
| Best lower bound ($10^3$\$) | 423.5 | 415.9 | 413.6 | 410.9 |
| Relative Optimality gap | 0.0% | 0.0% | 0.0% | 5.9% |
| EEV ($10^3$\$) | 433.4 | 421.8 | 418.8 | 418.2 |
| VSS ($10^3$\$) | 9.9 | 5.9 | 5.2 | - |

*No feasible solution found in 50,000 secs time limit.

Table 3.4: Computational statistics of the deterministic equivalent of the planning problem

| Number of scenarios | 3 | 9 | 27 | 81 |
|---|---|---|---|---|
| Binary variables | 84 | 228 | 660 | 1,956 |
| Continuous variables | 2,158 | 6,424 | 19,222 | 57,616 |
| Constraints | 2,167 | 6,451 | 19,303 | 57,859 |
| Wall time(s) by BARON | 188 | 50,000 | $50,000$ | $50,000$ |
| Wall time(s) by SBB | $50,000^*$ | $50,000^*$ | $50,000^*$ | $50,000^*$ |
| Wall time(s) by Alpha-ECP | 462 | $50,000$ | $50,000^*$ | $50,000^*$ |
| Wall time(s) by DICOPT | 24 | $44,150$ | $50,000^*$ | $50,000^*$ |
| NLP-relaxation ($10^6$\$) | 1309.4 | 1309.2 | 1297.2 | 1297.2 |
| Best upper bound ($10^6$\$) | 1509.3 | 1508.8 | 1500.4 | 1770.7 |
| Best lower bound ($10^6$\$) | 1509.3 | 1508.8 | 1468.1 | 1448.0 |
| Relative optimality gap | 0.0% | 0.0% | 2.2% | 22.3% |
| EEV ($10^6$\$) | 1703.1 | 1702.6 | 1698.5 | 1698.5 |
| VSS ($10^6$\$) | 193.8 | 193.8 | 198.1 | - |

Since the deterministic equivalent problems with a large number of scenarios are intractable, we implement different decomposition algorithms in GAMS. For the batch plant design problem, we have a tight NLP relaxation (see Table 3.3). The effects of the lift-and-project cuts are not significant in terms of tightening the Benders subproblems. Therefore, for the batch plant design problem, we just solve the Benders subproblem without generating the lift-and-project cuts. Lagrangean decomposition and progressive hedging are also implemented. The multipliers of the Lagrangean decomposition algorithm are updated by the subgradient method shown in Appendix B. The $\rho$ parameter in the progressive hedging algorithm is updated dynamically using the heuristic proposed by Watson and Woodruff (2011). In the progressive hedging algorithm, the lower bounds can be derived by neglecting the quadratic terms in the objective of the progressive hedging subproblems and solving subproblems similar to the Lagrangean subproblems, which are proved to be valid lower bounds by Gade et al. (2016). The upper bounds of both the Lagrangean decomposition and the progressive hedging algorithm are derived by the nearest scenario heuristic, i.e., at each iteration, after the Lagrangean (progressive hedging) subproblems are solved to optimality,

we take the weighted sum average of the first stage decisions, find the scenario whose first stage decisions are the closest to the weighted sum average, fix the first stage decisions at the first stage decisions of this scenario, and solve the upper bound subproblems in parallel to obtain a feasible solution.

All the problems are solved using the 12 processors of an Intel Xeon (2.67GHz) machine with 64 GB RAM. All the MINLP subproblems are solved using DICOPT and all the NLP subproblems are solved using CONOPT. All the subproblems are solved in parallel using Grid GAMS (Bussieck et al., 2009). It is observed that the Lagrangean multipliers are close to convergence after the first 30 iterations. A similar trend hold for the progressive hedging algorithm. Therefore, we stop adding Lagrangean cuts after 30 iterations but keep adding Benders cuts until the lower bounds of the problem fail to improve for 10 iterations. The computational results of the decomposition algorithms for the batch problem are shown in Tables 3.5-3.8. L+B represents the algorithm with both Lagrangean cuts and Benders cuts in the Benders master problem. L represents the algorithm with only Lagrangean cuts in the Benders master problem. B represents the algorithm with only Benders cuts in the Benders master problem. LD represents the Lagrangean decomposition algorithm. PH represents the progressive hedging algorithm. The decomposition algorithms are all able to solve the problems within the time limit. For the batch problem with different number of scenarios, the proposed algorithm with both Lagrangean and Benders cuts can provide solutions with the smallest optimality gap. The lower bounds of the proposed algorithm are always tighter than the lower bounds given by LD and PH. In some cases, for example, the 3-scenario problem, LD or PH can give better feasible solutions than the proposed algorithm. However, in most cases (problems with 27, 81, and 243 scenarios), the best feasible solutions are given by the proposed algorithm. This comes at the expense of longer computational times. The CPU times of the subproblems and the master problems are also shown in Tables 3.5-3.8. Note that the CPU time of the Lagrangean subproblems is the summation of the CPU times of all the Lagrangean subproblems that are solved in parallel on multiple processors. Each Lagrangean subproblem is solved on only one processor. Therefore, the summation of the

CPU times of all the Lagrangean subproblems can be greater than the wall time. An example can be found in Table 3.6.

Table 3.5: Computational results of solving the 3-scenario batch plant design problem with different decomposition algorithms

| 3 scenario | L+B | L | B | LD | PH |
|---|---|---|---|---|---|
| Upper bound ($10^3$\$) | 423.7 | 433.4 | 423.6 | 423.5 | 423.5 |
| Lower bound ($10^3$\$) | 423.3 | 422.7 | 421.6 | 419.5 | 408.4 |
| Relative optimality gap | 0.09% | 2.51% | 0.46% | 0.96% | 3.70% |
| Wall time (s) | 53 | 27 | 26 | 13 | 42 |
| Upper bound subproblem (s) | 32 | 11 | 25 | 6 | 9 |
| Lagrangean subproblem (s) | 38 | 37 | - | 21 | 24 |
| Benders subproblem (s) | 1 | - | 1 | - | - |
| Benders master problem (s) | 10 | 3 | 5 | - | - |
| Progressive hedging (s) | - | - | - | - | 49 |

Table 3.6: Computational results of solving the 27-scenario batch plant design problem with different decomposition algorithms

| 27 scenario | L+B | L | B | LD | PH |
|---|---|---|---|---|---|
| Upper bound ($10^3$\$) | 415.9 | 421.8 | 415.9 | 416.2 | 416.1 |
| Lower bound ($10^3$\$) | 413.7 | 404.7 | 413.6 | 403.6 | 404.2 |
| Relative optimality gap | 0.53% | 4.21% | 0.56% | 3.12% | 2.96% |
| Wall time (s) | 228 | 66 | 110 | 37 | 56 |
| Upper bound subproblem (s) | 43 | 22 | 43 | 13 | 15 |
| Lagrangean subproblem (s) | 370 | 359 | - | 310 | 317 |
| Benders subproblem (s) | 7 | - | 7 | - | - |
| Benders master problem (s) | 134 | 15 | 52 | - | - |
| Progressive hedging (s) | - | - | - | - | 334 |

The sizes of subproblems are shown in Table 3.9. The size of the Benders master problem corresponds to the size of the 243-scenario problem at the last iteration (59th iteration in this case). Each subproblem is comparatively easier to solve than the deterministic equivalent problems.

For the planning problem, it is shown in Table 3.4 that the NLP relaxation is weaker. Therefore, using the lift-and-project cuts to tighten the Benders subproblem should be effec-

Table 3.7: Computational results of solving the 81-scenario batch plant design problem with different decomposition algorithms

| 81 scenario | L+B | L | B | LD | PH |
|---|---|---|---|---|---|
| Upper bound ($10^3$\$) | 413.6 | 418.8 | 413.6 | 413.8 | 413.8 |
| Lower bound ($10^3$\$) | 411.4 | 402.9 | 411.3 | 402.6 | 403.2 |
| Relative optimality gap | 0.55% | 3.96% | 0.56% | 2.78% | 2.64% |
| Wall time (s) | 574 | 146 | 352 | 86 | 129 |
| Upper bound subproblem (s) | 137 | 72 | 132 | 42 | 41 |
| Lagrangean subproblem (s) | 1015 | 1002 | - | 1018 | 913 |
| Benders subproblem (s) | 21 | - | 19 | - | - |
| Benders master problem (s) | 319 | 32 | 183 | - | - |
| Progressive hedging (s) | - | - | - | - | 967 |

Table 3.8: Computational results of solving the 243-scenario batch plant design problem with different decomposition algorithms

| 243 scenario | L+B | L | B | LD | PH |
|---|---|---|---|---|---|
| Upper bound ($10^3$\$) | 413.1 | 418.2 | 413.1 | 413.1 | 413.3 |
| Lower bound ($10^3$\$) | 410.8 | 402.5 | 410.8 | 402.4 | 402.9 |
| Relative optimality gap | 0.55% | 3.89% | 0.56% | 2.68% | 2.58% |
| Wall time (s) | 1962 | 490 | 609 | 270 | 376 |
| Upper bound subproblem (s) | 416 | 208 | 412 | 129 | 127 |
| Lagrangean subproblem (s) | 2557 | 2522 | - | 2657 | 2508 |
| Benders subproblem (s) | 64 | - | 56 | - | - |
| Benders master problem (s) | 1132 | 153 | 1310 | - | - |
| Progressive hedging (s) | - | - | - | - | 2621 |

Table 3.9: Size of each subproblem for the batch plant design problem

| | Binary variables | Continuous variables | Constraints |
|---|---|---|---|
| Benders master problem | 24 | 16 | 21,895 |
| Upper bound subproblem | 24 | 54 | 116 |
| Benders subproblem | 0 | 78 | 116 |
| Lagrangean subproblem | 48 | 30 | 104 |

tive. Different decomposition algorithms are implemented and the computational results are shown in Tables 3.10-3.13. SB+L represents the algorithm with both strengthened Benders

cuts and Lagrangean cuts in the Benders master problem. SB represents the algorithm with only strengthened Benders cuts in the Benders master problem. Besides that, L, L+B, LD, and PH are also implemented as in the batch problem. All the decomposition algorithms are able to solve the planning problem within the time limit. Among all the algorithms, SB+L gives the smallest relative optimality gap in most cases. The algorithm with strengthened Benders cuts can give tighter optimality gap than the algorithm use Benders cuts. It can be observed that in most cases the strengthened Benders cuts can close the optimality gap of the algorithm that uses Benders cuts by more than one half. The algorithms that use strengthened Benders cuts or Lagrangean cuts always yield a tighter lower bound than the Lagrangean decomposition or the progressive hedging algorithm. SB+L uses less time than SB in all the cases because adding Lagrangean cuts reduces the number of iterations to converge. The computational time of SB+L is significantly higher than L+B, since we have to solve a large number of cut generating linear programs. It is shown in Tables 3.10-3.13 that the CPU time of the cut generating linear programs is small compared to other subproblems. Therefore, the significant increase in wall time is due to the data overhead of GAMS. If we can implement SB+L in languages like C++, we could expect the wall time to be much smaller. Note that in some cases, LD is able to give good feasible solutions. However, the lower bound of LD is weaker than SB+L. In practice, if the decision-maker is not limited by computational resources, our proposed algorithm, SB+L, can provide the tightest lower bound and good feasible solutions. Otherwise, the decision-maker can also use Lagrangean decomposition to obtain good feasible solutions.

Table 3.10: Computational results of solving the 3-scenario planning problem with different decomposition algorithms

| 3 scenario | SB+L | L | SB | L+B | B | LD | PH |
|---|---|---|---|---|---|---|---|
| Upper bound ($10^6$\$) | 1523.0 | 1703.1 | 1515.3 | 1537.4 | 1516.6 | 1532.8 | 1703.1 |
| Lower bound ($10^6$\$) | 1477.7 | 1472.1 | 1470.7 | 1472.4 | 1438.4 | 1471.9 | 1452.9 |
| Relative optimality gap | 3.07% | 15.69% | 3.03% | 4.42% | 5.44% | 4.14% | 17.22% |
| Wall time (s) | 404 | 76 | 408 | 100 | 68 | 66 | 1384 |
| Upper bound subproblem (s) | 65 | 17 | 87 | 39 | 66 | 9 | 8 |
| Lagrangean subproblem (s) | 175 | 108 | - | 102 | - | 119 | 103 |
| Benders subproblem (s) | - | - | - | 6 | 7 | - | - |
| Benders master problem (s) | 35 | 6 | 30 | 15 | 18 | - | - |
| Cut generating linear program (s) | 112 | - | 156 | - | - | - | - |
| Strengthened Benders subproblem (s) | 54 | - | 73 | - | - | - | - |
| Progressive Hedging (s) | - | - | - | - | - | - | 1770 |

Table 3.11: Computational results of solving the 9-scenario planning problem with different decomposition algorithms

| 9 scenario | SB+L | L | SB | L+B | B | LD | PH |
|---|---|---|---|---|---|---|---|
| Upper bound ($10^6$\$) | 1514.7 | 1702.6 | 1512.4 | 1528.6 | 1517.5 | 1545.5 | 1702.6 |
| Lower bound ($10^6$\$) | 1473.2 | 1439.3 | 1470.8 | 1456.5 | 1438.3 | 1438.0 | 1440.5 |
| Relative optimality gap | 2.82% | 18.30% | 2.83% | 4.95% | 5.51% | 7.48% | 18.19% |
| Wall time (s) | 771 | 135 | 942 | 220 | 123 | 111 | 1695 |
| Upper bound subproblem (s) | 197 | 52 | 271 | 111 | 210 | 27 | 25 |
| Lagrangean subproblem (s) | 695 | 446 | - | 471 | - | 427 | 308 |
| Benders subproblem (s) | - | - | - | 20 | 37 | - | - |
| Benders master problem (s) | 62 | 7 | 96 | 57 | 51 | - | - |
| Cut generating linear program (s) | 394 | - | 506 | - | - | - | - |
| Strengthened Benders subproblem (s) | 174 | - | 243 | - | - | - | - |
| Progressive Hedging (s) | - | - | - | - | - | - | 4377 |

The sizes of the subproblems and the master problem of the SB+L algorithm for the planning problem are shown in Table 3.14. The size of the Benders master problem and the strengthened Benders subproblem is the size of the 81-scenario problem in the last iteration (iteration 92). Note that the strengthened Benders subproblem can have as many as 2,208 more constraints than the Benders subproblem in the last iteration. Those correspond to the rank-one lift-and-project cuts that are generated.

Table 3.12: Computational results of solving the 27-scenario planning problem with different decomposition algorithms

| 27 scenario | SB+L | L | SB | L+B | B | LD | PH |
|---|---|---|---|---|---|---|---|
| Upper bound ($10^6$\$) | 1504.4 | 1698.5 | 1509.7 | 1516.0 | 1557.8 | 1509.0 | 1521.7 |
| Lower bound ($10^6$\$) | 1462.0 | 1422.2 | 1460.4 | 1441.7 | 1418.5 | 1420.7 | 1425.3 |
| Relative optimality gap | 2.90% | 19.43% | 3.38% | 5.16% | 9.82% | 6.22% | 6.76% |
| Wall time (s) | 2115 | 234 | 2434 | 1448 | 156 | 209 | 2593 |
| Upper bound subproblem (s) | 577 | 5 | 757 | 27423 | 121 | 99 | 93 |
| Lagrangean subproblem (s) | 2114 | 2141 | - | 2123 | - | 2047 | 1384 |
| Benders subproblem (s) | - | - | - | 67 | 24 | - | - |
| Benders master problem (s) | 167 | 23 | 204 | 156 | 68 | - | - |
| Cut generating linear program (s) | 959 | - | 1311 | - | - | - | - |
| Strengthened Benders subproblem (s) | 404 | - | 584 | - | - | - | - |
| Progressive Hedging (s) | - | - | - | - | - | - | 18827 |

Table 3.13: Computational results of solving the 81-scenario planning problem with different decomposition algorithms

| 81 scenario | SB+L | L | SB | L+B | B | LD | PH |
|---|---|---|---|---|---|---|---|
| Upper bound ($10^6$\$) | 1502.8 | 1698.5 | 1505.5 | 1512.7 | 1553.1 | 1509.3 | 1507.2 |
| Lower bound ($10^6$\$) | 1461.2 | 1418.5 | 1460.4 | 1441.5 | 1419.0 | 1416.7 | 1423.7 |
| Relative optimality gap | 2.85% | 19.74% | 3.09% | 4.94% | 9.45% | 6.53% | 5.87% |
| Wall time (s) | 5986 | 555 | 10437 | 1239 | 729 | 348 | 4912 |
| Upper bound subproblem (s) | 1155 | 21 | 1690 | 1476 | 1589 | 260 | 362 |
| Lagrangean subproblem (s) | 8835 | 10100 | - | 10476 | - | 8164 | 5227 |
| Benders subproblem (s) | - | - | - | 222 | 138 | - | - |
| Benders master problem (s) | 372 | 80 | 604 | 570 | 443 | - | - |
| Cut generating linear program (s) | 2412 | - | 3506 | - | - | - | - |
| Strengthened Benders subproblem (s) | 975 | - | 2597 | - | - | - | - |
| Progressive Hedging (s) | - | - | - | - | - | - | 125690 |

## 3.4 Conclusions

In this chapter, we have proposed an improved L-shaped method that can solve two-stage convex 0-1 mixed-integer nonlinear stochastic programs with mixed-integer variables in both first and second stage effectively. The proposed algorithm is a Benders-like decomposition algorithm that includes (strengthened) Benders cuts and Lagrangean cuts in the Benders

Table 3.14: Size of each subproblem for the planning problem

| | Binary variables | Continuous variables | Constraints |
|---|---|---|---|
| Benders master problem | 12 | 28 | 9,907 |
| Upper bound subproblem | 24 | 748 | 763 |
| Benders subproblem | 0 | 772 | 763 |
| Strengthened Benders subproblem | 0 | 772 | 2,971 |
| Cut generating linear program | 0 | 2,387 | 4,639 |
| Lagrangean subproblem | 36 | 736 | 739 |

master problem. The Benders cuts are derived by solving the Benders subproblems, which are the NLP relaxations of the subproblems for each scenario after the first stage decisions are fixed at the optimal solution of the Benders master problem. The strengthened Benders cut are derived by adding rank-one lift-and-project cuts to the Benders subproblem. The Lagrangean cuts are derived by solving the Lagrangean subproblems whose multipliers are updated by a new subgradient method.

Different decomposition algorithms including SB+L, L, SB, L+B, B, LD, PH are implemented and tested using the planning problem under demand and price uncertainty, and the batch plant design problem under demand uncertainty. It is shown that the combination of Lagrangean cuts and strengthened Benders cuts is able to provide the tightest lower bound among all the decomposition algorithms that we test although it requires more time than algorithms like LD. The proposed algorithm can be used as an effective computational strategy to solve problems of the same type in PSE applications. However, computational results on more examples would be desirable to show the performance of the algorithm. Moreover, the proposed algorithm is not guaranteed to close the optimality gap of the problems due to the duality gap of the Lagrangean cuts and the integrality gap of the strengthened Benders cuts. Therefore, a more rigorous algorithm that has finite convergence would also be desirable.

# Chapter 4

# A Finite $\epsilon$-convergence Algorithm for Two-Stage Stochastic Convex Nonlinear Programs with Mixed-Binary First and Second-Stage Variables

In this chapter, we propose a GBD-based algorithm that can solve two-stage convex mixed-binary nonlinear stochastic programs with mixed-binary first and second-stage variables with finite $\epsilon$-convergence. This work is inspired by Sherali and Zhu Sherali and Zhu (2006) who propose a Benders decomposition algorithm to solve two-stage mixed-binary linear stochastic programs having mixed-binary first and second-stage variables. The authors use reformulation linearization technique (RLT) to convexify the original MILP subproblems.

However, the algorithm proposed by Sherali and Zhu Sherali and Zhu (2006) cannot be applied to convex nonlinear problems directly. Due to recent advances in generalized disjunctive programming (GDP), especially the work of Ruiz and Grossmann Ruiz and Grossmann (2012), we now have a procedure to obtain the convex hull of the feasible region of a convex

MINLP. By adapting the convexification scheme proposed in Ruiz and Grossmann (2012) to two-stage stochastic programs, we are able to develop a GBD-based algorithm with finite $\epsilon$-convergence. The proof of convergence is more general and different than the convergence proof by Sherali and Zhu Sherali and Zhu (2006). We also propose a sequential convexification scheme to reduce the sizes of the subproblems. To the best of our knowledge, the proposed algorithm is the first GBD-based algorithm for this type of problem.

## 4.1 Background

In this section, we provide some background on disjunctive convex programming and basic steps used in Ruiz and Grossmann (2012) to construct the convex hull of the MINLP subproblem in closed-form.

A convex set $C$ can be defined as $C$=$\{x \in \mathbb{R}^n | \phi(x) \leq 0\}$, where $\phi(x) : \mathbb{R}^n \rightarrow \mathbb{R}^1$ is a convex function. Given a collection of sets such that $C_j$=$\{x \in \mathbb{R}^n | \phi_j(x) \leq 0\}$, $j \in M$, the union of the convex sets, which is an elementary disjunctive set, is defined as:

$$H = \bigcup_{j \in M} C_j = \{x \in \mathbb{R}^n | \bigvee_{j \in M} \phi_j(x) \leq 0\} \tag{4.1}$$

and the intersection of convex sets can be expressed as:

$$P = \bigcap_{j \in M} C_j = \{x \in \mathbb{R}^n | \bigwedge_{j \in M} \phi_j(x) \leq 0\} \tag{4.2}$$

A disjunctive set can be expressed in different forms. Two extreme cases are the conjunctive normal form (CNF) and the disjunctive normal form (DNF). The CNF is the intersection of some elementary disjunctive sets:

$$F_{CNF} = \bigcap_{i \in T} H_i \tag{4.3}$$

where each $H_i$ is an elementary disjunctive set. The DNF is the union of some convex sets:

$$F_{DNF} = \bigcup_{i \in D} P_i \tag{4.4}$$

where each set $P_i$ is a convex set defined by convex functions $\phi_j$, $j \in M_i$. $P_i$ can also be represented in a more succinct form $P_i = \{x \in \mathbb{R}^n | g_i(x) \leq 0\}$, where $g_i(x)$: $\mathbb{R}^n \rightarrow \mathbb{R}^m$.

A regular form (RF) between the CNF and DNF can be represented as:

$$F_{RF} = \bigcap_{k \in K} S_k \tag{4.5}$$

where $S_k = \cup_{i \in D_k} P_i$. Each $S_k$ is called a disjunction, which is over some convex sets $P_i$, $i \in D_k$. Each $P_i$, $i \in D_k$, is an element of disjunction $k$ and is called a disjunct. Balas Balas (1985) proposes an operation called basic step, which when applied to the regular form can reduce the number of conjuncts by one in the linear case. The basic step is extended by Ruiz and Grossmann Ruiz and Grossmann (2012) to convex nonlinear case. Here, we rewrite Theorem 2.1 in Ruiz and Grossmann (2012), which is an extension of the theorem in Balas Balas (1985) to show how a CNF can be transformed into a DNF.

**Theorem 1.** Let $F_{RF}$ be a disjunctive set in regular form. Then $F_{RF}$ can be brought to DNF by $|K| - 1$ recursive applications of the following basic step which preserves regularity:

For some $r, s \in K$, bring $S_r \cap S_s$ to DNF by replacing it with:

$$S_{rs} = \bigcup_{i \in D_r, j \in D_s} (P_i \cap P_j)$$

Theorem 1 will be used to construct the convex hull the MINLP subproblem in our proposed algorithm.

## 4.2 Overview of basic ideas

The goal of this chapter is to design a decomposition algorithm that has finite $\epsilon$-convergence for two-stage convex mixed-binary nonlinear stochastic programs with mixed-binary first and second-stage variables. While GBD cannot be applied to problems with mixed-binary recourse directly, it has finite $\epsilon$-convergence for problems with convex constraints and continuous recourse variables as proved by Geoffrion Geoffrion (1972). Therefore, we propose to construct the convex hull for the integer-constrained subproblems by applying basic steps so that GBD can be applied to solve the relaxed problem. However, the relaxed problem is not equivalent to the original problem if we have mixed-binary first-stage variables. Therefore, we first prove that the relaxed problem is equivalent to the original full-space problem for the special case of pure binary first-stage variables. Based on the theoretical results of the

problems with pure binary variables, we propose a GBD-based branch and bound algorithm for the general problems with mixed-binary first and second-stage variables where we may branch on the continuous first-stage variables. The details of the algorithm are described in section 4.3.

Constructing the convex hull by applying all the possible basic steps for each subproblem can be expensive, especially when we have a large number of binary variables in the second stage. Therefore, in section 4.4, we describe a sequential convexification scheme, which is a more limited way for convexifying the subproblem to avoid applying all the basic steps when certain sufficient conditions are satisfied. Computational results of the proposed GBDBAB algorithm are shown in section 4.5 with one small illustrative example and two process systems engineering applications.

## 4.3 GBD-based branch and bound algorithm

In this chapter, we consider two-stage convex mixed-binary nonlinear stochastic programs with mixed-binary first and second-stage variables. The problem is formally defined by (4.6a)-(4.6e).

$$(P) \quad \min \quad c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{4.6a}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{4.6b}$$

$$A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{4.6c}$$

$$x \in X, \quad X = \left\{ x : x_i \in \{0,1\}, \forall i \in I_1, \ 0 \leq x \leq x^{ub} \right\} \tag{4.6d}$$

$$y_\omega \in Y \quad \forall \omega \in \Omega, \quad Y = \left\{ y : y_j \in \{0,1\}, \forall j \in J_1, \ 0 \leq y \leq y^{ub} \right\} \tag{4.6e}$$

Here, $x$ represents the first-stage decisions. $y_\omega$ represents the second-stage decisions in scenario $\omega$. $g_0$, $g_{1,\omega}$, are smooth convex functions. Note that the convex functions include linear functions. Both the first and the second-stage decisions are mixed-binary. Let $I = \{1, 2, \cdots, n\}$ be the index set of all the first-stage variables. Set $I_1 \subseteq I$ is the subset for indices of the binary first-stage variables. Let $J = \{1, 2, \cdots, m\}$ be the index set of all the second-stage variables. Set $J_1 \subseteq J$ is the subset for the indices of the binary second-stage

variables. Vector $x^{ub}$ represents the upper bound of all the first-stage variables. Vector $y^{ub}$ represents the upper bound of all the second-stage variables. In this chapter, we make the following assumptions about the problem $(P)$. The readers will see that these assumptions are used to prove the convergence of the proposed algorithm.

**Assumption 1.** Problem $(P)$ has relatively complete recourse, i.e., any solution $x$ that satisfies the first stage constraints has feasible recourse decisions in the second stage.

**Assumption 2.** The feasible region of $(P)$ is compact.

**Assumption 3.** $x^{ub}$ and $y^{ub}$ are finite, i.e., both the first and the second-stage decisions are bounded.

As $(P)$ has mixed-binary recourse, generalized Benders decomposition (GBD) (Geoffrion, 1972) cannot be applied to solve it directly as in the case of convex continuous recourse. Therefore, we construct convex relaxations of $(P)$ so that GBD can be applied to solve the problem.

We first define the set,

$$S_\omega = \left\{(x, y_\omega | A_{1,\omega}x + g_{1,\omega}(y_\omega) \leq b_{1,\omega}, y_\omega \in Y, 0 \leq x \leq x^{ub}\right\} \tag{4.7}$$

where the second stage constraints and the upper and lower bound constraints for the first-stage decisions are included for each scenario $\omega$. The full-space problem with $(x, y_\omega)$ in the convex hull of $S_\omega$ for all $\omega \in \Omega$ is defined as $(PC)$:

$$(PC) \quad \min \quad c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{4.8a}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{4.8b}$$

$$x \in X \tag{4.8c}$$

$$(x, y_\omega) \in conv(S_\omega) \quad \forall \omega \in \Omega \tag{4.8d}$$

Since the integrality constraints in the second stage are relaxed, $(PC)$ is a relaxation of $(P)$. Specifically, $conv(S_\omega)$ can be characterized by applying the hierarchy of relaxations for

nonlinear convex generalized disjunctive programs proposed by Ruiz and Grossmann (2012). As a subset of the second-stage variables $y_\omega$ are binary, i.e., $(y_\omega)_j \in \{0,1\}$, $\forall j \in J_1$, the set $S_\omega$ is equivalent to the disjunctions in (4.9).

$$\begin{bmatrix} A_{1,\omega}x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \\ 0 \leq x \leq x^{ub} \\ 0 \leq y_\omega \leq y^{ub} \\ (y_\omega)_j = 1 \end{bmatrix} \vee \begin{bmatrix} A_{1,\omega}x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \\ 0 \leq x \leq x^{ub} \\ 0 \leq y_\omega \leq y^{ub} \\ (y_\omega)_j = 0 \end{bmatrix} \quad \forall j \in J_1 \tag{4.9}$$

We represent (4.9) as $S^1_{\omega j} \cup S^0_{\omega j}$, $\forall j \in J_1$. Here, we slightly abuse notation and call equation (4.9) a conjunctive normal form (CNF) representation of $S_\omega$ where $S_\omega$ is represented as a conjunction over the disjunctions, i.e., $S_\omega = \cap_{j \in J_1}(S^1_{\omega j} \cup S^0_{\omega j})$. Note that the exact definition of CNF is shown in (4.3). The hull relaxation of the CNF (4.9), denoted by $h\text{-}rel\big(\cap_{j \in J_1}(S^1_{\omega j} \cup S^0_{\omega j})\big)$, is equivalent to $\cap_{j \in J_1} conv(S^1_{\omega j} \cup S^0_{\omega j})$. We know that $\cap_{j \in J_1} conv(S^1_{\omega j} \cup S^0_{\omega j}) \subseteq conv(\cap_{j \in J_1} S^1_{\omega j} \cup S^0_{\omega j}) = conv(S_\omega)$. Therefore, the hull relaxation of the CNF is a relaxation of $conv(S_\omega)$. In order to construct $conv(S_\omega)$, we apply the basic step proposed by Balas (1985) to the CNF (4.9). Based on the Theorem 1, the CNF can be converted to the following DNF shown in (4.10),

$$\vee_{r \in R} \begin{bmatrix} A_{1,\omega}x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \\ 0 \leq x \leq x^{ub} \\ 0 \leq y_\omega \leq y^{ub} \\ (y_\omega)_j = e_{rj} \quad \forall j \in J_1 \end{bmatrix} \tag{4.10}$$

where $R$ is the set that corresponds to all the possible combinations of the binary variables $(y_\omega)_j$, $\forall j \in J_1$. For each $r \in R$, $e_{rj}$ can be either zero or one. There are $2^{|J_1|}$ components in the set $R$. Ceria and Soares (1999) proved that the convex hull of the DNF (4.10) can be expressed as the projection of a higher dimensional convex set where $(x, y_\omega)$ has to satisfy

the following constraints:

$$
\begin{aligned}
& x = \sum_{r \in R} u_\omega^r \\
& y_\omega = \sum_{r \in R} v_\omega^r \\
& \sum_{r \in R} \gamma_\omega^r = 1, \quad 0 \le \gamma_\omega^r \le 1, \quad \forall r \in R \\
& A_{1,\omega} u_\omega^r + \gamma_\omega^r g_{1,\omega}(v_\omega^r/\gamma_\omega^r) \le b_{1,\omega}\gamma_\omega^r, \quad \forall r \in R \\
& 0 \le u_\omega^r \le x^{ub}\gamma_\omega^r, \quad \forall r \in R \\
& 0 \le v_\omega^r \le y^{ub}\gamma_\omega^r, \quad \forall r \in R \\
& (v_\omega)_j = e_{rj}\gamma_\omega^r \quad \forall j \in J_1, r \in R
\end{aligned}
\tag{4.11}
$$

where the perspective function of $g_{1,\omega}$, $\gamma_\omega^r g_{1,\omega}(v_\omega^r/\gamma_\omega^r)$, has to be used. According to Ruiz and Grossmann (2012), $(x, y_\omega)$ is in $conv(S_\omega)$ if and only if it satisfies the constraints in (4.11). Since the perspective function of $g_{1,\omega}$ used in (4.11) can give rise to numerical difficulties at $\gamma_\omega^r = 0$, the approximation proposed by Furman et al. (2016) is used in our implementation,

$$
\gamma_\omega^r g_{1,\omega}(v_\omega^r/\gamma_\omega^r) \approx \big((1-\kappa)\gamma_\omega^r + \kappa\big) g_{1,\omega}\Big(\frac{v_\omega^r}{(1-\kappa)\gamma_\omega^r + \kappa}\Big) - \kappa g_{1,\omega}(0)(1-\gamma_\omega^r) \tag{4.12}
$$

where $\kappa$ is a small number like $10^{-5}$. We should note that this approximation is exact at $\gamma_\omega^r = 0$ and $\gamma_\omega^r = 1$. As we are able to construct $conv(S_\omega)$ for all $\omega \in \Omega$ in closed-form, GBD can be applied to solve $(PC)$ to $\epsilon$-optimality.

We briefly describe how GBD can be used to solve $(PC)$, the reader can refer to Geoffrion (1972) for details. The basic idea of GBD is to decompose $(P)$ into a Benders master problem which only contains first-stage decisions and Benders subproblems each of which only contains second-stage decisions for a given scenario.

We first define the Benders master problem $(MB^h)$ at iteration $h$ of the GBD algorithm. Equation (4.13c) are Benders cuts that are generated up until iteration $h$, which are derived by solving the Benders subproblems $(SB_\omega^h)$. Benders master problem is a relaxation of $(PC)$ projected on the $x$ space. Therefore, a lower bound of $(PC)$ is obtained by solving the

MINLP Benders master problem:

$$(MB^h): \quad \min \quad z^h_{MB} = \sum_{\omega} \eta_\omega \tag{4.13a}$$

$$s.t. \quad A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{4.13b}$$

$$\eta_\omega \geq z^{*h'}_{SB,\omega} + (\lambda^{h'}_\omega)^T (x - \tilde{x}^{h'}) + \tau_\omega c^T x \quad \forall \omega \in \Omega, h' \leq h \tag{4.13c}$$

$$x \in X, \quad \eta_\omega \in \mathbb{R}^1, \quad \forall \omega \in \Omega \tag{4.13d}$$

Assume that the solution to the Benders master problem at iteration $h$ is $\tilde{x}^h$, $x$ is fixed at $\tilde{x}^h$ in $(PC)$. The rest of the problem can be decomposed into Benders subproblems $(SB^h_\omega)$.

$$(SB^h_\omega): \quad \min \quad z^h_{SB,\omega} = \tau_\omega d^T_\omega y_\omega \tag{4.14a}$$

$$s.t. \quad x = \tilde{x}^h \tag{4.14b}$$

$$(x, y_\omega) \in conv(S_\omega) \tag{4.14c}$$

A Benders cut can be generated at iteration $h$:

$$\eta_\omega \geq z^{*h}_{SB,\omega} + (\lambda^h_\omega)^T (x - \tilde{x}^h) + \tau_\omega c^T x \tag{4.15}$$

where $\lambda^h_\omega$ are the optimal dual multipliers for constraints (4.14b), $z^{*h}_{SB,\omega}$ is the optimal objective value of $(SB^h_\omega)$. A feasible solution to $(PC)$ is obtained at the optimal solution of the Benders subproblem. After solving the Benders subproblems at iteration $h$, GBD will add the newly generated Benders cuts to the Benders master problem and keep iterating between the Benders master problem and the Benders subproblems until $(PC)$ is solved to $\epsilon$-optimality.

Although GBD can be applied to solve $(PC)$, for the problem with both binary and continuous first-stage variables, Sherali and Zhu (2006) showed that in the MILP setting $(PC)$ is not always equivalent to $(P)$, since for fixed first stage decision $\bar{x}$, the extreme points of $conv(S_\omega) \cap \{(x, y) : x = \bar{x}\}$ will not always have binary values for $y_j$, $\forall j \in J_1$. As convex MINLP is a generalization of MILP, it is straightforward to show that $(PC)$ is not always equivalent to $(P)$.

However, it is also shown in Sherali and Zhu (2006) that in the MILP setting, under the restriction of pure binary first-stage variables, $(PC)$ and $(P)$ are equivalent. Therefore, before we consider the proposed GBD-based algorithm to solve problem $(P)$ with both binary and continuous first-stage variables, the equivalence of $(P)$ and the corresponding $(PC)$ with pure binary first-stage variables is proved in Proposition 1. The convergence of the GBD-based algorithm for the problem with mixed-binary first-stage variables will be proved later based on Proposition 1 and Corollary 1.

**Proposition 1.** Consider a special case of $(P)$ where the first-stage variables are all binary. We assume that in the corresponding problem $(PC)$, the convex hull of $S_\omega$ is expressed in the form of (4.11). Then $(P)$ and $(PC)$ are equivalent in the sense that they have the same optimal objective value and the optimal solution of $(P)$ can always be obtained based on the optimal solution of $(PC)$.

*Proof.* Note that $(PC)$ is a relaxation of $(P)$. Hence the optimal objective value of $(PC)$ should be less than or equal to $(P)$. Let $(x^*, y^*_{\omega_1}, y^*_{\omega_2}, \cdots, y^*_{\omega_{|\Omega|}})$ be the optimal solution of $(PC)$. Now if we select any scenario $\omega$, we have $(x^*, y^*_\omega) \in conv(S_\omega)$. For this scenario $\omega$, we have $x^* = \sum_{r\in R} u^{r*}_\omega$ and $y^*_\omega = \sum_{r\in R} v^{r*}_\omega$. As we have assumed that $x$ are pure binary variables and we have $0 \le u^{r*}_\omega \le e\gamma^{r*}_\omega$, for any $(x^*)_i = 1$, we must have $(u^{r*}_\omega)_i = \gamma^{r*}_\omega$, otherwise $\sum_{r\in R}(u^{r*}_\omega)_i$ will not sum up to 1. For any $(x^*)_i = 0$, $(u^{r*}_\omega)_i$ must be zero for all $r$. Therefore, for all $\gamma^{r*}_\omega > 0$, we have $u^{r*}_\omega/\gamma^{r*}_\omega = x^*$. Furthermore, for all $\gamma^{r*}_\omega > 0$, $d^T v^{r*}_\omega/\gamma^{r*}_\omega = d^T y^*_\omega$. Otherwise, there must exist $r'$ and $r''$ such that $d^T v^{r'*}_\omega/\gamma^{r'*}_\omega > d^T y^*_\omega$ and $d^T v^{r''*}_\omega/\gamma^{r''*}_\omega < d^T y^*_\omega$. Then $v^{r''*}_\omega/\gamma^{r''*}_\omega$ satisfies all the constraints of $(PC)$ and leads to a lower objective value than $y^*_\omega$, which contradicts with the fact that $y^*_\omega$ is the optimal solution. Therefore, we can select any $\gamma^{r*}_\omega > 0$, and let $y_\omega = v^{r*}_\omega/\gamma^{r*}_\omega$ to construct a feasible solution that yields the same cost in the objective as $y^*_\omega$. We can follow the same procedure to construct the optimal solution of every scenario $\omega \in \Omega$. Note that the solution that we construct satisfies all the constraints of $(P)$ and yields the same objective value as the optimal solution to $(PC)$. In that sense, $(PC)$ and $(P)$ are equivalent. $\square$

The following theorem can be proved based on Proposition 1.

**Theorem 2.** GBD can be applied to obtain the $\epsilon$-optimal solution of problem $(P)$ with pure binary first-stage variables by solving its relaxation $(PC)$.

*Proof.* As we have assumed that the feasible region of $(P)$ is nonempty compact and the problem has relatively complete recourse, $conv(S_\omega)$ is nonempty compact convex set for any fixed $x$, s.t. $x \in X$, $A_0 x \geq b_0$, $g_0(x) \leq 0$, for every $\omega \in \Omega$. It follows directly from Theorem 2.5 of Geoffrion (1972) that GBD has finite $\epsilon$-convergence when applied to solve $(PC)$. It is proved in Proposition 1 that $(P)$ and $(PC)$ are equivalent for the problem with pure binary first-stage variables. Therefore, GBD can be applied to obtain the $\epsilon$-optimal solution of problem $(P)$ with pure binary first-stage variables by solving its relaxation $(PC)$. □

Now we go back to focus on the problem $(P)$ with both binary and continuous first-stage variables. A corollary can be derived based on Proposition 1.

**Corollary 1.** For (PC) with both binary and continuous first-stage variables, if the optimal first-stage variables $x^*$ to $(PC)$ are all at their upper or lower bound, i.e., $(x^*)_i = 0$ or $(x^*)_i = (x^{ub})_i$, $\forall i \in I$, then $(PC)$ and its corresponding (P) are equivalent in the sense that they have the same optimal objective value.

*Proof.* The proof is similar to the case where we have pure binary first-stage variables. As $(x^*)_i = 0$ or $(x^*)_i = (x^{ub})_i$, $\forall i$, we must have $u_\omega^{r*}/\gamma_\omega^r = x^*$ for all $\gamma_\omega^{r*} > 0$. Therefore, we can construct a feasible solution to $(P)$ by letting $y_\omega = v_\omega^{r*}/\gamma_\omega^{r*}$ where $\gamma_\omega^{r*} > 0$ for all $\omega \in \Omega$. The feasible solution yields the same objective value as the optimal solution to $(PC)$. □

If the condition of Corollary 1 does not hold, the equivalence of $(PC)$ and $(P)$ cannot be guaranteed as we have discussed. In order to understand why $(PC)$ and $(P)$ are not equivalent in the general case, we can determine if the proof of Proposition 1 still holds for the problem with mixed-binary first-stage variables. As we construct the convex hull of each scenario separately, if we take $u_\omega^{r*}/\gamma_\omega^{r*}$ for some $\gamma_\omega^{r*} > 0$ for each scenario $\omega$ as we did in Proposition 1 and Corollary 1, it is possible that we fail to obtain a unique first-stage decision for all the scenarios if some $(x^*)_i$ is not at its lower or upper bound. The reason why this occurs is that for component $i$ of $x^*$ such that $(x)_i$ does not lie at its upper or

lower bound, there may exist one scenario $\omega$ for which we have $(x^*)_i = \sum_{r\in R}(u_\omega^{r*})_i$ but there exist $r', r''$ with $\gamma_\omega^{r'*} > 0$, $\gamma_\omega^{r''*} > 0$, $(u_\omega^{r'*}/\gamma_\omega^{r'*})_i < (x^*)_i$ , $(u_\omega^{r''*}/\gamma_\omega^{r''*})_i > (x^*)_i$. As $(u_\omega^{r*}/\gamma_\omega^{r*})_i$, $\gamma_\omega^{r*} > 0$, is not always equal to $(x^*)_i$, the uniqueness of $x$ obtained by taking $u_\omega^{r*}/\gamma_\omega^{r*}$ for some $\gamma_\omega^{r*} > 0$ for each scenario $\omega$ cannot be guaranteed. However, if we branch on the continuous first-stage variables $x$, for example, by forcing $(x)_i \geq x_i^*$ at a given branch-and-bound node and furthermore in the construction of convex hull $(u_\omega^r)_i \geq (x)_i^*\gamma_\omega^r$ are added, the solution where $(u_\omega^{r'*}/\gamma_\omega^{r'*})_i < (x^*)_i$ will be cut off.

Inspired by the work of Sherali and Zhu (2006), we propose a spatial branch-and-bound algorithm where we branch on the continuous first-stage variables. At each node $q$ of the branch-and-bound tree. The following problem $(PCBAB_q)$ is solved,

$$(PCBAB_q) \quad \min \quad c^T x + \sum_{\omega\in\Omega} \tau_\omega d_\omega^T y_\omega \tag{4.16a}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{4.16b}$$

$$x_q^{lb} \leq x \leq x_q^{ub} \tag{4.16c}$$

$$x \in X \tag{4.16d}$$

$$(x, y_\omega) \in conv(S_\omega^q) \tag{4.16e}$$

$$S_\omega^q = \left\{(x, y_\omega | A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega}, y_\omega \in Y, x_q^{lb} \leq x \leq x_q^{ub}\right\} \tag{4.16f}$$

Note that the only difference between $S_\omega^q$ and $S_\omega$ is that the first-stage decisions $x$ is constrained in $[x_q^{lb}, x_q^{ub}]$ in $S_\omega^q$. As we assume that problem $(P)$ has relatively complete recourse, $conv(S_\omega^q)$ is a nonempty compact convex set for any fixed $x$, s.t. $x \in X$, $x_q^{lb} \leq x \leq x_q^{ub}$, $A_0 x \geq b_0$, $g_0(x) \leq 0$, for every $\omega \in \Omega$. Therefore, at each node $q$ of the branch-and-bound tree, GBD can be applied to solve $(PCBAB_q)$ to $\epsilon$-optimality in the same way that we solve $(PC)$. We denote the optimal objective of $(PCBAB_q)$ as $v(PCBAB_q)$. The optimal solution to $(PCBAB_q)$ is $(x_q^*, y_{q\omega_1}^*, y_{q\omega_2}^*, \cdots, y_{q\omega_{|\Omega|}}^*)$. A heuristic algorithm can be applied to find a feasible solution to problem $(P)$, while $x$ has to satisfy the upper and lower bound constraints at node $q$.

**Heuristic 1:** Fix the first-stage decisions at the optimal solution of $(PCBAB_q)$, $x_q^*$. Solve the subproblems with integrality constraints in parallel with some MINLP solvers, such as DICOPT (Viswanathan and Grossmann, 1990), Pajarito (Lubin et al., 2016), SBB (Bussieck and Drud, 2001), AlphaECP (Westerlund and Lundqvist, 2001). Note that we assume relatively complete recourse. Therefore, a feasible solution to $(P)$ can be obtained at node $q$, which is denoted as $(x_q^f,\ y_{q\omega_1}^f, y_{q\omega_2}^f, \cdots, y_{q\omega_{|\Omega|}}^f)$.

We denote the objective of the feasible solution found at node $q$ as $V(PCBAB_q)$. Note that $v(PCBAB_q)$ and $V(PCBAB_q)$ are valid lower and upper bounds for the original problem $(P)$, respectively, if the domain of $x$ is restricted to $[x_q^{lb}, x_q^{ub}]$. In the GBD-based branch and bound algorithm, if $V(PCBAB_q) \leq v(PCBAB_q)+\epsilon$, node $q$ can be fathomed by optimality. We denote the global lower bound as $v$, the global upper bound as $V$. If $v(PCBAB_q) \geq V$, node $q$ is fathomed by bound. Otherwise, we select one component of the continuous first-stage variables to branch on. The following branching rule is used to select one component of the continuous first stage variable to branch on.

**Branching rule A:**

At node $q$, calculate the distance of each component of the optimal continuous first-stage variables to its bounds at node $q$.

$$\sigma_i = \min\{(x_q^*)_i - (x_q^{lb})_i, (x_q^{ub})_i - (x_q^*)_i\}, \quad \forall i \in I \backslash I_1 \tag{4.17}$$

The variable with maximum distance to its bounds is selected to branch on.

$$p = \arg\max_{i \in I \backslash I_1} \sigma_i \tag{4.18}$$

By using Branching rule A, we are able to prove the convergence of the proposed algorithm later in this section.

The node $q$ is partitioned into two new nodes $q_1$, $q_2$. Constraints $(x_q^{lb})_p \leq (x)_p \leq (x_q^*)_p$ and $(x_q^*)_p \leq (x)_p \leq (x_q^{ub})_p$ are added to node $q_1$ and $q_2$ respectively.

The steps of the GBD-based branch-and-bound algorithm are outlined as follows.

**Algorithm GBDBAB**

**Step 0: Initialization Step.** Initialize the global upper bound $V = +\infty$, the global lower bound $v = -\infty$, the iteration counter $k = 1$, the list of active nodes $L_k = \{q^0\}$. Set $x_{q^0}^{lb} = 0$,

$x^{ub}_{q^0} = x^{ub}$. Let $\epsilon \geq 0$ be a selected optimality tolerance. Use GBD to solve the problem at the root node $q^0$ to $\epsilon$-optimality. Let $v = v(PCBAB_{q^0})$. Apply a heuristic algorithm like heuristic 1 to obtain a feasible solution to $(P)$ denoted as $(x^f_{q^0}, y^f_{q^0})$. Let the objective value of the feasible solution at node $q^0$ be $V_{q_0}$. Set $V = V_{q^0}$. If $V \leq v + \epsilon$, stop. Otherwise, go to step 1.

**Step 1: Branching Step.** Select the node $q$ in the active node list with the minimum lower bound $v(PCBAB_q)$. Branch on first stage variable $(x)_p$ according to the Branching rule A. Create two new nodes $q_1$, $q_2$ and add constraints $(x^{lb}_q)_p \leq (x)_p \leq (x^*_q)_p$ and $(x^*_q)_p \leq (x)_p \leq (x^{ub}_q)_p$ to node $q_1$ and $q_2$ respectively. Let $k = k+1$. Delete node $q$ in $L_k$ and add node $q_1$, $q_2$ to $L_k$. Go to step 2.

**Step 2: Bounding Step.** Solve the two problems $(PCBAB_{q_1})$ and $(PCBAB_{q_2})$ using generalized Benders decomposition to $\epsilon$-optimality respectively. Update the global lower bound $v$ with $v = min_{q \in L_k} v(PCBAB_q)$. A heuristic algorithm is applied to obtain feasible solutions at node $q_1$ and $q_2$ that yield objective values of $V(PCBAB_{q_1})$ and $V(PCBAB_{q_2})$ respectively. Update the global upper bound $V$ if $V(PCBAB_{q_1}) \leq V$ or $V(PCBAB_{q_2}) \leq V$. Go to Step 3.

**Step 3: Fathoming Step.** For the two new nodes $q_i$, $i = 1, 2$, fathom the node if $V(PCBAB_{q_i}) \leq v(PCBAB_{q_i}) + \epsilon$, $i = 1, 2$. Fathom any node $q$ with $v(PCBAB_q) + \epsilon \geq V$. If the node list $L_k$ becomes empty, stop. Otherwise go to step 1.

In order to prove the finite $\epsilon$-convergence of our proposed GBDBAB algorithm we make the following assumption.

**Assumption 4.** $\forall \epsilon > 0$, $\exists \delta_\epsilon > 0$, such that for any node $q$ with $\sigma_i \leq \delta_\epsilon$, $\forall i \in I \backslash I_1$, $V(PCBAB_q) \leq v(PCBAB_q) + \epsilon$, i.e., the node is fathomed by optimality if $\sigma_i \leq \delta_\epsilon$, $\forall i \in I \backslash I_1$. Here $\sigma_i$, $\forall i \in I \backslash I_1$, is defined in Branching rule A. $V(PCBAB_q)$ is obtained by heuristic 1.

As we have assumed that problem $(P)$ is bounded, the change of the first-stage variables

can only have finite change of value in the objective. As $\delta_\epsilon \to 0$, node $q$ can be fathomed by optimality based on Corollary 1. Therefore, Assumption 4 always holds for problems that are bounded.

**Proposition 2.** If Assumption 4 holds, the algorithm GBDBAB has finite $\epsilon$-convergence.

*Proof.* First we prove that for a given node $q$ and any continuous first-stage variables $(x)_i$ in node $q$ that are constrained in the interval $\left[(x_q^{lb})_i, (x_q^{ub})_i\right]$ where $(x_q^{ub})_i - (x_q^{lb})_i \leq \delta_\epsilon$, variable $i$ will not be branched on again if branching rule A is used. Assuming that variable $i$ is branched on again when branching rule A is used, we have $\sigma_{i'} \leq \sigma_i \leq \delta_\epsilon$, $\forall i' \in I \backslash \{I_1 \cup i\}$. However, by Assumption 4, any node $q$ with $\sigma_i \leq \delta_\epsilon$, $\forall i \in I \backslash I_1$,is fathomed by optimality. We reach a contradiction. Therefore, in the worst case, the domain of each continuous first stage variable will be partitioned into intervals with length $\delta_\epsilon$. There will be a finite number of hyperrectangle partitions for the domain of the continuous first-stage variables $[0, x^{ub}]$, i.e., a finite number of nodes in the branch-and-bound tree. In each node, GBD can converge in a finite number of iterations according to Geoffrion (1972). Thus, the algorithm GBDBAB is able to converge to $\epsilon$-optimum in a finite number of iterations. □

**Remark.** Here, in order to prove the convergence of the algorithm, we assume that the convex hull of $S_\omega^q$ is constructed by the hierarchy of relaxations proposed (Ruiz and Grossmann, 2012). If there are only a few binary variables in the second stage, the representation of the convex hull is tractable. However, if the number of binary variables in the second stage is large, it is computationally expensive to represent the convex hull since there can be $2^{|J_1|}$ disjuncts. Therefore, a more limited sequential convexification approach should be applied, which will be discussed in section 4.4.

**Remark.** Note that at each branching step, the child nodes generated are restrictions of their parent node. The Benders cuts in the master problem of the parent node can be by inherited by the child nodes, which will make the GBD algorithm for the child node require fewer iterations to converge compared to running it from scratch.

**An illustrative example**

The following problem is solved by GBDBAB as an illustrative example.

$$\min \quad x_1 + x_2 + 3x_3 + 3x_4 + \sum_{\omega=\omega_1,\omega 2} \tau_\omega \big(y_{1\omega} - 12y_{2\omega} + 100y_{3\omega} + 3y_{4\omega} - 3y_{5\omega}\big) \tag{4.19a}$$

$$x_1 \leq 4x_3, \quad x_2 \leq 2x_4, \tag{4.19b}$$

$$x_1, x_2 \geq 0 \quad x_3, x_4 \in \{0, 1\} \tag{4.19c}$$

$$y_{1\omega} \leq x_1, \quad y_{2\omega} \leq x_2 \quad \forall \omega = \omega_1, \omega_2 \tag{4.19d}$$

$$(y_{1\omega} - 3)^2 + (y_{2\omega} - 2)^2 \leq 1 + 16(1 - y_{4\omega}) \quad \forall \omega = \omega_1, \omega_2 \tag{4.19e}$$

$$(y_{1\omega} - 1)^2 + y_{2\omega}^2 \leq 1 + 16y_{4\omega} \quad \forall \omega = \omega_1, \omega_2 \tag{4.19f}$$

$$y_{1\omega}^2 + (y_{2\omega} - 1)^2 \leq 1 + 16(1 - y_{5\omega}) \quad \forall \omega = \omega_1, \omega_2 \tag{4.19g}$$

$$(y_{1\omega} - 4)^2 + (y_{2\omega} - 1)^2 \leq 1 + 16y_{5\omega} \quad \forall \omega = \omega_1, \omega_2 \tag{4.19h}$$

$$y_{1\omega} + y_{2\omega} + y_{3\omega} \geq d_\omega \quad \forall \omega = \omega_1, \omega_2 \tag{4.19i}$$

$$y_{1\omega}, y_{2\omega}, y_{3\omega} \geq 0, \quad y_{4\omega}, y_{5\omega} \in \{0, 1\} \quad \forall \omega = \omega_1, \omega_2 \tag{4.19j}$$

where $x_1, x_2, x_3, x_4$ are the first-stage variables, $y_{1\omega}, y_{2\omega}, y_{3\omega}, y_{4\omega}, y_{5\omega}$ are the second-stage variables. Both the first and the second-stage variables are mixed-binary. $\tau_{\omega_1} = \tau_{\omega_2} = 0.5$. $d_\omega$ is the uncertain parameter. $d_{\omega_1} = 1.5$, $d_{\omega_2} = 2$. Here we describe how this problem can be solved by GBDBAB. Recall that at each node $q$ of the branch and bound tree, GBD is applied to solve problem $(PCBAB_q)$.

For this problem, three nodes are visited in order to solve the problem to a relative optimality gap within 0.1% (see Figure 4.1). At each node, the lower bound $v(PCBAB_q)$ is the lower bound obtained by the GBD algorithm, i.e., the objective value of the Benders master problem in the last iteration. The upper bound $V(PCBAB_q)$ is obtained by using Heuristic 1.

$v(PCBAB_q)$, $V(PCBAB_q)$, the optimal solution at each node are shown in Table 4.1. The Benders cuts that are generated at all the nodes are shown in Tables B.1-B.3 in Appendix 1. At the root node, 15 iterations are needed for the GBD algorithm to converge. However, the relative optimality gap of the global upper and lower bound that can be obtained at the root node is not within 0.1%. Therefore, we select one continuous first stage variable to branch on according to branching rule A. $x_1$ is branched on and two new nodes 1 and 2 are created. GBD is applied to solve the problem at nodes 1 and 2, respectively. Both nodes 1 and 2 inherit the Benders cuts of the root node because the problems solved at these two nodes are restrictions of the root node and the Benders cuts of the root node are still valid. As a result, only 3 and 2 iterations are needed for the GBD to converge at nodes 1 and 2, respectively (see Tables B.2, B.3). The upper and the lower bounds of nodes 1 and 2 are within 0.1% optimality gap. Nodes 1 and 2 can be fathomed by optimality and GBDBAB terminates. An optimal value of -6.02080 is obtained.

However, if the hull relaxation of the CNF is used, which is a relaxation of $conv(S^q_\omega)$, the lower bound that can be obtained at the root node is -12.5767. The DNF yields a much tighter lower bound than the CNF at the root node, which shows the impact of applying the basic step.

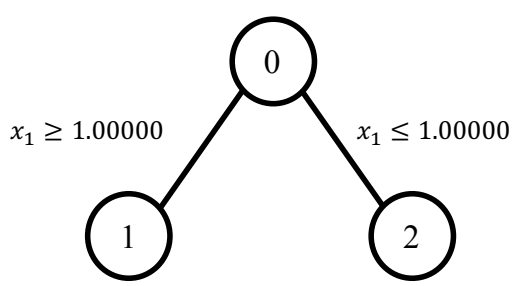


Figure 4.1: Branch and bound tree for the illustrative example

## 4.4 A sequential convexification scheme to solve ($PCBAB_q$)

In solving problem ($PCBAB_q$), we assume that the convex hulls of $S^q_\omega$, $\forall \omega \in \Omega$ are constructed by applying the basic steps and converting each GDP from a CNF to a DNF.

Table 4.1: Optimal solution, upper and lower bound at each node of the BAB tree

| Node | $v(PCBAB_q)$ | $V(PCBAB_q)$ | $x_1^*$ | $x_2^*$ | $x_3^*$ | $x_4^*$ | Branching constraints |
|---|---|---|---|---|---|---|---|
| 0 | -6.04451 | -5.98613 | 1.00000 | 1.03467 | 1 | 1 | - |
| 1 | -6.02092 | -6.02072 | 1.00005 | 1.00003 | 1 | 1 | $x_1 \geq 1.00000$ |
| 2 | -6.02082 | -6.02080 | 1.00000 | 1.00000 | 1 | 1 | $x_1 \leq 1.00000$ |

However, the number of disjuncts can be large if we have a large number of binary variables in the second-stage decisions. The number of variables that are needed to represent the convex hull grows exponentially with the number of binary variables in the second-stage decisions. Therefore, we propose a sequential convexification scheme, which progressively applies the basic steps to the CNF representation of $S_\omega^q$. As we will see later, in some cases, it is not necessary to convert the CNF to DNF by applying all the basic steps. Namely, in some cases applying part of but not all the basic steps is sufficient to solve $(PCBAB_q)$. The partial application of all the possible basic steps can be regarded as a sequential convexification scheme to solve $(PCBAB_q)$, which will be discussed next.

Similarly to (4.9), we define the disjunction that specifies the $j$th binary variable as $S_{\omega j}^{q1} \cup S_{\omega j}^{q0}$, $j \in J_1$. The CNF of $S_\omega^q$ can be denoted as $\cap_{j \in J_1}(S_{\omega j}^{q1} \cup S_{\omega j}^{q0})$. In the sequential convexification scheme to solve $(PCBAB_q)$, we start with the hull relaxation of the CNF, and progressively apply basic steps to construct tighter relaxations if the problem $(PCBAB_q)$ is not solved.

Before we describe the details of the sequential convexification scheme, we first define some notation. A partial application of basic steps to the CNF $\cap_{j \in J_1}(S_{\omega j}^{q1} \cup S_{\omega j}^{q0})$ can be represented as $S_\omega^q = \cap_{t \in T_\omega^q}\big(\cap_{j \in D_{\omega t}^q}(S_{\omega j}^{q1} \cup S_{\omega j}^{q0})\big)$, where $T_\omega^q$ is the set of conjuncts, $D_{\omega t}^q$ is the set of the indices of the binary variables specified by conjunct $t$. $D_{\omega t}^q$ are disjoint sets and $\cup_{t \in T_\omega^q} D_{\omega t}^q = J_1$, i.e., the value of each binary variable $j \in J_1$ has to be specified in one and only one of the conjuncts. For each $t \in T_\omega^q$, $\cap_{j \in D_{\omega t}^q}(S_{\omega j}^{q1} \cup S_{\omega j}^{q0})$ can be represented in DNF by applying the basic steps, $\cap_{j \in D_{\omega t}^q}(S_{\omega j}^{q1} \cup S_{\omega j}^{q0}) = \cup_{r \in R_{\omega t}^q} S_{\omega tr}^q$. $R_{\omega t}^q$ is the set of disjuncts for the DNF representation. In each $S_{\omega tr}^q$, the values of all $(y_\omega)_j$, $j \in D_{\omega t}^q$, are specified to 0 or 1. Note that the dimension of $R_{\omega t}^q$ is $2^{|D_{\omega t}^q|}$, which corresponds to all possible combinations

of the binary variables in $D^q_{\omega t}$. For the CNF, $|T^q_\omega| = |J_1|$ and $|D^q_{\omega t}| = 1$. If we apply all the possible basic steps to convert the CNF to the corresponding DNF, $|T^q_\omega| = 1$ and $|D^q_{\omega t}| = |J_1|$. The hull relaxation of $\cap_{t\in T^q_\omega}(\cup_{r\in R^q_{\omega t}} S^q_{\omega tr})$ is given in (4.20a)-(4.20h),

$$x = \sum_{r\in R^q_{\omega t}} u^r_{\omega t}, \quad \forall t \in T^q_\omega \tag{4.20a}$$

$$y_\omega = \sum_{r\in R^q_{\omega t}} v^r_{\omega t}, \quad \forall t \in T^q_\omega \tag{4.20b}$$

$$\sum_{r\in R^q_{\omega t}} \gamma^r_{\omega t} = 1, \quad \forall t \in T^q_\omega \tag{4.20c}$$

$$\gamma^r_{\omega t} \geq 0, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t} \tag{4.20d}$$

$$A_{1,\omega} u^r_{\omega t} + \gamma^r_{\omega t} g_{1,\omega}(v^r_{\omega t}/\gamma^r_{\omega t}) \leq b_{1,\omega}\gamma^r_{\omega t}, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t} \tag{4.20e}$$

$$x^{lb}_q \gamma^r_{\omega t} \leq u^r_{\omega t} \leq x^{ub}_q \gamma^r_{\omega t}, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t} \tag{4.20f}$$

$$0 \leq v^r_{\omega t} \leq y^{ub}\gamma^r_{\omega t}, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t} \tag{4.20g}$$

$$(v^r_{\omega t})_j = (e^r_{\omega t})_j \gamma^r_{\omega t}, \quad \forall t \in T^q_\omega, r \in R^q_{\omega t}, j \in D^q_{\omega t} \tag{4.20h}$$

Instead of solving ($PCBAB_q$) directly, we can solve a relaxation of ($PCBAB_q$) where $(x, y_\omega) \in$ *h-rel*$\big(\cap_{t\in T^q_\omega}(\cup_{r\in R^q_{\omega t}} S^q_{\omega tr})\big)$. Problem ($PCBAB_q$) is solved in iterations where the number of iteration is denoted by $l$. In the first iteration, $l = 1$, the problem with $(x, y_\omega) \in$ *h-rel*$(\cap_{j\in J_1}(S^{q1}_{\omega j} \cup S^{q0}_{\omega j})$, $\forall \omega \in \Omega$, is solved. We denote the relaxation of ($PCBAB_q$) that is solved at iteration $l$ as ($PCBAB^l_q$). The hull relaxation of $\cap_{t\in T^q_\omega}(\cup_{r\in R^q_{\omega t}} S^q_{\omega tr})$ at iteration $l$ for scenario $\omega$ is denoted as *h-rel*$(S^q_{\omega l})$. ($PCBAB^l_q$) is formally defined by (4.21a)-(4.21d).

$$(PCBAB^l_q) \quad \min \quad c^T x + \sum_{\omega\in\Omega} \tau_\omega d^T_\omega y_\omega \tag{4.21a}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{4.21b}$$

$$x^{lb}_q \leq x \leq x^{ub}_q \tag{4.21c}$$

$$(x, y_\omega) \in h\text{-}rel(S^q_{\omega l}), \quad \forall \omega \in \Omega \tag{4.21d}$$

$(PCBAB^l_q)$ can be solved using GBD to $\epsilon$-optimality in a finite number of steps. The optimal value of $(PCBAB^l_q)$ is denoted as $v(PCBAB^l_q)$. The optimal solution is denoted as $(x_l^{q*}, y^{q*}_{l\omega_1}, y^{q*}_{l\omega_2}, \cdots, y^{q*}_{l\omega_{|\Omega|}})$. If we can prove that $(x_l^{q*}, y^{q*}_{l\omega_1}, y^{q*}_{l\omega_2}, \cdots, y^{q*}_{l\omega_{|\Omega|}})$ satisfies all the constraints of $(PCBAB_q)$, then $(x_l^{q*}, y^{q*}_{l\omega_1}, y^{q*}_{l\omega_2}, \cdots, y^{q*}_{l\omega_{|\Omega|}})$ solves $(PCBAB_q)$. More specifically, if $(x_l^{q*}, y^{q*}_{l\omega}) \in conv(S^q_\omega)$, $\forall \omega \in \Omega$, $(x_l^{q*}, y^{q*}_{l\omega_1}, y^{q*}_{l\omega_2}, \cdots, y^{q*}_{l\omega_{|\Omega|}})$ solves $(PCBAB_q)$.

However, deciding whether $(x_l^{q*}, y^{q*}_{l\omega})$ is in $conv(S^q_\omega)$ is not an easy problem as we do not want to use the closed-form representation of $conv(S^q_\omega)$, since this representation could be expensive when the number of second stage binary variables is large. Here, we provide a quick way to prove $(x_l^{q*}, y^{q*}_{l\omega}) \in conv(S^q_\omega)$ when some conditions are satisfied, which is only a sufficient condition for $(x_l^{q*}, y^{q*}_{l\omega})$ to be in $conv(S^q_\omega)$. Note that it is easy to check if a point is in $S^q_\omega$ just by checking if this point satisfies all the constraints that define $S^q_\omega$. Therefore, the basic idea of this procedure is that if we could find that $(x_l^{q*}, y^{q*}_{l\omega})$ can be expressed as convex combination of some points that are in $S^q_\omega$, then $(x_l^{q*}, y^{q*}_{l\omega})$ is in $conv(S^q_\omega)$. More specifically, note that after we solve $(PCBAB^l_q)$, we also obtain $v^{r*}_{\omega t}$, $\gamma^{r*}_{\omega t}$, $\forall \omega \in \Omega$, $t \in T^q_\omega$, $r \in R^q_{\omega t}$, where $y^{q*}_{l\omega} = \sum_{r \in R^q_{\omega t}} v^{r*}_{\omega t}$, $\forall \omega \in \Omega$, $t \in T^q_\omega$. We can establish a sufficient condition on $(x_l^{q*}, y^{q*}_{l\omega}) \in conv(S^q_\omega)$ by inspecting the values of $v^{r*}_{\omega t}$, $\gamma^{r*}_{\omega t}$ in Proposition 3.

**Proposition 3.** For a given scenario $\omega$, if $\exists t' \in T^q_\omega$ such that $(v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'})_j$, $\forall r \in R^q_{\omega t'}, \gamma^{r*}_{\omega t'} > 0, j \in J_1$, are 0 or 1, i.e., $v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'}$ satisfy the integrality constraints in $S^q_\omega$, we have $(x_l^{q*}, y^{q*}_{l\omega}) \in conv(S^q_\omega)$.

*Proof.* By construction, $(u^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'}, v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'})$, $\forall r \in R^q_{\omega t'}, \gamma^{r*}_{\omega t'} > 0$ already satisfies the continuous constraints that define $S^q_\omega$. Therefore, $(u^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'}, v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'})$, $\forall r \in R^q_{\omega t'}, \gamma^{r*}_{\omega t'} > 0$ are in $S^q_\omega$. $(x_l^{q*}, y^{q*}_{l\omega})$ can be expressed as convex combination of $(u^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'}, v^{r*}_{\omega t'}/\gamma^{r*}_{\omega t'})$, $\forall r \in R^q_{\omega t'}, \gamma^{r*}_{\omega t'} > 0$, i.e., $(x_l^{q*}, y^{q*}_{l\omega}) \in conv(S^q_\omega)$. □

Note that Proposition 3 is only a sufficient condition for $(x_l^{q*}, y^{q*}_{l\omega}) \in conv(S^q_\omega)$. If we can prove that $(x_l^{q*}, y^{q*}_{l\omega}) \in conv(S^q_\omega)$, $\forall \omega \in \Omega$, $(PCBAB_q)$ is solved. Otherwise, for the scenarios where we cannot prove that $(x_l^{q*}, y^{q*}_{l\omega}) \in conv(S^q_\omega)$, we apply basic steps by updating $T^q_\omega$, $D^q_{\omega t}$

to construct tighter relaxations until we are able to prove $(x_l^{q*}, y_{l\omega}^{q*}) \in conv(S_\omega^q)$, $\forall \omega \in \Omega$. Here, we propose two heuristics on how to apply the basic steps.

**Heuristic A** Note that in Proposition 3, in order to prove $(x_l^{q*}, y_{l\omega}^{q*}) \in conv(S_\omega^q)$ we need to find one disjunction $t'$ such that $(v_{\omega t'}^{r*}/\gamma_{\omega t'}^{r*})_j$, $\forall j \in J_1$, $\gamma_{\omega t'}^{r*} > 0$, satisfy the integrality constraints. Therefore, in this heuristic, for a given scenario $\omega$ and for each disjunction $t \in T_\omega^q$, we first identify the most fractional $(v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j$, $\forall r \in R_{\omega t}^q, \gamma_{\omega t}^{r*} > 0, j \in J_1$ and denote it as $f_t$.

$$f_t = \max_{r \in R_{\omega t}^q, \gamma_{\omega t}^{r*} > 0, j \in J_1} \min\{1 - (v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j, (v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j\} \tag{4.22}$$

We also denote the index of the most fractional $(v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j$ as $g_t$,

$$g_t = \arg\max_{j: r \in R_{\omega t}^q, \gamma_{\omega t}^{r*} > 0, j \in J_1} \min\{1 - (v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j, (v_{\omega t}^{r*}/\gamma_{\omega t}^{r*})_j\} \tag{4.23}$$

In the sufficient condition given by Proposition 3, we try to find disjunction $t'$ such that $f_{t'}$ is close to zero, i.e., all the $(v_{\omega t'}^{r*}/\gamma_{\omega t'}^{r*})_j$, $\forall j \in J_1$, $\gamma_{\omega t'}^{r*} > 0$ satisfy the integrality constraints. Here, we slightly abuse notation and denote the disjunction $t$ with the least $f_t$ as $t'$,

$$t' = \arg\min_{t \in T_\omega^q} f_t \tag{4.24}$$

If $f_{t'}$ is close to zero, we can prove $(x_l^{q*}, y_{l\omega}^{q*}) \in conv(S_\omega^q)$ by Proposition 3. Otherwise, we apply a basic step between $t'$ and the disjunction that specifies $g_{t'}$. The idea is to make the most fractional $(v_{\omega t'}^{r*}/\gamma_{\omega t'}^{r*})_j$ in disjunction $t'$ satisfy the integrality constraint in the next iteration.

**Remark.** For the problem with a large number of binary variables in the second stage, the sequential convexification scheme could potentially reduce the computational time by avoiding applying unnecessary basic steps. While the sequential convexification scheme is less expensive than using the DNF representation directly, solving ($PCBAB_q^l$) in the first iteration with the CNF representation can also be expensive. Therefore, one may start with the NLP relaxation of $S_\omega^{q^0}$ and start using the CNF representation when the lower bound cannot be improved by the NLP relaxation anymore. In fact, this strategy is used when we solve the planning problem using GBDBAB with sequential convexification scheme in section 4.5.

## 4.5 Computational results

In this section, computational results of three different problems are presented to demonstrate the effectiveness of GBDBAB. The tolerance for the relative optimality gap for GBDBAB is set to 0.1%. The GBDBAB algorithm to solve the three problems with an increasing number of scenarios is implemented in JuMP/Julia (Dunning et al., 2017). All the problems are solved on the 12 processors of an Intel Xeon (2.67GHz) machine with 64 GB RAM. Depending on the computational efficiency on different problems, different NLP solvers including Knitro (Byrd et al., 2006), and IPOPT (Wächter and Biegler, 2006) are used to solve the NLP subproblems in parallel. The choice of NLP solvers for each problem are discussed in detail in the subsequent subsections. CPLEX (CPLEX, 2009) is used to solve the MILP master problems. The corresponding deterministic equivalent formulations (DEFs) are implemented in GAMS and solved using convex MINLP solvers including DICOPT, AlphaECP, and SBB.

### 4.5.1 Computational results of the illustrative example with quadratic constraints

In section 4.3, an illustrative example with 2 scenarios is solved using GBDBAB. The convex nonlinear constraints in the model are all quadratic as shown in (4.19e)-(4.19h). $d_\omega$ is the only uncertain parameter in the problem. In this section, we generate stochastic convex MINLP problems with an increasing number of scenarios by uniformly sampling $d_\omega$ in the interval $[0, 3]$. We assume that all the scenarios are of equal probability. The time limit for all DEFs of this small problem is set to 10,000 secs. The sizes of the DEFs, the total running time and the relative optimality gap of each solver are shown in Table 4.2. Although AlphaECP and DICOPT are able to solve the 20-scenario problem to optimality within 10 secs, the computational time increases dramatically with the increase in the number of scenarios. For instance, for the DEF with 300 scenarios, neither SBB nor DICOPT is able to obtain the optimal solution within the time limit. AlphaECP solves the problem in 917 secs.

Table 4.2: Computational statistics of the DEFs of the illustrative problem

| Scenarios | Linear Constr | Nonlinear Constr | Binary Var | Continuous Var | SBB sec(gap) | AlphaECP sec(gap) | DICOPT sec(gap) |
|---|---|---|---|---|---|---|---|
| 20 | 62 | 80 | 42 | 62 | Timed out(7%) | 9 | 2 |
| 60 | 182 | 240 | 122 | 182 | Timed out(234%) | 60 | 10 |
| 150 | 452 | 600 | 302 | 452 | Timed out(245%) | 254 | 685 |
| 300 | 902 | 1200 | 602 | 902 | Timed out(247%) | 917 | Timed out(9%) |

GBDBAB is also implemented to solve this illustrative problem. Since there are only two binary variables in the second stage, $conv(S^q_\omega)$ can be constructed easily with the DNF representation. The NLP subproblems are solved using IPOPT in parallel. The upper bound at each node is obtained by applying Heuristic 1 where the upper bound subproblems are solved using Pajarito. The total wall time, the wall time corresponding to solving the master problems, subproblems and upper bound subproblems, and the number of nodes visited are shown in Table 4.3. In all the cases, only three nodes are visited in order to solve the problems to optimality due to the tight relaxations given by the DNF representation. It is easy to observe that the wall time for solving the subproblems is the most significant part of the total wall time. For the problems with 150 and 300 scenarios, GBDBAB performs better than using the solvers to solve the DEFs.

Table 4.3: Computational statistics of solving the illustrative problem using GBDBAB

| Scenarios | Time (s) | Master (s) | Subproblem (s) | UB subproblem (s) | Nodes |
|---|---|---|---|---|---|
| 20 | 80 | 3 | 59 | 10 | 3 |
| 60 | 76 | 3 | 58 | 5 | 3 |
| 150 | 111 | 10 | 81 | 9 | 3 |
| 300 | 121 | 11 | 81 | 11 | 3 |

### 4.5.2 Computational results of the constrained layout problem

The constrained layout problem under price uncertainty described in Appendix 2 is tested with 3 rectangles and 2 areas. Since the model is formulated as a GDP, the DEFs of the model are converted to MINLP with the big-M and the hull reformulation, respectively. The approach to convert a GDP to the big-M and the hull reformulation can be found in Table

7 of Grossmann and Trespalacios (2013)). The time limit is set to 10,000 secs. The sizes of the DEFs with the big-M and the hull reformulation are shown in Table 4.4 and Table 4.5 respectively. Although all the solvers can solve the DEFs with 3 and 9 scenarios within the time limit, they all fail to obtain the optimal solution for the problems with 36 and 100 scenarios.

Table 4.4: Computational statistics of the big-M reformulation of the constrained layout problem

| Scenarios | Linear Constr | Nonlinear Constr | Binary Var | Continuous Var | SBB sec(gap) | AlphaECP sec(gap) | DICOPT sec(gap) |
|---|---|---|---|---|---|---|---|
| 3 | 54 | 72 | 30 | 36 | 14 | 120 | 3 |
| 9 | 108 | 216 | 66 | 84 | 4,142 | 1,478 | 43 |
| 36 | 351 | 864 | 228 | 300 | Timed out(94%) | Timed out* | Timed out(84%) |
| 100 | 927 | 2,400 | 612 | 812 | Timed out(98%) | Timed out* | Timed out(97%) |

Table 4.5: Computational statistics of the hull reformulation of the constrained layout problem

| Scenarios | Linear Constr | Nonlinear Constr | Binary Var | Continuous Var | SBB sec(gap) | AlphaECP sec(gap) | DICOPT sec(gap) |
|---|---|---|---|---|---|---|---|
| 3 | 360 | 72 | 30 | 180 | 38 | 32 | 6 |
| 9 | 702 | 216 | 66 | 372 | 5,626 | 638 | 49 |
| 36 | 2,241 | 864 | 228 | 1,236 | Timed out(52%) | Timed out (46%) | Timed out* |
| 100 | 5,889 | 2,400 | 612 | 3,284 | Timed out(82%) | Timed out (67%) | Timed out* |

GBDBAB is also implemented to solve the constrained layout problems with an increasing number of scenarios. In order to obtain $conv(S^q_\omega)$, constraint (B.4) and the constraints setting the upper and lower bound of all the variables are added to each disjunct shown in (B.5). We further apply basic steps to convert the CNF to DNF and reformulate the DNF with hull relaxation, which gives us $conv(S^q_\omega)$. The NLP subproblems are solved using IPOPT in parallel. The statistics of GBDBAB to solve these problems are shown in Table 4.6. For the problems with 3, 36, and 100 scenarios, optimality is proved at the root node. For the problem with 9 scenarios, 3 nodes are visited in order to prove optimality. Most of the computational time is still in solving the NLP subproblems.

*No feasible solution found in 10,000 secs time limit.

Table 4.6: Computational statistics of solving the constrained layout problem with GBDBAB

| Scenarios | Time (s) | Master (s) | Subproblem (s) | UB subproblem (s) | Nodes |
|---|---|---|---|---|---|
| 3 | 137 | 40 | 63 | 8 | 1 |
| 9 | 276 | 65 | 17 | 17 | 3 |
| 36 | 444 | 117 | 277 | 20 | 1 |
| 100 | 537 | 85 | 363 | 39 | 1 |

### 4.5.3 Computational results of the planning problem

The planning problem reported in Li and Grossmann (2018a) is tested with 2 suppliers, 2 plants, 2 customers, and the number of scenarios from 3 to 81. The time limit for solving the DEFs is set to 50,000 secs. For the 81-scenario problem, none of the solvers is able to obtain the optimal solution within the time limit.

Table 4.7: Computational statistics of the DEFs of the planning problem

| Scenarios | Linear Constr | Nonlinear Constr | Binary Var | Continuous Var | AlphaECP sec(gap) | SBB sec(gap) | DICOPT sec(gap) |
|---|---|---|---|---|---|---|---|
| 3 | 332 | 12 | 32 | 338 | 6 | 49 | 3 |
| 9 | 980 | 36 | 80 | 998 | 81 | Timed out(2%) | 9 |
| 27 | 2,924 | 108 | 224 | 2,978 | 3,530 | Timed out(19%) | 96 |
| 81 | 8,756 | 324 | 656 | 8,918 | Timed out(2%) | Timed out(40%) | Timed out(0.2%) |

GBDBAB is also applied to solve the planning problem. The NLP subproblems are solved with Knitro in this case since it behaves best for this problem among the NLP solvers available in JuMP. GBDBAB is able to solve all the planning problems at the root node. Sequential convexification scheme with Heuristic A is applied and the maximum and the minimum number of basic steps that are applied in all the scenarios are shown in the last column of Table 4.8. In each of the four cases, the number of basic steps needed varies for different scenarios. For some of the scenarios, we are able to prove the solution $(x, y_\omega)$ is in $conv(S_\omega^{q^0})$ by Proposition 3 and therefore no more basic steps need to be applied. The computational statistics of GBDBAB are shown in Table 4.8. The problems with an increasing number of scenarios can all be solved to optimality by GBDBAB within 10,000 secs.

**Remark.** The computational results in this section demonstrate that GBDBAB can out-

Table 4.8: Computational statistics of solving the planning problem using GBDBAB

| Scenarios | Time (s) | Master (s) | Subproblem (s) | UB subproblem (s) | Basic step (max, min) |
|---|---|---|---|---|---|
| 3 | 705 | 59 | 491 | 16 | (7,2) |
| 9 | 1,221 | 92 | 861 | 23 | (5,1) |
| 27 | 1,859 | 216 | 1,403 | 17 | (0,0) |
| 81 | 7,994 | 1,534 | 5,091 | 83 | (1,0) |

perform the convex MINLP solvers in solving the DEFs with a large number of scenarios. However, for the first small problem and the planning problem, some of the solvers can obtain solutions to the DEFs with small optimality gaps. The effect of using GBDBAB is most significant for the constrained layout problem. First of all, it is relatively easy to construct the convex hull of each subproblem as there are only 3 disjunctions in each subproblem. Second, since this problem is highly nonlinear, it would take solvers that use linearization strategies like DICOPT and AlphaECP many iterations to converge. Third, the constrained layout problem also has poor NLP relaxation. Solvers that are sensitive to the quality of NLP relaxation of the original problem like SBB would need more iterations to converge. Therefore, GBDBAB is most effective in solving highly nonlinear problems with poor relaxations but with small number of binary variables in the second-stage decisions.

## 4.6 Conclusions

In this chapter, a generalized Benders decomposition-based branch-and-bound algorithm, GBDBAB, is proposed to solve two-stage convex mixed integer nonlinear stochastic programs with mixed-binary variables in both first and second-stage decisions. In order to obtain the convex hull of the mixed-binary nonlinear subproblems in closed-form, each subproblem is first represented in conjunctive normal form (CNF), and we apply basic steps to transform CNF to disjunctive normal form (DNF). The convex hull of each subproblem can be represented by the hull relaxation of the DNF. For the problems with pure binary first-stage variables, we are able to prove that GBD has finite $\epsilon$-convergence if the convex hull of each subproblem is constructed in closed-form. For the problems with mixed-binary first-stage

variables, we also prove that in order to have finite $\epsilon$-convergence, we may need to branch on the first stage continuous variables. A sequential convexification scheme is proposed to adaptively apply basic steps for the subproblems with a large number of binary variables.

Three problems with an increasing number of scenarios are solved with GBDBAB to within 0.1% optimality gap. Convex MINLP solvers including SBB, AlphaECP, and DICOPT are used to solve the corresponding DEFs to benchmark the proposed algorithm. It is shown that the proposed algorithm outperforms the solvers for the problems with a large number of scenarios. Especially, the algorithm is preferable for highly nonlinear problems with poor relaxations but with small number of binary variables in the second-stage decisions like the constrained layout problem.

# Chapter 5

# A Generalized Benders Decomposition-based Branch and Cut Algorithm for Two-stage Stochastic Programs with Nonconvex Constraints and Mixed-binary First and Second Stage Variables

## 5.1 Problem Statement

We first define the two-stage stochastic programming problem $(P)$ that we address in this chapter,

$$(P) \quad \min \quad z = c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{5.1a}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{5.1b}$$

$$A_{1,\omega}x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{5.1c}$$

$$x \in X, \quad X = \left\{x : x_i \in \{0,1\}, \forall i \in I_1,\ 0 \leq x \leq x^{ub}\right\} \tag{5.1d}$$

$$y_\omega \in Y_\omega \quad \forall \omega \in \Omega, \quad Y_\omega = \left\{y_\omega : y_{\omega j} \in \{0,1\}, \forall j \in J_1,\ 0 \leq y_\omega \leq y_\omega^{ub}\right\} \tag{5.1e}$$

Here, $x$ represents the first stage decisions. $y_\omega$ represents the second stage decisions in scenario $\omega$. $\tau_\omega$ represents the probability of scenario $\omega$. $g_0$, $g_{1,\omega}$, can be smooth nonconvex functions. Both the first and the second stage decisions are mixed-binary. Let $I = \{1, 2, \cdots, n\}$ be the index set of all the first stage variables. $I_1 \subseteq I$ is the subset for indices of the binary first stage variables. Let $J = \{1, 2, \cdots, m\}$ be the index set of all the second stage variables. $J_1 \subseteq J$ is the subset for the indices of the binary second stage variables. $x^{ub}$ is a vector that represents the upper bound of all the first stage variables. $y_\omega^{ub}$ is a vector that represents the upper bound of all the second stage variables. We make the following assumptions about problem (P).

**Assumption 5.** Problem $(P)$ has relatively complete recourse, i.e., any solution $x$ that satisfies the first stage constraints has feasible recourse decisions in the second stage.

**Assumption 6.** The feasible region of $(P)$ is compact.

**Assumption 7.** $x^{ub}$ and $y^{ub}$ are finite, i.e., both the first and the second stage decisions are bounded.

## 5.2 Motivation for a New Algorithm

In this chapter, we address the problem of solving problem $(P)$, i.e., nonconvex SMINLPs with mixed-binary first and second stage variables. Up until now, the decomposition algorithms that explicitly consider the nonconvex subproblems are the branch and bound algorithms proposed by Cao and Zavala (2017) and Kannan (2018) where the authors branch on the first stage variables and derive a valid bound at each node of the branch and bound process. The valid bound can come from Lagrangean relaxation or perfect information, which

can be considered as a special case of Lagrangean relaxation with zero dual multipliers. Cao and Zavala (2017) and Kannan (2018) prove finite convergence of their algorithms. However, the branch and bound algorithms tend to have slow convergence computationally, especially when the number of first stage variables is large, i.e., a large number of nodes need to be solved in order to prove global optimality. In this chapter, we explore the possibility of doing branch and cut instead of branch and bound.

The main challenge for doing branch and cut is how we can integrate cutting planes in the algorithm while still guaranteeing convergence. In the introduction section, we have reviewed the literature for using Benders-like decomposition to solve linear SMIPs where parametric cuts are used to convexify the subproblems. For nonconvex SMINLPs, cutting planes cannot be used directly to convexify the subproblems. However, if we replace the nonconvex functions in stage 2 by their convex relaxations or MILP relaxations, for example, we can relax a bilinear term by its McCormick envelope or piecewise McCormick envelope (Misener et al., 2011), the subproblems will become MILPs or convex MINLPs, which can be convexified by cutting planes. In principle, if the convexified subproblems are MILPs, parametric cuttting planes such as Gomory mixed-integer cut (Balas et al., 1996), lift-and-project cut (Balas et al., 1993), and RLT (Sherali and Adams, 1990) can be used. If the convexified subproblems become convex MINLPs, rank-one lift-and-project cuts can be used in the same way as in Li and Grossmann (2018a). In this chapter, only rank-one lift-and-project cuts for MILP subproblems are implemented. The framework can be easily extended to other types of cutting planes. Therefore, in order to integrate the cutting planes into the decomposition algorithm, the idea is to have a generalized Benders decomposition algorithm where the nonconvex subproblems are first relaxed to MILP problems, and the MILP problems are convexified by cutting planes to derive valid Benders cuts. However, this will not guarantee global optimality.

In order to solve the problem to global optimality, we add Lagrangean cuts to the Benders master problem as well. Lagrangean cuts are proved to be valid for the Benders master problem (Ogbe and Li, 2018; Li and Grossmann, 2018a). If we add Lagrangean cuts to the

Benders master problem, the lower bound obtained by the Benders master problem is at least as tight as using Lagrangean decomposition (see Proposition 2 in Li and Grossmann (2018a)). Therefore, we propose a generalized Benders decomposition (GBD) based branch and cut algorithm where we have both Benders cuts and Lagrangean cuts in the Benders master problem and branch on the first stage variables similar to Kannan (2018). We are able to prove that the proposed algorithm converges in the limit.

The paper is organized as follows: In section 4, we describe the decomposition algorithm. In section 5, we prove the convergence of the proposed algorithm. In section 6, we describe the implementation details. The computational results are given in section 7. We draw the conclusions in section 8.

## 5.3 The Proposed Algorithm

### 5.3.1 Overview of the Proposed Algorithm

The major steps of the proposed algorithm are shown in Figure 5.1. There are two major components in the proposed algorithm: (1) spatial branch and bound (described by the rounded rectangles in Figure 5.1). (2) the generalized Benders decomposition-based algorithm to solve a single node in the branch and bound process (described by the regular rectangles in Figure 5.1).

At a high level, the algorithm performs spatial branch and bound search on the first stage variables. We need to specify the node selection rule and branching rule for the spatial branch and bound algorithm. The details are discussed in subsection 5.3.3.

Note that we only branch on the first stage variables. Therefore, each node in the branch and bound process has a decomposable structure. We solve each node by the generalized Benders decomposition-based algorithm described by the regular rectangles in Figure 5.1. We have two types of valid inequalities, including Benders cuts and Lagrangean cuts in the Benders master problem. The Lagrangean cuts come from solving the Lagrangean subproblems iteratively. We update the Lagrangean multipliers after each Lagrangean iteration.

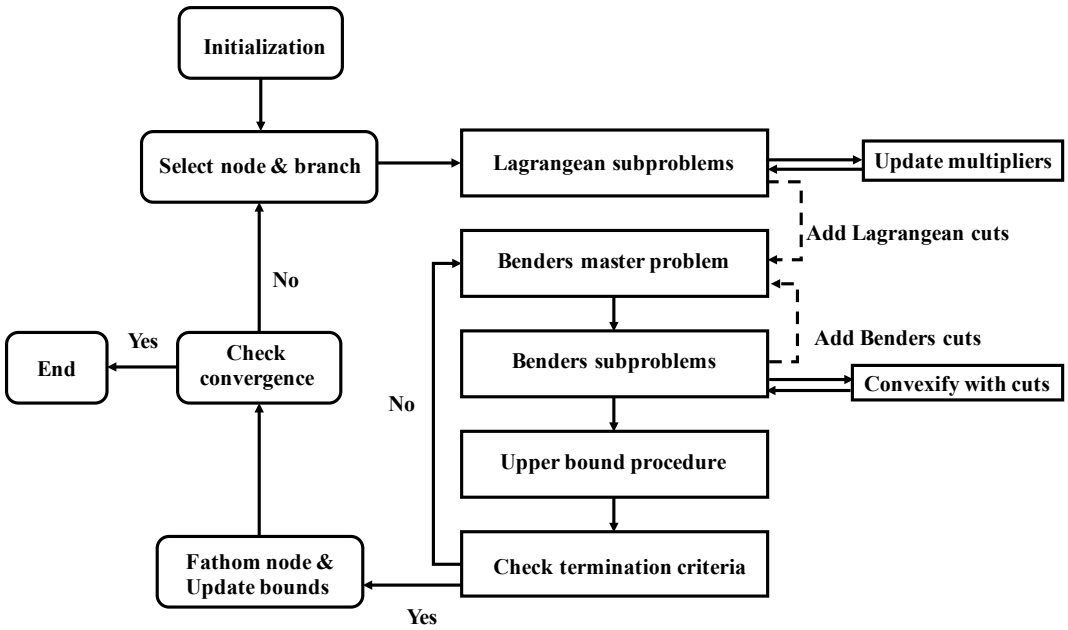


Figure 5.1: Flowchart of the proposed algorithm

Then we add all the Lagrangean cuts to the Benders master problem and start the Benders iterations. Note that the Benders iterations are *independent* of the Lagrangean iterations. In the Benders iterations, we solve the Benders master problem and the Benders subproblems iteratively. The Benders subproblems are convexified by cutting planes. An upper bound can be obtained at each Benders iteration using some heuristics. We check some termination criteria in each Benders iteration. The details of the decomposition algorithm to solve a single node will be described in subsection 5.3.2.

### 5.3.2 Decomposition Algorithm to Solve a Single Node

Before we address the branch and bound process, we first describe the decomposition algorithm to solve a single node since the branching rules of branch and bound process depend the solutions of the decomposition algorithm.

At a given node $q$ of the branch-and-bound tree, we solve the following problem $(P_q)$

using generalized-Benders decomposition with Benders cuts and Lagrangean cuts.

$$(P_q) \quad \min \quad c^T x + \sum_{\omega \in \Omega} \tau_\omega d_\omega^T y_\omega \tag{5.2a}$$

$$A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{5.2b}$$

$$A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{5.2c}$$

$$x \in X_q, \quad X_q = \left\{ x : x_i \in \{0,1\}, \forall i \in I_1, \;\; x_q^{lb} \leq x \leq x_q^{ub} \right\} \tag{5.2d}$$

$$y_\omega \in Y_\omega \quad \forall \omega \in \Omega, \quad Y_\omega = \left\{ y_\omega : y_{\omega j} \in \{0,1\}, \forall j \in J_1, \;\; 0 \leq y_\omega \leq y_\omega^{ub} \right\} \tag{5.2e}$$

where $x_q^{ub}$ and $x_q^{lb}$ are the upper and lower bounds for the first-stage decisions at node $q$, respectively. The only difference between problem $(P_q)$ and problem $(P)$ is that we have branched on the first stage variables $x$. The domain of $x$ at node $q$ is represented by set $X_q$.

Before we go to the steps of the proposed decomposition algorithm, we define the subproblems that are used in the proposed decomposition algorithm, i.e., the regular rectangles in Figure 5.1. We have both Benders iterations and Lagrangean iterations, which are solved separately. We use letter $k$ to denote the Benders iteration number. We use letter $l$ to denote the Lagrangean iteration number. Note that the Benders iteration number is *independent* of the Lagrangean iteration number, which will become clear when we describe the steps of the decomposition algorithm in Algorithm 1.

**Lagrangean subproblems**

In order to define the Lagrangean subproblems, problem $(P_q)$ is first reformulated as $(PNAC_q)$ by adding nonanticipativity constraints (NACs) in equation (5.3d), where $x_{\omega_i}$ is used to represent the first stage variables corresponding to the $i$th scenario. There are different ways to express NACs. In (5.3d), all the NACs contain $x_{\omega_1}$, which is shown to be advantageous based on the empirical study of Oliveira et al. (2013).

$$(PNAC_q) \quad \min \quad \sum_{\omega \in \Omega} \tau_\omega \left( c^T x_\omega + d_\omega^T y_\omega \right) \tag{5.3a}$$

$$A_0 x_\omega \geq b_0, \quad g_0(x_\omega) \leq 0 \tag{5.3b}$$

$$A_{1,\omega}x_\omega + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{5.3c}$$

$$x_{\omega_1} = x_{\omega_2}, x_{\omega_1} = x_{\omega_3}, \cdots, x_{\omega_1} = x_{\omega_{|\Omega|}} \tag{5.3d}$$

$$x_\omega \in X_q, \quad y_\omega \in Y_\omega \quad \forall \omega \in \Omega \tag{5.3e}$$

The $NACs$ can be dualized by multiplying the constraints, $x_{\omega 1} = x_{\omega+1}, \omega = \omega_1, \omega_2, \cdots, \omega_{|\Omega|-1}$ by $\pi^l_\omega$. The Lagrangean subproblem at iteration $l$ of Lagrangean decomposition at node $q$ is defined as follows:

$$(SL^l_\omega)^q : \quad \min \quad z^{l,q}_{SL,\omega} = \tau_\omega(c^T x_\omega + d^T_\omega y_\omega) + \mu^l_\omega x_\omega \tag{5.4a}$$

$$s.t. \quad A_0 x_\omega \geq b_0, \quad g_0(x_\omega) \leq 0 \tag{5.4b}$$

$$A_{1,\omega}x_\omega + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \tag{5.4c}$$

$$x_\omega \in X_q \tag{5.4d}$$

$$y_\omega \in Y_\omega \tag{5.4e}$$

where $\mu^l_{\omega 1} = \sum_{\omega=\omega 1}^{\omega_{|\Omega|-1}} \pi^l_\omega$, $\mu^l_{\omega+1} = -\pi^l_\omega$, $\omega = \omega_1, \omega_2, \cdots, \omega_{|\Omega|-1}$. In this chapter, the Lagrangean multipliers are updated using the subgradient method (Oliveira et al., 2013).

Let $z^{*l,q}_{SL,\omega}$ be the optimal objective value of the Lagrangean subproblem $(SL^l_\omega)^q$. $\sum_\omega z^{*l,q}_{SL,\omega}$ yields a lower bound of problem $(P_q)$. Lagrangean cuts $\eta_\omega \geq z^{*l,q}_{SL,\omega} - \mu^l_\omega x$ can be added to the Benders master problem after each Lagrangean iteration as proved by Ogbe and Li (2018), Li and Grossmann (2018a).

Let set $L$ be the set of Lagrangean iteration numbers, i.e., we solve the Lagrangean subproblems for $|L|$ iterations. Note that the Lagrangean iterations are separate from the Benders iterations. The only interaction between Lagrangean and Benders is that Lagrangean cuts are added to the Benders master problem. We will not start the Benders iterations until the Lagrangean subproblems are solved for $|L|$ iterations.

**Benders master problem**

The Benders master problem at the $k$th Benders iteration at node $q$ is defined as follows,

$$(MB_k)^q: \quad \min \quad z_{MB}^{k,q} = \sum_{\omega} \eta_\omega \tag{5.5a}$$

$$s.t. \quad A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{5.5b}$$

$$\eta_\omega \geq z_{SL,\omega}^{*l,q} - \mu_\omega^l x \quad \forall \omega \in \Omega, l \in L \tag{5.5c}$$

$$\eta_\omega \geq z_{SB,\omega}^{*k',q} + (\lambda_\omega^{k'})^T (x - \tilde{x}^{k'}) + \tau_\omega c^T x \quad \forall \omega \in \Omega, k' < k \tag{5.5d}$$

$$x \in X_q, \quad \eta_\omega \in \mathbb{R} \quad \forall \omega \in \Omega \tag{5.5e}$$

where (5.5c) are Lagrangean cuts, (5.5d) are Benders cuts. Note that in each Benders iteration $k$, the Lagrangean cuts are added for all Lagrangean iterations $l \in L$ for all scenario $\omega$. The Lagrangean cuts are derived by solving the Lagrangean subproblems as described before. The Benders cuts are added for all Benders iterations $k' < k$, and all scenarios $\omega \in \Omega$. The Benders cuts are derived by solving the Benders subproblems, which will be described next.

Let $z_{MB}^{*k,q}$ be the optimal objective value of the Benders master problem. $z_{MB}^{*k,q}$ is a valid lower bound of problem $(P_q)$. Let $\tilde{x}^k$ be the optimal solution of problem $(MB_k)^q$.

**Benders subproblems**

As we have discussed in the introduction section, a valid Benders cut cannot be derived by simply fixing $x$ at $\tilde{x}^k$ and solving the rest of the subproblems because of the nonconvexities in stage 2. The nonconvexities in the second stage lie in the nonconvex functions $g_{1,\omega}$, and the integrality constraints on some of the $y_\omega$ variables. In order to derive Benders cuts, we need to have convex relaxations for the second stage constraints. $g_{1,\omega}(y_\omega)$ can be relaxed by some convex function $\tilde{g}_{1,\omega}(y_\omega, t_\omega)$ where $t_\omega$ are the additional variables that are introduced for the convex relaxations. Note that the variables $t_\omega$ can be mixed-binary. For example, we may relax a blinear term by its piecewise McCormick envelope where both continuous and binary variables need to be introduced. Without loss of generality, we assume that $J_2$ is the index set where $(t_\omega)_j$, $j \in J_2$ are the binary variables in $(t_\omega)$. If $(t_\omega)$ are all continuous, set

$J_2$ is an empty set.

After $g_{1,\omega}$ is convexified with $\tilde{g}_{1,\omega}$, the rest of the problem may still have integrality constraints, i.e., a convex MINLP or an MILP. In order to have a continuous convex relaxation, one alternative is to simply relax the integrality constraints. Another alternative is to use the convexification schemes for convex MINLP/MILP subproblems. MILP subproblems can be convexified by parametric cuttting planes such as Gomory mixed-integer cut (Balas et al., 1996), lift-and-project cut (Balas et al., 1993), RLT (Sherali and Adams, 1990). Convex MINLP subproblems can be convexified by rank-one lift-and-project cuts (Kılınç et al., 2017; Li and Grossmann, 2018a).

In this chapter, only rank-one lift-and-project cuts for MILP subproblems are implemented. Here, we assume that the relaxed constraints $\tilde{g}_{1,\omega}$ are linear. For lift-and-project cuts for convex nonlinear constraints, we refer the readers to Li and Grossmann (2018a). To simplify the notation, we denote the constraints $A_{1,\omega}x + \tilde{g}_{1,\omega}(y_\omega, t_\omega) \leq b_{1,\omega}$, $0 \leq y \leq y^{ub}$, $x_q^{lb} \leq x \leq x_q^{ub}$ as $\tilde{A}_q \begin{pmatrix} x \\ y_\omega \\ t_\omega \end{pmatrix} \geq \tilde{b}_q$. Suppose $\tilde{x}^k$ is the optimal solution of the Benders master problem at iteration $k$. $\tilde{y}_\omega^k$, $\tilde{t}_\omega^k$ is the optimal solution of Benders subproblem before adding the lift-and-project cuts generated at iteration $k$. If $(\tilde{y}_\omega^k)_j, j \in J_1$ is fractional, rank-one lift-and-project cuts are generated by solving the following cut generating linear program $(CGLP_j)^q$ (Balas et al., 1993).

$$(CGLP_j)^q \quad \min \quad \alpha^x \tilde{x}^k + \alpha^y \tilde{y}_\omega^k + \alpha^t \tilde{t}_\omega^k - \beta \tag{5.6a}$$

$$\alpha = u^T \tilde{A}_q - u_0 e_{j+n}, \quad \alpha = v^T \tilde{A}_q + v_0 e_{j+n} \tag{5.6b}$$

$$\beta = u^T \tilde{b}_q, \quad \beta = v^T \tilde{b}_q + v_0 \tag{5.6c}$$

$$u^T e + v^T e + u_0 + v_0 = 1 \tag{5.6d}$$

$$u, v, u_0, v_0 \geq 0 \tag{5.6e}$$

Note that we can also generate lift-and-project cuts corresponding to the binary variables in

$t_\omega$. We only need to change equation (5.6b) to

$$\alpha = u^T \tilde{A}_q - u_0 e_{j+n+m}, \quad \alpha = v^T \tilde{A}_q + v_0 e_{j+n+m} \tag{5.7}$$

to generate the rank-one cut corresponding to $(t_\omega)_j$, $j \in J_2$. The newly generated cuts are then added to the Benders subproblem $(SB_\omega^k)^q$ at iteration $k$ for scenario $\omega$ defined as follows,

$$(SB_\omega^k)^q : \quad \min \quad z_{SB,\omega}^k = \tau_\omega d^T y_\omega \tag{5.8a}$$

$$x = \tilde{x}^k \tag{5.8b}$$

$$A_{1,\omega} x + \tilde{g}_{1,\omega}(y_\omega, t_\omega) \leq b_{1,\omega} \tag{5.8c}$$

$$\alpha_{k'c}^x x + \alpha_{k'c}^y y_\omega + \alpha_{k'c}^t t_\omega \geq \beta_{k'c} \quad \forall k' \leq k, c \in C^{k'} \tag{5.8d}$$

$$0 \leq y \leq y_\omega^{ub} \tag{5.8e}$$

where (5.8d) are the cutting planes that are generated to convexify the subproblem. Set $C^{k'}$ represents the set of cuts that are generated at iteration $k'$. Note that we accumulate the cuts generated in all iterations $k' \leq k$.

**Upper bound procedure**

There are multiple ways to obtain a feasible solution to the original problem. A simple way would be fixing the first stage decisions and solving the upper bound subproblem $(UB_\omega)$ for each scenario $\omega$ separately,

$$(UB_\omega) : \quad \min \quad z_{UB,\omega} = \tau_\omega d_\omega^T y_\omega \tag{5.9a}$$

$$s.t. \quad g_{1,\omega}(y_\omega) \leq b_{1,\omega} - A_{1,\omega} \tilde{x} \tag{5.9b}$$

$$g_{2,\omega}(y_\omega) \leq b_{2,\omega} \tag{5.9c}$$

$$y_\omega \in Y_\omega \tag{5.9d}$$

where the value of the first stage decisions is fixed at $\tilde{x}$. The choice of $\tilde{x}$ decides the upper bound that is obtained from $(UB_\omega)$. Here we propose three heuristics to obtain a "good" $\tilde{x}$.

- Let $\tilde{x} = \tilde{x}^k$, i.e., be the optimal solution that is obtained in the Benders master problem $(MB_k)^q$ at iteration $k$.

- Let $x^{*l,q}_{\omega 1}, x^{*l,q}_{\omega_2}, \cdots, x^{*l,q}_{\omega_{|\Omega|}}$ be the optimal solution for the Lagrangean subproblems $(SL^l_\omega)^q$ at iteration $l$. We define

$$x^{*l,q}_{avg} = \sum_{\omega \in \Omega} \tau_\omega x^{*l,q}_\omega$$

  as the probability weighted average of the optimal first stage solutions over all scenarios. Then we select scenario $\omega^*$, such that

$$\omega^* = \arg\min_{\omega \in \Omega} \sum_{i \in I} \left( \frac{(x^{*l,q}_{avg})_i - (x^{*l,q}_\omega)_i}{(x^{ub})_i} \right)^2$$

  $\omega^*$ is the scenario where the optimal solution $x^{*l,q}_\omega$ has the smallest distance with the weighted average $x^{*l}_{avg}$. Note that in calculating the distances all the first stage variables are scaled by the distances of their original upper and lower bounds (the original lower bounds are assumed to be zero in problem $(P)$). We can fix $\tilde{x}$ at $x^{*l,q}_{\omega^*}$ and solve $(UB_\omega)$ to obtain a feasible solution.

- Randomly select one scenario $\omega$ and fix the first stage decisions at the optimal value of the Lagrangean subproblem corresponding to this scenario, i.e., $x^{*l,q}_\omega$.

**Remark.** The first upper bound procedure obtains good feasible $\tilde{x}$ from Benders decomposition. In the second and the third procedure, the good feasible $\tilde{x}$ comes from the solutions of the Lagrangean subproblems. Therefore, all the upper bound procedures can be applied simultaneously.

### The steps of the decomposition algorithm to solve a single node

With the definitions of the subproblems, we outline the steps of the decomposition algorithm to solve a single node $q$ in Algorithm 1. The decomposition algorithm is defined as a *solvenode*$(q)$ function. First, Lagrangean subproblems $(SL^l_\omega)^q$ are solved for all Lagrangean iterations $l \in L$. We add all the Lagrangean cuts (5.5c) to the Benders master problem

**Algorithm 1:** Algorithm to solve a single node

```
Function solvenode(q)
    /* Lagrangean iterations */
    for l = 1, 2, ··· , L do
        for ω = ω_1, ω_2, ··· , ω_|Ω| do
            Solve Lagrangean subproblems (SL^l_ω)^q. Let z^{*l,q}_{SL,ω} be the optimal
              objective value
        end
        Update LB_q = max(LB_q, Σ_ω z^{*l,q}_{SL,ω}).
        Update UB_q using the upper bound procedure.
        if UB_q − LB_q ≤ ϵ then
            return q
        end
        Update Lagrangean multipliers μ^l_ω with the subgradient method
    end
    Add Lagrangean cuts (5.5c) to Benders master problem (MB_1)^q.
    /* Benders iterations */
    for k = 1, 2, ··· , K do
        Solve (MB_k)^q. Let z^{*k,q}_{MB} be the optimal objective value of (MB_k)^q. Set
          LB_q = z^{*k,q}_{MB}
        for ω = ω_1, ω_2, ··· , ω_|Ω| do
            Solve Benders subproblems (SB^k_ω)^q. Let y^*_ω, t^*_ω be the optimal value.
            if there is fractional (y^*_ω)_j for j ∈ J_1, (t^*_ω)_j for j ∈ J_2 then
                Generate cutting planes (5.8d) by solving (CGLP_j)^q and add to
                  (SB^k_ω)^q.
                Solve (SB^k_ω)^q again with the newly generated cutting planes.
            end
            Generate Benders cuts (5.5d) and add to Benders master problem
              (MB_k)^q.
        end
        Update UB_q with the upper bound procedure.
        if UB_q − LB_q ≤ ϵ then
            return q
        end
    end
    return q.
end
```

$(MB_1)^q$ (the Benders master problem in the first iteration at node $q$). Then we iteratively solve the Benders master problem $(MB_k)^q$ and the Benders subproblems $(SB_\omega^k)^q$ with cutting planes. After each Benders iteration, we add Benders cuts (5.5d) to the Benders master problem. The proposed upper bound procedure can be used to obtain feasible solutions based on the optimal solutions of the Lagrangean subproblems or the Benders master problem. We can terminate if the node is solved to optimality. Otherwise, the algorithm terminates by setting the maximum Lagrangean and Benders iterations to $|L|$ and $|K|$, respectively. The decomposition algorithm always returns an upper bound $UB_q$ and a lower $LB_q$ at node $q$.

### 5.3.3 Branch and bound

If we apply the function $solvenode(q)$ described in Algorithm 1 to the rootnode (problem $(P)$), we cannot guarantee to solve it to global optimality because of the duality gap. That is why we need to branch and bound on the first stage decisions as in Carøe and Schultz (1999), Cao and Zavala (2017), Kannan (2018). The spatial branch and bound corresponds to the rounded rectangles in Figure 5.1. Most importantly, we need to define the node selection rule and the branching rule used in the branch and bound process. Let $\Gamma$ be the set of active nodes.

**Node selection rule**

Select node $q$ such that $q = \arg\min_{q \in \Gamma} LB_q$.

After a node is selected, we branch on one of the first stage variables of this node. The branching rules are defined according to the bounds of the first stage variables and the solutions obtained by the proposed decomposition algorithm (Algorithm 1) at this node.

**Branching rules**

1. Select the first stage variable with the largest normalized relative diameter,

$$i^* = \arg\max_{i \in I} \frac{(x_q^{ub})_i - (x_q^{lb})_i}{(x_{q_0}^{ub})_i - (x_{q_0}^{lb})_i} \delta_i$$

$q_0$ represents the root node. $\delta_i$ is a normalization factor for variable $i$. Two new nodes $q_1$ and $q_2$ are then created.

$(x^{ub}_{q_j})_i = (x^{ub}_q)_i, \quad (x^{lb}_{q_j})_i = (x^{lb}_q)_i \quad j \in \{1,2\}, i \neq i^*$

$(x^{lb}_{q_1})_{i^*} = (x^{lb}_q)_{i^*}, \quad (x^{ub}_{q_1})_{i^*} = \frac{1}{2}\Big((x^{lb}_q)_{i^*} + (x^{ub}_q)_{i^*}\Big)$

$(x^{lb}_{q_2})_{i^*} = \frac{1}{2}\Big((x^{lb}_q)_{i^*} + (x^{ub}_q)_{i^*}\Big), \quad (x^{ub}_{q_2})_{i^*} = (x^{ub}_q)_{i^*},$

where the domain of variable $i^*$ is bisected. 2. Let $(x^*_q, y^*_{q\omega_1}, y^*_{q\omega_2}, \cdots, y^*_{q\omega_{|\Omega|}})$ be the best feasible solution found at node $q$. Then we select variable $i^*$ where best feasible first stage variable has the largest normalized distance to its bounds,

$$i^* = \arg\max_{i \in I} \frac{\min\left\{(x^{ub}_q)_i - (x^*_q)_i, (x^*_q)_i - (x^{lb}_q)_i\right\}}{(x^{ub}_{q_0})_i - (x^{lb}_{q_0})_i}\delta_i$$

Two new nodes $q_1$ and $q_2$ are then created.

$(x^{ub}_{q_j})_i = (x^{ub}_q)_i, \quad (x^{lb}_{q_j})_i = (x^{lb}_q)_i \quad j \in \{1,2\}, i \neq i^*$

$(x^{lb}_{q_1})_{i^*} = (x^{lb}_q)_{i^*}, \quad (x^{ub}_{q_1})_{i^*} = (x^*_q)_{i^*}$

$(x^{lb}_{q_2})_{i^*} = (x^*_q)_{i^*}, \quad (x^{ub}_{q_2})_{i^*} = (x^{ub}_q)_{i^*},$

where the domain of $(x)_{i^*}$ is divided based on the best feasible solution found on node $q$.

3. Let $x^{avg}_q$ be the weighted average of first stage decisions obtained from Lagrangean subproblems in the iteration where the tightest lower bound can be found. Then we select variable $i^*$ such that

$$i^* = \arg\max_{i \in I} \frac{\min\left\{(x^{ub}_q)_i - (x^{avg}_q)_i, (x^{avg}_q)_i - (x^{lb}_q)_i\right\}}{(x^{ub}_{q_0})_i - (x^{lb}_{q_0})_i}\delta_i$$

Similarly, two new nodes $q_1$ and $q_2$ are then created.

$(x^{ub}_{q_j})_i = (x^{ub}_q)_i, \quad (x^{lb}_{q_j})_i = (x^{lb}_q)_i \quad j \in \{1,2\}, i \neq i^*$

$(x^{lb}_{q_1})_{i^*} = (x^{lb}_q)_{i^*}, \quad (x^{ub}_{q_1})_{i^*} = (x^{avg}_q)_{i^*}$

$(x^{lb}_{q_2})_{i^*} = (x^{avg}_q)_{i^*}, \quad (x^{ub}_{q_2})_{i^*} = (x^{ub}_q)_{i^*},$

With the node selection rule and the branching rules, we outline the steps of the proposed

**Algorithm 2:** Generalized-Benders decomposition-based branch and cut algorithm

```
/* Initialization                                                    */
Initialize UB = +∞, LB = −∞ Create root node q_0. Set X_{q_0} = X. Set all the
  Lagrangean multipliers μ_ω to zero. solvenode(q_0). Create a node list Γ = {q_0}.
while Γ ≠ ∅ do
    Select node q such that q = arg min_{q∈Γ} LB_q.
    Apply one of the proposed branching rules and create two child node of q,
      denoted as q_1, q_2. Branching rule 1 should be applied at least once after a finite
      number of iterations. Let Γ = Γ\{q}
    solvenode(q_1); solvenode(q_2).
    Let UB = min_{q∈Γ} UB_q, LB = min_{q∈Γ} LB_q.
    for q ∈ Γ do
        if UB_q − LB_q ≤ ϵ then
            Γ = Γ\{q}   /* Fathom by optimality                      */
        end
        if LB_q − UB ≥ ϵ then
            Γ = Γ\{q}   /* Fathom by bound                           */
        end
    end
end
```

generalized-Benders decomposition-based branch and cut algorithm in Algorithm 2.

**The steps of generalized-Benders decomposition-based branch and cut algorithm**

At a high level, the proposed algorithm is performing spatial branch and bound search that includes node selection and branching, node fathom. The steps of the spatial branch and bound are described in Algorithm 2. The branch and bound algorithm selects a node with the tightest lower bound. Different branching rules described above can be applied. However, branching rule 1 must be applied at least once after a finite number of iterations to have the theoretical guarantee of convergence, which will be discussed in detail in section 5.4. Each node in the branch and bound process is solved with the generalized Benders decomposition-based algorithm described in Algorithm 1. The flowchart of the algorithm is shown in Figure 5.1 where the rounded rectangles describe the high-level spatial branch-and-

bound algorithm (Algorithm 2); the regular rectangles describe the decomposition algorithm that solves a single node (Algorithm 1).

**Remark.** The Benders cuts and Lagrangean cuts in a parent node can be inherited by its child nodes. The parametric cutting planes in the Benders subproblems of a parent node can also be inherited by its child nodes. Therefore, the solution process of the child nodes can be warm-started by inheriting those cuts.

**Remark.** After branching, the Lagrangean multipliers of a child node can be either initialized to zero or initialized to the multipliers that give the tightest lower bound for its parent node.

## 5.4 Convergence of the proposed algorithm

Cao and Zavala (2017) and Kannan (2018) have proved the convergence of their algorithms, both of which are Lagrangean decomposition-based branch and bound algorithm. Both proofs rely on Chapter IV of Horst and Tuy (2013). Since we also have Lagrangean cuts in the Benders master problem, the lower bound that we can obtain at each node would be at least as tight as Lagrangean decomposition as proved by Li and Grossmann (Proposition 2 of Li and Grossmann (2018a)). Therefore, it would be straightforward to prove convergence based on previous work.

In order to make this chapter self-contained, we briefly describe the convergence proof for our proposed algorithm. We first define some notations for the branch and bound algorithm. $D$ represents the feasible region of the first stage decisions. $D = \{x | \exists (x, y_{\omega 1}, y_{\omega_2}, \cdots, y_{\omega|\Omega|})$ feasible for problem $(P)\}$. Since we have assumed that the problem has relatively complete recourse, we can let $D = \{x | A_0 x \geq b_0, g_0(x) \leq 0, x \in X\}$. The domain of the first stage decisions at node $q$ is defined as $X_q$ in equation (5.2d). $X_{q_0}$ is the domain of $x$ at the root node. $X_{q_p}$ denotes the domain of $x$ of a node at level $p$ of the tree. A path in the branch and bound tree from the rootnode to one leaf node has one node in each level $p$ of the tree, which can be represented as $\{X_{q_p}\}$. $q_{p+1}$ is a child of $q_p$ and $X_{q_{p+1}} \subset X_{q_p}$. $\delta(X_q)$

represents the diameter of $X_q$. $\delta(X_q) = ||x_q^{ub} - x_q^{lb}||_\infty$. $UB_q$ and $LB_q$ are upper and lower bound obtained at node $q$, respectively. In order to prove convergence, we need to prove $\lim_{q\to\infty} UB_q = \lim_{q\to\infty} LB_q = z^*$.

**Definition 1.** A subdivision is called exhaustive if $\lim_{p\to\infty} \delta(X_{q_p}) = 0$, for all decreasing subsequences $X_{q_p}$ generated by the subdivision (Definition IV.10 in Horst and Tuy (2013)).

**Lemma 1.** The subdivision process of the proposed algorithm is exhaustive.

*Proof.* Since branching rule 1 is applied at least once after a finite number of iterations and the bounds for each variable at the rootnode and the normalization factors $\delta_i$ are all finite, the variable $i$ with the largest diameter is divided by half once after a finite number of iterations. Therefore, $\delta(X_{q_p}) = ||x_{q_p}^{ub} - x_{q_p}^{lb}||_\infty$ is divided by half once after a finite number of iterations. $\lim_{p\to\infty} \delta(X_{q_p}) = 0$ (see Lemma 3.5.1 in Kannan (2018)). □

**Definition 2.** A selection operation is said to be bound improving if, after a finite number of steps, at least one partition element where the actual lower bounding is attained is selected for further partition. (Definition IV.6 in Horst and Tuy (2013))

**Lemma 2.** The selection operation of the proposed algorithm is bound improving.

*Proof.* This is obvious from the node selection rule used by the proposed algorithm. □

**Definition 3.** The "deletion by infeasibility" rule throughout a branch and bound procedure is called certain in the limit if, for every infinite decreasing sequence $\{X_{q_p}\}$ of successively refined partition elements with limit $\bar{X}$, we have $\bar{X} \cap D \neq \emptyset$. (Definition IV.8. in Horst and Tuy (2013))

**Lemma 3.** Deletion by infeasibility is certain in the limit in the proposed algorithm.

*Proof.* Since branching rule 1 is applied at least once after a finite number of iterations in the proposed branch and bound algorithm, any infinite decreasing sequence $\{X_{q_p}\}$ would converge to a point $\bar{x}$. We prove this by contradiction. Suppose $\bar{x} \notin D$. Since D is compact, there exist $r > 0$, such that $\{x|\ ||x - \bar{x}||_2 \leq r\} \cap D = \emptyset$. Therefore, $\exists p_0$, such that $\forall p \geq p_0$,

$X_{q_p} \cap D = \emptyset$. $X_{q_{p_0}} \cap D = \emptyset$, which means the sequence $\{X_{q_p}\}$ should have been fathomed at $X_{q_{p_0}}$. We complete the proof by contradiction. (Similar proof in Lemma 2 of Cao and Zavala (2017)) □

**Definition 4.** A lower bounding operation is called strongly consistent if, at every iteration, any undeleted partition set can be further refined and if any infinite decreasing sequence $\{X_{q_p}\}$ successively refined partition elements contains a sub-sequence $\{X_{q_{p'}}\}$ satisfying $\bar{X} \cap D \neq \emptyset$, $\lim_{p\to\infty} LB_{q_p} = z^*(\bar{X} \cap D)$, where $\bar{X} = \cap_p X_{q_p}$. (Definition IV.7. in Horst and Tuy (2013))

**Lemma 4.** The proposed algorithm is strongly consistent.

*Proof.* Since the division is exhaustive, the sequence $\{X_{q_p}\}$ would converge to a singleton $\bar{x}$. From Lemma 3, $\bar{x} \in D$. Note that the lower bound $LB_{q_p}$ is at least as tight as $\sum_\omega z^{*l,q_p}_{SL,\omega}$.

$$\begin{aligned}\lim_{p\to\infty} LB_{q_p} \geq \lim_{p\to\infty} \sum_{\omega\in\Omega} z^{*l,q_p}_{SL,\omega} = \quad & \sum_{\omega\in\Omega} \min \quad \tau_\omega(c^T\bar{x}_\omega + d^T_\omega y_\omega) + \mu^l_\omega \bar{x}_\omega \\ & s.t. \quad A_{1,\omega}\bar{x}_\omega + g_{1,\omega}(y_\omega) \leq b_{1,\omega}, \quad y_\omega \in Y_\omega \quad \forall\omega \in \Omega \\ & = z^*(\bar{X} \cap D)\end{aligned}$$

The last equality holds because the $NACs$ are satisfied. Therefore, $\lim_{p\to\infty} LB_{q_p} = z^*(\bar{X} \cap D)$ □

Now we are ready to prove the convergence of the proposed algorithm.

**Theorem 3.** The proposed algorithm is convergent, i.e., $\lim_{q\to\infty} LB_q = \lim_{q\to\infty} UB_q = z^*$.

*Proof.* Since we have proved that the subdivision process is exhaustive (Lemma 1), "deletion by infeasibility" is certain in the limit (Lemma 3), and the lower bounding operation is strongly consistent (Lemma 4), we have that the lower bounding operation is consistent according to Lemma IV.5 in Horst and Tuy (2013). Since we have proved that the selection operation is bound improving, the branch and bound procedure is convergent according to Theorem IV.3 in Horst and Tuy (2013). □

## 5.5 Implementation

The proposed algorithm is implemented as part of the PlasmoAlgorithms package in Julia programming language. PlasmoAlgorithms is a package for decomposition algorithms that use PlasmoGraph (Jalving et al., 2017) as input. PlasmoGraph is a graph based data structure based on JuMP/Julia. Each node in a PlasmoGraph contains an optimization model written in JuMP. In this case, in order to access our algorithm, the user only needs to write the first and second stage problems in different nodes in PlasmoGraph and connect the first and second stage nodes with linking constraints. A brief tutorial is provided in Brunaud et al..

The default solver for the Lagrangean subproblems, upper bound subproblems, which are MINLPs or NLPs, is BARON. The default solver for the MILP Benders master problem and the LP Benders subproblems is CPLEX.

Rank-one lift-and-project cuts are implemented as the routine to convexify the MILP subproblems after the nonconvex functions in stage 2 are replaced by their convex relaxations. Numerical issues can occur if the rank-one cuts are not implemented properly especially for problems that are not scaled well. To avoid the numerical instability, we do not generate all the theoretically valid cuts. We only generate a rank-one cut when the optimal solution of the binary variables in solving the subproblems are between 0.01 and 0.99. A cut will not be added if the absolute value of any coefficient in the optimal solution of $\alpha$ in $(CGLP_j)^q$ is nonzero but less than $10^{-6}$. A cut will not be added if any of the dual variables correspond to equation (5.6c) is less than 0.01.

The maximum Lagrangean and Benders iterations $|L|$ and $|K|$ can affect the efficiency of the proposed algorithm. The general guideline to set the maximum number of iterations is to check whether increasing the maximum number of iterations can improve the optimality gap or not. Therefore, the optimal $|L|$ and $|K|$ are problem-dependent. A trial and error procedure is usually needed, i.e., we need to solve a small scale problem with Benders decomposition and Lagrangean decomposition separately to decide the maximum number of iterations. The $|L|$ and $|K|$ used in each test case in section 5.6 will be reported.

As for branching rules in the proposed algorithm, branching rule 1 is used if the iteration number in the branch and bound process is a multiple of 3. Otherwise, branching rule 2 is used.

## 5.6 Computational results

The proposed algorithm is tested with three problems of this category. A brief description and the computational results for the three problems are described in the subsections below. In all the three problems, we solve the deterministic equivalent of the stochastic programs with three different global solvers, BARON (Tawarmalani and Sahinidis, 2005), ANTIGONE (Misener and Floudas, 2014), and SCIP (Achterberg, 2009). All the problems are solved using one processor of an Intel Xeon (2.67GHz) machine with 64 GB RAM. The computational results of solving the deterministic equivalent are shown in Tables 5.1, 5.3, 5.5. We also use three different decomposition algorithms to solve each problem shown in Tables 5.2, 5.4, and 5.6. GBD(with cuts) + L represents the proposed algorithm, where each node of the branch and bound tree is solved by generalized Benders decomposition with Lagrangean cuts and Benders cuts coming from the subproblems convexified by cutting planes. GBD+L is similar to the proposed algorithm, but cutting planes are not used to convexified the Benders subproblems. LD represents the Lagragean decomposition-based branch and bound algorithm proposed by Carøe and Schultz (1999). The tolerance of the relative optimality gap is set to 0.1%. The walltime limit is set to $10,000$ seconds.

### 5.6.1 Stochastic pooling problem with contract selection

The stochastic pooling problem with contract selection is an extension of the stochastic pooling problem studied by Li et al. (2011a). The first stage decisions are design decisions, which include the selection and capacity of feeds and pools. Note that a binary variable is used to model whether a feed or pool is selected or not. Continuous variables are used to represent the capacity of the feeds and pools. The second stage decisions are mainly

operating decisions, including the mass flowrates that are subject to product quality specifications. Moreover, we consider purchasing the feeds with three different types of contracts, i.e., fixed price, discount after a certain amount, and bulk discount (Park et al., 2006). A binary variable is used to model which contract to select for each feed. The details of the mathematical formulation are described in Appendix 1.

The uncertainties in this problem include demands of the products, prices of the feeds, and selling prices of the products. We assume that all the three uncertain parameters can have high, medium, or low realizations. We create instances of stochastic programs with 3, 9, and 27, scenarios. In the 3-scenario problem, only the demands of the products are uncertain. In the 9 scenario-problem, we consider the Cartesian product of the demands of the feeds and the prices of the feeds. In the 27-scenario problem, the Cartesian product of all the three uncertain parameters are considered.

This problem has 9 binary variables, 9 continuous variables, and 18 linear constraints in stage 1. There are 35 binary variables, 112 continuous variables, 116 linear constraints, 22 nonlinear constraints in stage 2 per scenario. The nonlinear constraints correspond to the bilinear terms in the pooling problem.

The computational results of the deterministic deterministic equivalent are shown in Table 5.1. While BARON and ANTIGONE can solve the problems with 3 and 9 scenarios to optimality within the time limit, all the solvers fail to solve the problem with 27 scenarios to optimality within the time limit.

Table 5.1: Walltime and relative optimality gap of solving the deterministic equivalent of the stochastic pooling problem with 3, 9, 27, scenarios

| #Scenarios | 3 | 9 | 27 |
|---|---|---|---|
| BARON 18.5.8 | 5/0.1% | 3005/0.1% | $10^4$/8.7% |
| ANTIGONE 1.1 | 16/0.1% | 251/0.1% | $10^4$/1.4% |
| SCIP 5.0 | $10^4$/54.4% | $10^4$/100.0% | $10^4$/100.0% |

We also apply the three decomposition algorithms, GBD(with cuts)+L, GBD+L, LD, to solve this problem. The maximum Lagrangean iterations $|L|$ and the maximum Benders iterations $|K|$ at each node is set to 20, and 60, respectively. The walltime, the relative

Table 5.2: Walltime, relative optimality gap, and number of nodes by using different decomposition algorithms to solve the stochastic pooling problem with 3, 9, 27, scenarios

| #Scenarios | 3 | 9 | 27 |
|---|---|---|---|
| GBD(with cuts)+L | 152/0.1%/1 | 502/0.1%/1 | 2113/0.1%/1 |
| GBD+L | $10^4$/0.1%/381 | $10^4$/0.8%/39 | $10^4$/1.3%/7 |
| LD | $10^4$/0.2%/363 | $10^4$/7.1%/43 | $10^4$/12.2%/9 |

optimality gap, and the number of nodes in the branch and bound tree are shown in Table 5.2 separated by "/". The proposed algorithm, GBD(with cuts)+L, can solve all the problems to optimality at the root node within the time limit. GBD+L can solve the 3 scenario problem with 381 nodes. However, for the problems with 9 and 27 scenarios, GBD+L cannot close the gap but can provide solutions with reasonably good optimality gaps, namely 0.8% and 1.3%, respectively. LD runs out of time for all the problems. For the 3 scenario problem, LD provides 0.2% gap by solving 363 nodes. For the 9 and 27 scenario problem, LD is only able to solve 43 and 9 nodes, respectively, within the time limit. Therefore, the gaps are quite large for LD in large problems.

### 5.6.2 Crude selection and refinery optimization under uncertainty

The crude selection and refinery optimization problem comes from the stochastic programming library in GOSSIP (decomposition software for the Global Optimization of nonconvex two-Stage Stochastic mixed-Integer nonlinear Programs) (Kannan, 2018). The problem was first proposed by Yang and Barton (2016). In their model, crudes are purchased in certain predefined discrete amounts in the first stage. Kannan (2018) extend this work by considering the purchased crudes as 'semi-continuous' variables i.e., the crude purchase quantity can either be zero if the decision maker decides not to purchase that particular crude, or it must lie between some prespecified lower and upper bounds. The second stage decisions correspond to the operating conditions of the refinery, such as the mass flow rates and the split fractions. The decisions need to be subject to mass balances, quality specifications, market demands, and capacity and supply restriction. The uncertainties are in

the crude oil qualities and yields. The nonlinearities in the model come from the bilinear terms in the quality specifications in stage 2, which is similar to the formulation in a pooling problem.

This problem has 10 binary variables, 10 continuous variables, and 21 linear constraints, in stage 1. There are 142 continuous variables, 85 linear constraints, 26 nonlinear constraints in stage 2 per scenario. Note that this problem has no binary variables in stage 2.

The two-stage stochastic program for this problem is solved with 5, 10, 20, 40, and 120 scenarios. The computational results of the deterministic equivalent are shown in Table 5.3. The deterministic equivalent with less than 40 scenarios can be solved to optimality within the time limit with all the solvers where ANTIGONE has the best performance. For the largest instance, the 120 scenario problem, the solvers can obtain good gaps, i.e., 0.4% for SCIP.

Table 5.3: Walltime and relative optimality gap of solving the deterministic equivalent of the crude selection and refinery optimization under uncertainty with 5,10,20,40,120 scenarios

| #Scenarios | 5 | 10 | 20 | 40 | 120 |
|---|---|---|---|---|---|
| BARON 18.5.8 | 24/0.1% | 5092/0.1% | $10^4$/0.4% | $10^4$/0.1% | $10^4$/0.9% |
| ANTIGONE 1.1 | 5/0.1% | 26/0.1% | 322/0.1% | $10^4$/0.1% | $10^4$/0.9% |
| SCIP 5.0 | 4/0.1% | 60/0.1% | $10^4$/0.2% | $10^4$/0.1% | $10^4$/0.4% |

Table 5.4: Walltime, relative optimality gap, and number of nodes by using different decomposition algorithms to solve the crude selection and refinery optimization under uncertainty with 5, 10, 20, 40, 120 scenarios

| #Scenarios | 5 | 10 | 20 | 40 | 120 |
|---|---|---|---|---|---|
| GBD+L (Mc) | $10^4$/1.8%/155 | $10^4$/0.8%/85 | $10^4$/1.1%/37 | $10^4$/0.9%/17 | $10^4$/0.9%/7 |
| GBD+L (Mc, 1 node) | 34/2.3% | 47/1.0% | 91/1.1% | 184/0.9% | 473/0.9% |
| GBD+L (pMc, 1 node) | 317/0.4% | 405/0.2% | 389/0.4% | 387/0.9% | 4027/0.2% |
| LD | $10^4$/11.7%/165 | $10^4$/7.8%/81 | $10^4$/16.2%/33 | $10^4$/80.7%/17 | $10^4$/145.3%/7 |

Since there are no binary variables in stage 2, we only test the decomposition algorithms GBD+L and LD. The maximum Lagrangean iterations $|L|$ and the maximum Benders iterations $|K|$ at each node is set to 20, and 60, respectively. The computational results are shown in Table 5.4. For GBD+L, two different convex relaxations are used for the Benders

subproblems. In Table 5.4, we use "Mc" to represent the convex relaxation where the bilinear terms are relaxed by their McCormick envelopes. We use "pMc" to represent the convex relaxation where the bilinear terms are relaxed by the hull relaxation of the piecewise McCormick envelopes (Misener et al., 2011). The details of "pMc" are described in Appendix 2. In the first row of Table 5.4, we show the walltime, the optimality gap, and the number of nodes in the algorithm GBD+L where "Mc" is used as the convex relaxation. None of the problems can be solved within 0.1% optimality within $10,000$ seconds. We also find that the optimality gaps do not improve much when we branch on the first stage variables. Therefore, we show the gap at the rootnode in the second row. One can observe that only in the case of 5 and 10 scenarios branching leads to an improvement in the optimality gaps.

Piecewise McCormick envelop ("pMc") is tighter than "Mc", while yielding larger problems. In the third row, we show the walltime and the optimality gaps for those problems at the rootnode when "pMc" is used as the convex relaxation. The results corresponding to the whole branch and bound process for "pMc" are omitted since the gaps did not improve when we branch within the time limit. The optimality gaps obtained from "pMc" relaxations are smaller than using "Mc" at the rootnode at the expense of spending more time to solve each Benders subproblem. In the largest case of 120 scenarios, GBD+L with "pMc" relaxation is able to obtain smaller optimality gap than using any of the global solvers to solve the deterministic equivalent.

In the fourth row, we report the results of LD where the optimality gaps are larger than using GBD+L. The results indicate that the Lagrangean relaxations for this problem are weak, which is consistent with the results obtained by Kannan (2018) using their modified Lagrangean decomposition-based branch and bound algorithm. The results also suggest that in both "Mc" and "pMc" for GBD+L, the Benders cuts "dominate" the Lagrangean cuts, which is why branching does not have a significant impact on the optimality gaps in GBD+L. In this example, the convex relaxation provided by "Mc" and "pMc" can yield tighter lower bound than the Lagrangean relaxation.

### 5.6.3 Storage design for a multi-product plant under uncertainty

The storage design for a multi-product plant under uncertainty problem also comes from the library in GOSSIP. The case study considers a chemical plant that uses a single reactor to produce products and stores them in storage tanks. The stage 1 decisions are design decisions, i.e., the product tank sizes. In stage 2, binary variables are used to represent the assignment of products to campaigns. Continuous decisions include the production amount in each campaign, the duration of the campaigns, etc. The uncertainties considered are the demands of the products. The details can be found in Rebennack et al. (2011).

This problem has 3 continuous variables, and 1 nonlinear constraint, in stage 1. The nonlinearity corresponds to a univariate signomial term, which is used to calculate the tank investment cost. There are 9 binary variables, 41 continuous variables, 85 linear constraints, 26 nonlinear constraints in stage 2 per scenario. These nonlinear constraints come from the bilinear terms that are used to calculate the variable inventory costs.

The deterministic equivalent of the stochastic programs is solved with 2, 3, 4, 5, 9, and 27 scenarios. From the results in Table 5.5, one can observe that BARON and SCIP can solve the small problems to optimality within the time limit but fail to solve the problem to optimality with more than 4 scenarios. On the other hand, ANTIGONE can solve the 9 and 27 scenario problem, but fails to solve the small problems to optimality within the time limit.

Table 5.5: Walltime and relative optimality gap of solving the deterministic equivalent of the storage design for a multi-product plant under uncertainty with 2,3,4,5,9,27 scenarios

| #Scenarios | 2 | 3 | 4 | 5 | 9 | 27 |
|---|---|---|---|---|---|---|
| BARON 18.5.8 | 1117/0.1% | $10^4$/2.2% | $10^4$/7.0% | $10^4$/11.0% | $10^4$/13.0% | $10^4$/13.0% |
| ANTIGONE 1.1 | $10^4$/2.8% | $10^4$/8.0% | $10^4$/11.0% | $10^4$/13.5% | 3/0.1% | 87/0.1% |
| SCIP 5.0 | 404/0.1% | 7960/0.1% | $10^4$/1.0% | $10^4$/2.4% | $10^4$/4.6% | $10^4$/12.2% |

Similarly, the three decomposition algorithms are used to solve this problem. The maximum Lagrangean iterations $|L|$ and the maximum Benders iterations $|K|$ at each node is set to 1, and 1, respectively. The results are shown in Table 5.6. All the decomposition algorithms can solve this problem to optimality within the time limit. However, in this

Table 5.6: Walltime, relative optimality gap, and number of nodes by using different decomposition algorithms to solve the storage design for a multi-product plant under uncertainty with 2,3,4,5,9,27 scenarios

| #Scenarios | 2 | 3 | 4 | 5 | 9 | 27 |
|---|---|---|---|---|---|---|
| GBD(with cuts)+L | 127/0.1%/47 | 239/0.1%/63 | 761/0.1%/141 | 365/0.1%/59 | 1250/0.1%/99 | 8681/0.1%/159 |
| GBD+L | 74/0.1%/47 | 277/0.1%/149 | 353/0.1%/141 | 188/0.1%/59 | 549/0.1%/99 | 4108/0.1%/159 |
| LD | 61/0.1%/47 | 214/0.1%/149 | 270/0.1%/141 | 152/0.1%/59 | 436/0.1%/99 | 3219/0.1%/165 |

case LD performs better than GBD(with cuts)+L and GBD+L. Note that in most of the instances all the three algorithms have to solve the same number of nodes in order to prove optimality. Only in the case of 3 and 27 scenarios, GBD(with cuts)+L requires fewer nodes than LD, which means that the Benders cuts with cutting planes can help to improve the bounds. The results indicate that in the cases where the three algorithms need to solve the same number of nodes, adding Benders cuts does not help to improve the bounds, i.e., the Benders cuts are dominated by the Lagrangean cuts. This is due to the fact that the convex relaxation of the problem is weaker than the Lagrangean relaxation of the problem.

## 5.7 Conclusions

In this chapter, we have proposed a generalized-Benders decomposition-based branch and cut algorithm for two stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables. We include both Lagrangean cuts and Benders cuts in the Benders master problem. The convergence of the proposed algorithm relies on adding Lagrangean cuts to the Benders master problem and performing a branch-and-bound search on the first stage variables, which follows from the convergence proof of general branch and bound by Horst and Tuy (2013). The Benders cuts are derived from the convexified subproblems where cutting planes can be added. Our computational results show that adding cutting planes can potentially accelerate the convergence of the branch-and-bound procedure, especially in the stochastic pooling problem with contract selection. However, in the storage design for a multi-product plant under uncertainty problem, adding rank-one lift-project-cut does not help the branch-and-bound procedure.

For the problems with tight Lagrangean relaxation, the corresponding Benders cuts derived from the Benders subproblem convexified by cutting planes would be dominated by the Lagrangean cuts (storage design problem in subsection 5.6.3). On the other hand, for the problems with weak Lagrangean relaxation (stochastic pooling problem in subsection 5.6.1, and crude selection problem in subsection 5.6.2), adding Benders cuts may be able to close the duality gap significantly. Therefore, in order to predict whether it is preferable to add cutting planes, heuristics need to be proposed to decide cut generation.

In the spatial branch and bound process, a hybrid of the proposed branching rules is used. It would be desirable to have a large benchmark library of stochastic programming problem of the proposed category in order to test different hybrid branching rules.

While the proposed algorithm can solve the test cases to small optimality gaps, solving stochastic nonconvex MINLPs is still challenging. It would be desirable to come up with tighter convex relaxations for the nonconvex terms in stage 2 in order to generate tight Benders cuts. We have shown that in the crude selection problem using the hull relaxation of piecewise McCormick envelope, yields smaller optimality gaps than using the McCormick envelope. Tighter convex relaxations are in general more difficult to solve. Therefore, there is a tradeoff between using tight convex relaxations and doing more branching. We propose to obtain more computational studies in the future to learn the appropriate convex relaxations based on the features of each problem to make the proposed algorithm more efficient.

# Chapter 6

# Sample Average Approximation for Stochastic Nonconvex Mixed Integer Nonlinear Programming via Outer-Approximation

## 6.1 Problem Statement

The problem that we address in this chapter is defined in Eq. (6.1),

$$\text{(SP-MINLP)} \quad \min_{x \in X, y \in \{0,1\}^m} \mathbb{E}_{\theta \sim \mathbb{P}} \min_{z \in Z} f\left(x, y, z; \theta\right) \tag{6.1a}$$

$$\text{s.t.} \quad g(x, y, z; \theta) \leq 0 \quad \forall \theta \in \Theta \tag{6.1b}$$

where, $x$ and $y$ are vectors of continuous and discrete decision variables, respectively. Notice that any bounded general integer variable can be modeled using binary variables. $z$ is a vector of continuous recourse variables, which can vary depending on the realization of uncertain parameters, which are represented by the vector $\theta$. The uncertain parameter $\theta \in \Theta$ follows a continuous joint probability distribution $\mathbb{P}$, where $\Theta$ is the support of the uncertain parameter. Functions $f$ and $g$ are smooth functions, which can be nonconvex and

sets $X$ and $Z$ are compact. Problem (6.1) is a two-stage SP where variables $x$ and $y$ represent the first stage decisions, variable $z$ represents the second stage decisions. The expectation is taken over the probability distribution of uncertainty parameter $\theta$. We make the following assumption about problem (SP-MINLP).

**Assumption 8.** Problem (SP-MINLP) has relatively complete recourse, i.e., any solution $(x, y)$ that satisfies the first stage constraints has feasible recourse decisions in the second stage.

With this assumption, $\Theta$ is usually a bounded set in practice. Solving (SP-MINLP) with continuous probability distribution directly involves integrating over the distribution, which is usually computationally intractable. Instead of minimizing the 'true' expectation, one can generate $N$ i.i.d. samples for the uncertain parameter $\theta$ and minimize the empirical risk. The empirical risk minimization problem described in Eq. (6.2) is called sample average approximation (SAA) (Shapiro et al., 2009) in the SP literature.

$$\text{(SAA-MINLP)} \qquad \min_{x \in X, y \in \{0,1\}^m, z_i \in Z} \frac{1}{N} \sum_{i=1}^{N} f\left(x, y, z_i, \theta_i\right) \tag{6.2a}$$

$$\text{s.t.} \quad g(x, y, z_i, \theta_i) \leq 0 \quad \forall i \in [N] \tag{6.2b}$$

where $[N]$ represents the set $\{1, 2, \cdots, N\}$, from which we obtain $\theta_i$, $i \in [N]$, that are the $N$ i.i.d. samples of the uncertain parameter vector $\theta$, and $z_i$ is the stage 2 variable corresponding to each $\theta_i$. One option to solve (SP-MINLP) is to approximate it with (SAA-MINLP), which can be regarded as 'external sampling'. The convergence properties of (SAA-MINLP) have been widely studied (Shapiro et al., 2009; Ahmed et al., 2002). In this chapter, we take an internal sampling approach to solve (SP-MINLP) by using a nonconvex OA algorithm.

## 6.2 Sampling Average Approximation within the Outer-Approximation Algorithm (SAAOA)

We first give a high-level overview of the SAAOA algorithm before we go to the details. Let us take a step back and consider a nonconvex OA algorithm to solve deterministic MINLP. We have an MILP master problem where the nonconvex functions in the original MINLP are replaced by their polyhedral relaxations. The MILP master problem provides a lower bound (LB) to the original MINLP's optimal objective function value. After solving the MILP master problem, we fix the binary variables in the original MINLP to the optimal solution of the master problem and solve the resulting nonconvex NLP. If the nonconvex NLP provides a feasible solution, it is feasible to the original problem and yields an upper bound (UB) of the optimal objective function value.

Additionally, we include a 'no-good' cut that removes from the feasible set at iteration $k$ all the previously found binary variable combinations,

$$\sum_{j=1}^{m} a_j^{k'} y_j \leq b^{k'}, \text{where } a_j^{k'} = \begin{cases} 1 \text{ if } y_j^{k'} = 1 \\ -1 \text{ if } y_j^{k'} = 0 \end{cases} \quad \text{and } b^{k'} = \sum_{j=1}^{m} y_j^{k'} - 1, \forall k' < k \tag{6.3}$$

where arameter $y_i^{k'}$ is the optimal value of the $i$th binary variable at iteration $k'$.

Finally, we keep iterating between the MILP master problem and the NLP subproblem until the upper and lower bounds lie within certain tolerance.

The deterministic nonconvex OA algorithm can be extended to the solution of (SP-MINLP) by generating i.i.d. samples for both the master problem and the nonconvex subproblem using an internal sampling approach. We define all the subproblems in the SAAOA algorithm in the next subsection.

### 6.2.1 Subproblems definition

The (SP-OA-MILP) master problem at iteration $k$ is defined as

$$\text{(SP-OA-MILP)} \quad LB^k = \min_{x \in X, y \in \{0,1\}^m} \mathbb{E}_{\theta \sim P} \min_{z \in Z} \hat{f}(x, y, z; \theta) \tag{6.4a}$$

$$\text{s.t.} \quad \hat{g}(x, y, z; \theta) \leq 0 \quad \forall \theta \in \Theta; \quad \text{Eq. (6.2)} \tag{6.4b}$$

where $\hat{f}$ and $\hat{g}$ are polyhedral relaxations for function $f$, and $g$, respectively, which can be obtained by convexifying (SP-MINLP). $LB^k$ represents the LB of (SP-MINLP) after $k-1$ iterations, where $k-1$ 'no-good' cuts (6.2) have been added.

Problem (SP-OA-MILP) can be approximated by generating $N$ i.i.d. samples of $\theta_i$ from probability distribution $\mathbb{P}$ and solving the following (SAA-OA-MILP) master problem.

$$\text{(SAA-OA-MILP)} \quad \hat{w}_N^k = \min_{x \in X, z_i \in Z, y \in \{0,1\}^m} \frac{1}{N} \sum_{i=1}^{N} \hat{f}\left(x, y, z_i, \theta_i\right) \tag{6.5a}$$

$$\text{s.t.} \quad \hat{g}(x, y, z_i, \theta_i) \leq 0 \quad \forall i \in [N]; \quad \text{Eq. (6.2)} \tag{6.5b}$$

An UB to (SP-MINLP) can be found by fixing the binary variables at a value $\bar{y}$ and solving problem (SP-nonconvex-NLP),

$$\text{(SP-nonconvex-NLP)} \quad UB(\bar{y}) = \min_{x \in X} \mathbb{E}_{\theta \sim \mathbb{P}} \min_{z \in Z} f\left(x, \bar{y}, z, \theta\right) \tag{6.6a}$$

$$\text{s.t.} \quad g(x, \bar{y}, z, \theta) \leq 0 \quad \forall \theta \in \Theta \tag{6.6b}$$

The nonconvex NLP at fixed binary variable $\bar{y}$ by SAA is defined as,

$$\text{(SAA-nonconvex-NLP)} \quad \hat{u}_N(\bar{y}) = \min_{x \in X, z_i \in Z} \frac{1}{N} \sum_{i=1}^{N} f\left(x, \bar{y}, z_i, \theta_i\right) \tag{6.7a}$$

$$\text{s.t.} \quad g(x, \bar{y}, z_i, \theta_i) \leq 0 \quad \forall i \in [N] \tag{6.7b}$$

where $N$ i.i.d. samples of $\theta_i \in \Theta$ are generated from probability distribution $\mathbb{P}$.

### 6.2.2 Internal sampling using outer-approximation

The OA algorithm iterates between the (SAA-OA-MILP) master problem and the (SAA-nonconvex-NLP) subproblem. $N$ i.i.d. samples are generated for both the master and the subproblem at each iteration. A 'no-good' cut (6.2) is added to the (SAA-OA-MILP) master problem to eliminate the current integer solution. The steps of the SAAOA algorithm are described in Algorithm 1.

**Algorithm 1**

**Initialization:** Iteration counter $k = 1$; upper bound $UB = +\infty$; sample size $N$
**Step 1** Convexify the (SAA-nonconvex-MINLP) with samples of size $N$ and generate (SAA-OA-MILP)
**Step 2** Solve the (SAA-OA-MILP). Denote the optimal objective value as $\hat{w}_N^k$ and the vector of the values for the integer variables as $\bar{y}^k$. Set $LB = \hat{w}_N^k$
If $LB \geq UB - \epsilon$, then go to Step 5; otherwise, go to Step 3.
**Step 3** Fix the binary variables in the (SAA-nonconvex-NLP) to $\bar{y}^k$, i.e., set $\bar{y} = \bar{y}^k$
Solve the (SAA-nonconvex-NLP) to global optimality. Denote the objective value as $\hat{u}_N^k$ and the optimal solution as $\tilde{x}^k, \bar{y}^k$.
If $\hat{u}_N^k \leq UB - \epsilon$ , then let $UB = \hat{u}_N^k$, $x_p^* = \tilde{x}^k$, $y_p^* = \bar{y}^k$.
If $LB \geq UB - \epsilon$, then go to Step 5.
**Step 4** Let $k = k + 1$ and return to Step 2
**Step 5** Stop, the optimal solution is $x_p^*$, $y_p^*$, and the $\epsilon$-optimal objective value is $UB$.

## 6.3 Convergence results

### 6.3.1 Convergence of objective values and solutions

In this subsection, we prove that the optimal value and solutions obtained by the proposed Algorithm 1 converge to the optimal value and solutions of (SP-MINLP) with probability (w.p.) 1 as the sample size $N \to \infty$. Note that by optimal solutions, we mean the optimal first-stage decisions.

#### 6.3.1.1 Convergence of the upper bound.

In order to establish the results, we first prove that the convergence of the UB (SAA-nonconvex-NLP) to (SP-nonconvex-NLP). Note that the optimal objective values of (SAA-nonconvex-NLP) and (SP-nonconvex-NLP) when $y$ is fixed at $\bar{y}$ are defined as $UB(\bar{y})$ and $\hat{u}_N(\bar{y})$, respectively. We further define the set of optimal solutions of (SP-nonconvex-NLP) as $S^*(\bar{y})$; the set of optimal solutions of (SAA-nonconvex-NLP) as $\hat{S}_N(\bar{y})$. We want to prove that $\lim_{N\to\infty} \hat{u}_N(\bar{y}) = UB(\bar{y})$ and the event $\{S^*(\bar{y}) = \hat{S}_N(\bar{y})\}$ happens w.p. 1 as $N \to \infty$.

We define function $UB(\bar{y}, x)$ as

$$UB(\bar{y}, x) = \mathop{\mathbb{E}}_{\theta\sim\mathbb{P}} \min_{z\in Z} f(x, \bar{y}, z, \theta) \tag{6.8a}$$

$$\text{s.t.} \quad g(x, \bar{y}, z, \theta) \leq 0 \quad \forall \theta \in \Theta \tag{6.8b}$$

Function $\hat{u}_N(\bar{y}, x)$ is defined as,

$$\hat{u}_N(\bar{y}, x) = \frac{1}{N} \sum_{i=1}^{N} \min_{z_i \in Z} f\left(x, \bar{y}, z_i, \theta_i\right) \tag{6.9a}$$

$$\text{s.t.} \quad g(x, \bar{y}, z_i, \theta_i) \leq 0 \quad \forall i \in [N] \tag{6.9b}$$

Note that $\hat{u}_N(\bar{y}, x)$ contains the summation of $N$ i.i.d samples. According to the Law of Large Numbers, $\hat{u}_N(\bar{y}, x)$ converges pointwise w.p. 1 to $UB(\bar{y}, x)$ as $N \to \infty$. Moreover, under some mild conditions ((Shapiro, 1991; Shapiro et al., 2009; Shapiro, 1993; King and Rockafellar, 1993)), the convergence is uniform. To prove the convergence of the UB, we make the following assumptions, which are similar to the assumptions in Theorem 5.3 of Shapiro et al. (2009).

**Assumption 9.** All the samples $\theta_i$, $i \in [N]$, in all the SAA problems are i.i.d. from a distribution $\mathbb{P}$.

**Assumption 10.** Set $X$ is compact.

**Assumption 11.** The set $S^*(\bar{y})$ of optimal solutions of (SP-nonconvex-NLP) is nonempty, for any vector $\bar{y} \in \{0, 1\}^m$.

**Assumption 12.** The function $UB(\bar{y}, x)$ is finite valued and continuous in $x$ on $X$, for any vector $\bar{y} \in \{0, 1\}^m$.

**Remark.** Note that for a given uncertain parameter $\theta_i$, the cost-to-go function is discontinuous in general. However, $UB(\bar{y}, x)$ is the expected cost over a continuous distribution. Under some mild assumptions, $UB(\bar{y}, x)$ is continuous for all $x \in X$ (Schultz, 1995).

**Assumption 13.** The set $\hat{S}_N(\bar{y})$ is nonempty w.p. 1 for $N \to \infty$, for any vector $\bar{y} \in \{0, 1\}^m$.

**Assumption 14.** The function $\hat{u}_N(\bar{y}, x)$ converges to $UB(\bar{y}, x)$ w.p. 1, as $N \to \infty$, uniformly on $X$, for any vector $\bar{y} \in \{0, 1\}^m$.

Under assumptions 10-14, the following lemma holds.

**Lemma 5.** If Assumptions 10-14 hold true, $\lim_{N\to\infty} \hat{u}_N(\bar{y}) = UB(\bar{y})$ and the event $\{S^*(\bar{y}) = \hat{S}_N(\bar{y})\}$ happens w.p. 1 as $N \to \infty$, for any vector $\bar{y} \in \{0,1\}^m$ (Theorem 5.3 of Shapiro et al. (2009)).

*Proof.* According to Assumption 14,

$\max_{x\in X} |\hat{u}_N(\bar{y}, x) - UB(\bar{y}, x)| \to 0$, w.p. 1 as $N \to \infty$

We can bound $|\hat{u}_N(\bar{y}) - UB(\bar{y})|$ by,

$$\begin{aligned}
|\hat{u}_N(\bar{y}) - UB(\bar{y})| &= |\min_{x\in X} \hat{u}_N(\bar{y}, x) - \min_{x\in X} UB(\bar{y}, x)| \\
&= \max\Big\{ \min_{x\in X} \hat{u}_N(\bar{y}, x) - \min_{x\in X} UB(\bar{y}, x), \min_{x\in X} UB(\bar{y}, x) - \min_{x\in X} \hat{u}_N(\bar{y}, x)\Big\} \\
&\le \max\Big\{ \hat{u}_N(\bar{y}, x^*_{UB}(\bar{y})) - UB(\bar{y}, x^*_{UB}(\bar{y})), UB(\bar{y}, x^*_U(\bar{y})) - \hat{u}_N(\bar{y}, x^*_U(\bar{y}))\Big\} \\
&\le \max_{x\in X} |\hat{u}_N(\bar{y}, x) - UB(\bar{y}, x)|
\end{aligned}$$

where $x^*_{UB}(\bar{y}) = \text{argmin}_{x\in X} UB(\bar{y}, x)$, $x^*_U(\bar{y}) = \text{argmin}_{x\in X} \hat{u}_N(\bar{y}, x)$. Therefore, we have $\lim_{N\to\infty} \hat{u}_N(\bar{y}) = UB(\bar{y})$.

Now we need to prove that the event $\{S^*(\bar{y}) = \hat{S}_N(\bar{y})\}$ happens w.p. 1 as $N \to \infty$, for any vector $\bar{y} \in \{0,1\}^m$. We prove this by contradiction. Due to the compactness of $X$, we can assume that there exists $\hat{x}_N(\bar{y}) \in \hat{S}_N(\bar{y})$ such that $\text{dist}\big(\hat{x}_N(\bar{y}), S^*(\bar{y})\big) \ge \epsilon$, for some $\epsilon > 0$, and that $\hat{x}_N(\bar{y})$ tends to a point $x^*(\bar{y}) \in X$. Since $x^*(\bar{y}) \notin S^*(\bar{y})$, we have $UB(\bar{y}, x^*(\bar{y})) > UB(\bar{y})$.

$$\begin{aligned}
\hat{u}_N\big(\bar{y}, \hat{x}_N(\bar{y})\big) - UB\big(\bar{y}, x^*(\bar{y})\big) =& \Big[\hat{u}_N\big(\bar{y}, \hat{x}_N(\bar{y})\big) - UB(\bar{y}, \hat{x}_N(\bar{y}))\Big] \\
&+ \Big[UB\big(\bar{y}, \hat{x}_N(\bar{y})\big) - UB\big(\bar{y}, x^*(\bar{y})\big)\Big]
\end{aligned}$$

The first term on the right hand side goes to zero by the uniform convergence of $\hat{u}_N(\bar{y}, x)$ (Assumption 14). The second term goes to zero by the continuity of function $UB(\bar{y}, x)$ (Assumption 12). Then we have $\lim_{N\to\infty} \hat{u}_N\big(\bar{y}, \hat{x}_N(\bar{y})\big) = UB\big(\bar{y}, x^*(\bar{y})\big) > UB(\bar{y})$, which contradicts with the convergence of the objective value $\hat{u}_N(\bar{y})$ to $UB(\bar{y})$. □

#### 6.3.1.2 Convergence of the optimal values and solutions of the lower bound.

Next, we prove the convergence of the LB estimators. Similar to the analysis of the UB, we define function, $LB^k(x, y)$,

$$LB^k(x, y) = \mathop{\mathbb{E}}_{\theta\sim\mathbb{P}} \min_{z\in Z} \hat{f}\left(x, y, z; \theta\right) \tag{6.10a}$$

$$\text{s.t.} \quad \hat{g}(x, y, z; \theta) \leq 0 \quad \forall \theta \in \Theta; \quad \text{Eq. (6.2)} \tag{6.10b}$$

Function $\hat{w}_N^k(x, y)$ is defined as,

$$\hat{w}_N^k(x, y) = \frac{1}{N} \sum_{i=1}^{N} \min_{z_i\in Z} \hat{f}\left(x, y, z_i, \theta_i\right) \tag{6.11a}$$

$$\text{s.t.} \quad \hat{g}(x, y, z_i, \theta_i) \leq 0 \quad \forall i \in [N]; \quad \text{Eq. (6.2)} \tag{6.11b}$$

The optimal values of (SP-OA-MILP) and (SAA-OA-MILP) are defined as $LB^k$, $\hat{w}_N^k$, respectively. The sets of optimal solutions for (SP-OA-MILP) and (SAA-OA-MILP) at iteration $k$ are denoted as $T_k^*$, $\hat{T}_k^N$, respectively. We want to prove that $\lim_{N\to\infty} \hat{w}_N^k = LB^k$ and the event $\{T_k^* = \hat{T}_k^N\}$ happens w.p. 1 as $N \to \infty$. The complication compared with the previous subsection is the presence of binary variables in (SP-OA-MILP). Note that in the proof for the convergence of (SP-nonconvex-NLP) the binary variables are fixed. However, if we consider some fixed $\bar{y}$ that is feasible for (SP-OA-MILP) at iteration $k$, by making similar assumptions to Assumptions 10-12, we can prove the convergence of the optimal values and solutions of the LB estimators in (SAA-OA-MILP) to (SP-OA-MILP) with the $y$ variables fixed at $\bar{y}$. Let the optimal value of (SP-OA-MILP) with $y$ fixed at $\bar{y}$ be $LB^k(\bar{y})$. The optimal value of (SP-OA-MILP) can be obtained by taking the minimum over all the possible $LB^k(\bar{y})$. Since the combinations of binary variables are finite, the convergence of both the optimal values and optimal solutions to (SP-OA-MILP) can be established. We state the following lemma whose proof is similar to Lemma 5, and hence we omit it.

**Lemma 6.** Under mild conditions similar to Assumptions 10-14, the optimal value and the optimal solutions of (SAA-OA-MILP) converges to those of (SP-OA-MILP) w.p. 1 as $N \to \infty$, i.e., $\lim_{N\to\infty} \hat{w}_N^k = LB^k$, the event $\{T_k^* = \hat{T}_k^N\}$ happens w.p. 1 as $N \to \infty$.

#### 6.3.1.3 Convergence of optimal objective values and solutions of Algorithm 1

The convergence of the optimal values and solutions of the LB and the UB is proved setting up the main theorem of this section.

**Theorem 4.** The proposed Algorithm 1 returns the optimal value and the set of optimal solutions $(x^*, y^*)$ of (SP-MINLP) w.p. 1 as $N \to \infty$.

*Proof.* Note that Algorithm 1 can only miss finding the optimal solution $y^*$ in 1) the comparison of the optimal objective value of (SAA-nonconvex-NLP), $\hat{u}_N^k$, and the current upper bound $UB$, and 2) the comparison of the objective value of the (SAA-OA-MILP) with the current UB. We need to prove that neither of these two cases can happen as $N \to \infty$. Note that Lemmas 5 and 6 prove that all the estimators are 'exact' in the sense they converge to the optimal value of the corresponding true stochastic programming problems (SP-nonconvex-NLP) and (SP-OA-MILP). Therefore, the optimal solution cannot be missed in either of the two cases. Since there are only finite combinations of binary variables, it takes Algorithm 1 finite number of iterations to converge. The optimal value returned is the optimal value of the true (SP-MINLP) and the optimal solution corresponds to the $y^*$ that yields the best UB. □

### 6.3.2 Estimates of sample sizes

Having proved that Algorithm 1 converges in the limit, it is desirable to estimate the sample size to achieve finite error $\epsilon$ with high probability.

Ahmed et al. (2002) give the sample size estimator for two-stage SP. An SAA estimator, which is solved to $\delta$-optimality, gives the $\epsilon$-optimal solution to the corresponding true problem with probability at least $1 - \alpha$ if the sample size

$$N \geq \frac{12\sigma^2}{(\epsilon - \delta)^2}\Big(n_1 \ln \frac{2DL}{\epsilon - \delta} - \ln \alpha\Big) \tag{6.12}$$

where $D$ is the diameter of set $X$, i.e., $D = \sup_{x \in X, x' \in X} ||x - x'||$, the objective function of the SP problem is assumed to be $L$-Lipschitz continuous on $X$, $n_1$ is the dimension of the

first stage variables, and $\sigma^2$ is the maximal variance of certain differences between values of the objective function of the SAA problem (see Kleywegt et al. (2002)).

**Remark.** Eq. (6.12) gives the rate of convergence for any SAA estimators of any two-stage SP. Theoretically, it can be used to calculate the sample sizes of the MILP master problem and the nonconvex NLP subproblem, respectively, if we have the upper bounds of the diameter of the feasible set $X$ and the Lipschitz constant $L$. However, for stochastic MINLP problems, the Lipschitz constant $L$ is difficult to estimate. Moreover, the sample size estimates from (6.12) is usually too conservative in practice (Kleywegt et al., 2002). Therefore, we need to design an algorithm based on Algorithm 1 that is more efficient in practice. Note that the convergence results and the sample size estimates only apply to Algorithm 1. The algorithm that will be discussed in the next section is only a 'practical' algorithm to solve the (SP-MINLP) problem.

## 6.4 Algorithm Design

Here we extend Algorithm 1 to a more practical algorithm based on the ideas of Kleywegt et al. (2002) where an empirical method is proposed to construct confidence intervals for the optimal objective value of the 'true' SP problem. Throughout this section, we assume that all the problems are solved to global optimality.

For a finite sample size $N$, Mak et al. (1999) prove that the expectation of the SAA problems provides lower bounds for the corresponding true SP problems. In Algorithm 1, the expectation of the objective of (SAA-OA-MILP), $\hat{w}_N^k$, provides a lower bound for the objective of (SP-OA-MILP), $LB^k$, at every iteration $k$,

$$\mathbb{E}_{\theta_i \sim \mathbb{P}} \left[ \hat{w}_N^k \right] \leq LB^k$$

For the objective of (SP-nonconvex-NLP), $UB(\bar{y})$, and the objective of the corresponding (SAA-nonconvex-NLP), $\hat{u}_N(\bar{y})$, we have,

$$\mathbb{E}_{\theta_i \sim \mathbb{P}} \left[ \hat{u}_N(\bar{y}) \right] \leq UB(\bar{y})$$

Therefore, the SAA estimators, $\hat{w}_N^k$ and $\hat{u}_N(\bar{y})$, can be regarded as the lower estimators of (SP-OA-MILP) and (SP-nonconvex-NLP), respectively. In order to construct the confidence intervals for (SP-OA-MILP) and (SP-nonconvex-NLP), we need to provide their upper estimators.

### 6.4.1 Upper estimators

Mak et al. (1999) prove that the upper estimator of a SP can be obtained by evaluating the sample mean at a given feasible first stage decision. In our case, the upper estimator of the (SP-OA-MILP) can be obtained by solving the following problem for random sample $\theta_i$, $i = 1, ..., N'_L$

$$\text{(single-OA-LP)} \quad \hat{w}_1^{(i),k} = \min_{z_i \in Z} \hat{f}\left(\tilde{x}, \tilde{y}, z_i, \theta_i\right) \tag{6.13a}$$

$$\text{s.t.} \quad \hat{g}(\tilde{x}, \tilde{y}, z_i, \theta_i) \leq 0 \tag{6.13b}$$

where the first stage decisions $(x, y)$ are fixed at $(\tilde{x}, \tilde{y})$. The values of $(\tilde{x}, \tilde{y})$ can come from a good estimate of the optimal solution of (SP-OA-MILP), for example, the optimal solution of (SAA-OA-MILP). The upper estimator of the (SP-OA-MILP) can be the average of the $N'_L$ (single-OA-LP) problems,

$$\bar{w}_{N'_L}^k = \frac{1}{N'_L} \sum_{i=1}^{N'_L} \hat{w}_1^{(i),k}$$

Similarly, the upper estimator of (SP-nonconvex-NLP) can be derived by fixing the first stage decisions $(x, y)$ to $(\bar{x}, \bar{y})$ and solve the rest of the nonconvex NLP for $N'_U$ samples of $\theta_i$. The values for $(\bar{x}, \bar{y})$ can come from the optimal solution of (SAA-nonconvex-NLP) $(\bar{x}^k, \bar{y}^k)$. The $i$th single size nonconvex NLP at iteration $k$ is defined as,

$$\text{(single-nonconvex-NLP)} \quad \hat{u}_1^{(i),k} = \min_{z_i \in Z} f\left(\bar{x}, \bar{y}, z_i, \theta_i\right) \tag{6.14a}$$

$$\text{s.t.} \quad g(\bar{x}, \bar{y}, z_i, \theta_i) \leq 0 \tag{6.14b}$$

The upper estimator of the (SP-nonconvex-NLP) can be the average of the $N'_U$ (single-

nonconvex-NLP) problems,

$$\bar{u}^k_{N'_U} = \frac{1}{N'_U}\sum_{i=1}^{N'_U} \hat{u}_1^{(i),k}$$

Note that the major difference between (6.13) and (6.14) is that in (6.13) polyhedral relaxations of functions $f$ and $g$ are used.

### 6.4.2 Confidence intervals for the upper and lower bound

We show how the confidence intervals of the upper and the lower bounds at every iteration $k$ of the SAAOA Algorithm 2 can be constructed.

We first show how the confidence interval of (SP-nonconvex-NLP) can be constructed at each iteration. Recall that the expectation of (SAA-nonconvex-NLP) provides a LB of (SP-nonconvex-NLP), i.e., $\mathbb{E}[\hat{u}_N(\bar{y})] \leq UB(\bar{y})$. To have a good estimate for the expectation, we solve $M_U$ batches of (SAA-nonconvex-NLP). We make the following assumption about the batches of the SAA problems,

**Assumption 15.** All the $M_U$ ($M_L$) batches of samples in the (SAA-nonconvex-NLP) and (SAA-OA-MILP) problems are independent and identically distributed.

At each iteration $k$, (SAA-nonconvex-NLP) with $N_U$ samples is solved $M_U$ times to obtain a lower estimator of the upper bound $UB(\bar{y}^k)$. We use random variable $\hat{u}^{(m),k}_{N_U}$ to denote the optimal objective value of the $m$th batch of (SAA-nonconvex-NLP) at iteration $k$. From Mak et al. (1999), random variable $\hat{u}^{(m),k}_{N_U}$ is a biased estimator of $UB(\bar{y}^k)$, i.e.,

$$\mathbb{E}_{\theta_i\sim\mathbb{P}}\left[\hat{u}^{(m),k}_{N_U}\right] \leq UB(\bar{y}^k) \tag{6.15}$$

The mean of the $M_U$ batches (SAA-nonconvex-NLP) is defined as $\bar{u}^k_{N_U,M_U} = \frac{1}{M_U}\sum_{m=1}^{M_U} \hat{u}^{(m),k}_{N_U}$. By the central limit theorem,

$$\sqrt{M_U}\Big(\bar{u}^k_{N_U,M_U} - \mathbb{E}_{\theta_i\sim\mathbb{P}}\left[\bar{u}^k_{N_U,M_U}\right]\Big) \Rightarrow \mathcal{N}(0,\sigma^2_{u,k}) \tag{6.16}$$

where '$\Rightarrow$' denotes convergence in distribution, and $\mathcal{N}(0,\sigma^2)$ denotes a normal distribution with mean zero and variance $\sigma^2$. $(\hat{S}^{u,k}_{M_U})^2$ is the standard sample variance estimator of $\sigma^2_{u,k}$,

which is defined by

$$\frac{(\hat{S}^{u,k}_{M_U})^2}{M_U} = \frac{1}{M_U(M_U-1)}\sum_{m=1}^{M_U}(\hat{u}^{(m),k}_{N_U} - \bar{u}^k_{N_U,M_U})^2 \tag{6.17}$$

Therefore, the $(1-\alpha)$ confidence interval of the lower estimator of the UB can be approximated by,

$$\left(\bar{u}^k_{N_U,M_U} - t^{\alpha/2}_{M_U-1}\frac{\hat{S}^{u,k}_{M_U}}{\sqrt{M_U}}, \quad \bar{u}^k_{N_U,M_U} + t^{\alpha/2}_{M_U-1}\frac{\hat{S}^{u,k}_{M_U}}{\sqrt{M_U}}\right) \tag{6.18}$$

where $t^{\alpha/2}_{M_U-1}$ is the $1-\alpha/2$ quantile of $t$-distribution with $M_U-1$ degrees of freedom.

At each iteration $k$, $N'_U$ (single-nonconvex-NLP)s are solved to obtain the upper estimator of (SP-nonconvex-NLP). We use $\hat{u}^{(i),k}_1$ to denote the optimal objective value of the $i$th sample of (single-nonconvex-NLP) at iteration $k$. By definition, we also have the mean of the $N'_U$ samples, $\bar{u}^k_{N'_U} = \frac{1}{N'_U}\sum_{i=1}^{N'_U}\hat{u}^{(i),k}_1$. From Mak et al. (1999), we have

$$\mathbb{E}_{\theta_i\sim\mathbb{P}}\left[\bar{u}^k_{N'_U}\right] \geq UB(\bar{y}^k) \tag{6.19}$$

In Eqs. (6.16) and (6.17), by the central limit theorem,

$$\sqrt{N'_U}\left(\bar{u}^k_{N'_U} - \mathbb{E}_{\theta_i\sim\mathbb{P}}\left[\bar{u}^k_{N'_U}\right]\right) \Rightarrow N(0,\sigma^2_{u',k}) \tag{6.20}$$

The standard sample variance estimator $(\hat{S}^{u,k}_{N'_U})^2$ of $\sigma^2_{u',k}$ is defined by,

$$\frac{(\hat{S}^{u,k}_{N'_U})^2}{N'_U} = \frac{1}{N'_U(N'_U-1)}\sum_{i=1}^{N'_U}\left(\hat{u}^{(i),k}_1 - \bar{u}^k_{N'_U}\right)^2 \tag{6.21}$$

Therefore, the $(1-\alpha)$ confidence interval of the upper estimator of the UB can be approximated by,

$$\left(\bar{u}^k_{N'_U} - t^{\alpha/2}_{N'_U-1}\frac{\hat{S}^{u,k}_{N'_U}}{\sqrt{N'_U}}, \quad \bar{u}^k_{N'_U} + t^{\alpha/2}_{N'_U-1}\frac{\hat{S}^{u,k}_{N'_U}}{\sqrt{N'_U}}\right) \tag{6.22}$$

where $t^{\alpha/2}_{N'_U-1}$ is the $1-\alpha/2$ quantile of the $t$-distribution with $N'_U-1$ degrees of freedom. By combining (6.22) and (6.18), we have with probability at least $(1-\alpha)$, the UB obtained at iteration $k$, $UB(\bar{y}^k)$, lies within the interval,

$$(\underline{UB}, \overline{UB}) = \left(\bar{u}^k_{N_U,M_U} - t^{\alpha/2}_{M_U-1}\frac{\hat{S}^{u,k}_{M_U}}{\sqrt{M_U}}, \quad \bar{u}^k_{N'_U} + t^{\alpha/2}_{N'_U-1}\frac{\hat{S}^{u,k}_{N'_U}}{\sqrt{N'_U}}\right) \tag{6.23}$$

where $(\hat{S}^{u,k}_{N'_U})^2$ is the standard sample variance estimator of $\sigma^2_{u',k}$, the variance of the scaled normal distribution to what $\bar{u}^k_{N'_U}$ minus its expected value converges in distribution.

Similarly, to construct the confidence interval for (SP-OA-MILP) at each iteration $k$, we can solve the (SAA-OA-MILP) with $N_L$ i.i.d. samples for $M_L$ i.i.d. batches. The optimal objective of the $m$th batch is denoted as $\hat{w}_{N_L}^{(m),k}$. We compute the mean and variance of the $M_L$ batches by,

$$\bar{w}_{N_L,M_L}^k = \frac{1}{M_L}\sum_{m=1}^{M_L} \hat{w}_{N_L}^{(m),k} \text{ and } \frac{(\hat{S}_{M_L}^{w,k})^2}{M_L} = \frac{1}{M_L(M_L-1)}\sum_{m=1}^{M_L}(\hat{w}_{N_L}^{(m),k} - \bar{w}_{N_L,M_L}^k)^2$$

Similar to Eq. (6.18), the $(1-\alpha)$ confidence interval of the lower estimator of the LB is approximately,

$$\left(\bar{w}_{N_L,M_L}^k - t_{M_L-1}^{\alpha/2}\frac{\hat{S}_{M_L}^{w,k}}{\sqrt{M_L}}, \quad \bar{w}_{N_L,M_L}^k + t_{M_L-1}^{\alpha/2}\frac{\hat{S}_{M_L}^{w,k}}{\sqrt{M_L}}\right) \tag{6.24}$$

For the upper estimator of (SP-OA-MILP), we can solve (single-OA-LP) $N_L'$ times with $N_L'$ i.i.d. samples of $\theta_i$. The optimal objective value of the $i$th (single-OA-LP) is denoted as $\hat{w}_1^{(i),k}$. We can compute the mean and variance of the objective values with,

$$\bar{w}_{N_L'}^k = \frac{1}{N_L'}\sum_{i=1}^{N_L'} \hat{w}_1^{(i),k} \text{ and } \frac{(\hat{S}_{N_L'}^{w,k})^2}{N_L'} = \frac{1}{N_L'(N_L'-1)}\sum_{i=1}^{N_L'}(\hat{w}_1^{(i),k} - \bar{w}_{N_L'}^k)^2$$

As in Eq. (6.22), the $(1-\alpha)$ confidence interval of the for the upper estimator of the LB can be approximated by,

$$\left(\bar{w}_{N_L'}^k - t_{N_L'-1}^{\alpha/2}\frac{\hat{S}_{N_L'}^{w,k}}{\sqrt{N_L'}}, \quad \bar{w}_{N_L'}^k + t_{N_L'-1}^{\alpha/2}\frac{\hat{S}_{N_L'}^{w,k}}{\sqrt{N_L'}}\right) \tag{6.25}$$

With Eqs. (6.24) and (6.25), the $(1-\alpha)$ confidence interval of the LB can be approximated by,

$$(\underline{LB}, \overline{LB}) = \left(\bar{w}_{N_L,M_L}^k - t_{M_L-1}^{\alpha/2}\frac{\hat{S}_{N_L,M_L}^{w,k}}{\sqrt{M_L}}, \quad \bar{w}_{N_L'}^k + t_{N_L'-1}^{\alpha/2}\frac{\hat{S}_{N_L'}^{w,k}}{\sqrt{N_L'}}\right) \tag{6.26}$$

### 6.4.3 SAAOA with confidence intervals

With the confidence interval results from Eqs. (6.23) and (6.26), we can approximate the values of the upper and lower bounds of the 'true' SP at each iteration of the OA algorithm with high probability. A high-level overview of the SAAOA with confidence intervals is shown in Figure 6.1. At the initialization step, (SAA-MINLP) problems are convexified

to generate (SAA-OA-MILP) and (single-OA-LP) problems. At each iteration, $M_L$ (SAA-OA-MILP) problems each with size $N_L$ are solved. Then we fix the first stage binary and continuous variables and solve $N'_L$ (single-OA-LP) to construct the upper estimator of the LB. After that, we only fix the first stage binary variables $y$ and solve $M_U$ batches of (SAA-nonconvex-NLP) each with sample size $N_U$ to construct the lower estimator of the UB. Then $N_U$ (single-nonconvex-NLP) are solved to construct the upper estimator of the UB. At the end of each iteration, we check if the algorithm converges (if $\underline{LB} \geq \overline{UB} - \epsilon$). If not, sample sizes might need to be updated and the algorithm keeps iterating. The steps of the SAAOA method with confidence interval estimators are described in Algorithm 2.

It is difficult to estimate the desired sample sizes for the estimators *a priori*. Therefore, we may need to update the sample sizes, $N_U$, $M_U$, $N'_U$, $N_L$, $M_L$, $N'_L$, if the confidence intervals are not tight enough. The update policies for these parameters are not unique. We discuss the update policies in the next subsection.

### 6.4.4 Update policies

As we have discussed, estimating the proper sample sizes before we solve any of the SAA problems is challenging. Small sample sizes may yield large confidence intervals and provide poor estimates for the upper and lower bounds. On the other hand, large sample sizes become too conservative and increase the computational time. Therefore, choosing the update policies on $N_U$, $M_U$, $N'_U$, $N_L$, $M_L$, $N'_L$, is crucial to the performance of the algorithm.

For the upper estimators, $N'_U$ or $N'_L$ single sample size problems are solved. The numbers $N'_U$, $N'_L$ affect the accuracy of the estimates of the feasible solutions $(\tilde{x}, \tilde{y})$. The sample sizes $N'_U$, $N'_L$ should increase if the confidence intervals for the upper estimators in Eqs. (6.22) and (6.25) are too large.

For the lower estimators, we have two types of parameters to tune, i.e., the batch sizes $M_L$ and $M_U$, and the sample sizes of the batches $N_L$, $N_U$. Increasing the value of the batch sizes $M_L$ and $M_U$ make the estimators for the expected value more accurate, i.e., the length of the confidence intervals for the lower estimators in Eqs. (6.18) and (6.24) will decrease. The

**Algorithm 2**

**Initialization:** Iteration counter $k = 1$; upper bound $UB = +\infty$.
Generate $M_L$ i.i.d. batches of samples with size $N_L$, one batch of samples with size $N'_L$, $M_U$ i.i.d. batches of samples with size $N_U$ and one batch of samples with size $N'_U$ of $\theta \sim \mathbb{P}$
**Step 1**
**for** $m = 1$ to $M_L$ **do**
Convexify the (SAA-nonconvex-MINLP) with the $m$th batch of samples of size $N_L$ and generate (SAA-OA-MILP).
**end for**
**Step 2**
**for** $i = 1$ to $N'_L$ **do**
Convexify (SAA-MINLP) with the $i$th single sample. Generate a problem to be solved in Step 5 as (single-OA-LP).
**end for**
**Step 3**
**for** $m = 1$ to $M_L$ **do**
Solve the (SAA-OA-MILP) with the $m$th batch of samples. Denote the objective value as $\hat{w}^{(m),k}_{N_L}$ and the optimal binary variable value as $\bar{y}^{(m),k}$
**end for**
**Step 4** Compute the mean and variance of the solutions:

$$\bar{w}^k_{N_L,M_L} = \frac{1}{M_L}\sum_{m=1}^{M_L} \hat{w}^{(m),k}_{N_L} \text{ and } \frac{(\hat{S}^{w,k}_{M_L})^2}{M_L} = \frac{1}{M_L(M_L-1)}\sum_{m=1}^{M_L}(\hat{w}^{(m),k}_{N_L} - \bar{w}^k_{N_L,M_L})^2$$

Let $\bar{y}^k$ be the most common integer solution among all the $\bar{y}^{(m),k}$, $m \in [M_L]$.
**Step 5** For all the following $N'_L$ problems, fix $x := \tilde{x}$ ($\tilde{x}$ can be any feasible solution or a solution of any batch of (SAA-OA-MILP) at iteration $k$ where the optimal solution $\bar{y}^k$ is found).
**for** $i = 1$ to $N'_L$ **do**
Solve (single-OA-LP), with a single sample $\theta_i$. Denote the objective value as $\hat{w}^{(i),k}_1$
**end for**
Compute the mean and variance of the objective values:

$$\bar{w}^k_{N'_L} = \frac{1}{N'_L}\sum_{i=1}^{N'_L} \hat{w}^{(i),k}_1 \text{ and } \frac{(\hat{S}^{w,k}_{N'_L})^2}{N'_L} = \frac{1}{N'_L(N'_L-1)}\sum_{i=1}^{N'_L}(\hat{w}^{(i),k}_1 - \bar{w}^k_{N'_L})^2$$

**Step 6** Compute the $(1-\alpha)$ confidence interval of the lower bound

$$(\underline{LB}, \overline{LB}) = \left(\bar{w}^k_{N_L,M_L} - t^{\alpha/2}_{M_L-1}\frac{\hat{S}^{w,k}_{N_L,M_L}}{\sqrt{M_L}},\quad \bar{w}^k_{N'_L} + t^{\alpha/2}_{N'_L-1}\frac{\hat{S}^{w,k}_{N'_L}}{\sqrt{N'_L}}\right)$$

Let $LB = \bar{w}^k_{N'}$,$\underline{LB} = \bar{w}^k_{N_L,M_L} - t^{\alpha/2}_{M_L-1}\frac{\hat{S}^{w,k}_{N_L,M_L}}{\sqrt{M_L}}$, $\overline{LB} = \bar{w}^k_{N'_L} + t^{\alpha/2}_{N'_L-1}\frac{\hat{S}^{w,k}_{N'_L}}{\sqrt{N'_L}}$.
If $\underline{LB} \geq \overline{UB} - \epsilon$, then go to Step 11; otherwise go to Step 7.
**Step 7** Fix $y := \bar{y}^k$.
**for** $m = 1$ to $M_U$ **do**
Solve (SAA-nonconvex-NLP) to global optimality. Denote objective value as $\hat{u}^{(m),k}_{N_U}$
**end for**
Compute the mean and variance of the lower estimator of the upper bound,

$$\bar{u}^k_{N_U,M_U} = \frac{1}{M_U}\sum_{m=1}^{M_U} \hat{u}^{(m),k}_{N_U},\quad \frac{(\hat{S}^{u,k}_{M_U})^2}{M_U} = \frac{1}{M_U(M_U-1)}\sum_{m=1}^{M_U}(\hat{u}^{(m),k}_{N_U} - \bar{u}^k_{N_U,M_U})^2$$

**Step 8** For all the following $N'_U$ problems, fix $x := \tilde{x}^k$, given by heuristics, e.g., take the mean value of the optimal solutions in (SAA-nonconvex-NLP).
**for** $i = 1$ to $N'_U$ **do**
Solve the single-nonconvex-NLP. Denote the objective value as $\hat{u}^{(i),k}_1$
**end for**
Compute the mean and variance of the upper estimator of the upper bound,

$$\bar{u}^k_{N'_U} = \frac{1}{N'}\sum_{i=1}^{N'_U} \hat{u}^{(i),k}_1,\quad \frac{(\hat{S}^{u,k}_{N'_U})^2}{N'_U} = \frac{1}{N'_U(N'_U-1)}\sum_{i=1}^{N'_U}\left(\hat{u}^{(i),k}_1 - \bar{u}^k_{N'_U}\right)^2$$

**Step 9** Compute the confidence interval of the optimality gap for the upper bound

$$(\underline{UB}, \overline{UB}) = \left(\bar{u}^k_{N_U,M_U} - t^{\alpha/2}_{M_U-1}\frac{\hat{S}^{u,k}_{M_U}}{\sqrt{M_U}},\quad \bar{u}^k_{N'_U} + t^{\alpha/2}_{N'_U-1}\frac{\hat{S}^{u,k}_{N'_U}}{\sqrt{N'_U}}\right)$$

Update the sample size with policy P1 or P2 described in subsection 5.4. Go to Step 1, Step 2, Step 7, Step 8, or Step 10 depending on the chosen policy. Otherwise if $\bar{u}^k_{N'_U} < UB - \epsilon$ , then let $UB = \bar{u}^k_{N'_U}$, $\underline{UB} = \bar{u}^k_{N_U,M_U} - t^{\alpha/2}_{M_U-1}\frac{\hat{S}^{u,k}_{M_U}}{\sqrt{M_U}}$, $\overline{UB} = \bar{u}^k_{N'_U} + t^{\alpha/2}_{N'_U-1}\frac{\hat{S}^{u,k}_{N'_U}}{\sqrt{N'_U}}$, $x^*_p = \tilde{x}^k$.
Add 'no-good' cut corresponding to $\bar{y}^k$ to (SAA-OA-MILP).
If $\underline{LB} \geq \overline{UB} - \epsilon$, then go to Step 11; otherwise go to Step 10.
**Step 10** Let $k = k + 1$ and return to Step 2
**Step 11** Stop, the optimal solution is $y^*_p$, $x^*_p$ and the optimal objective value is $UB$.

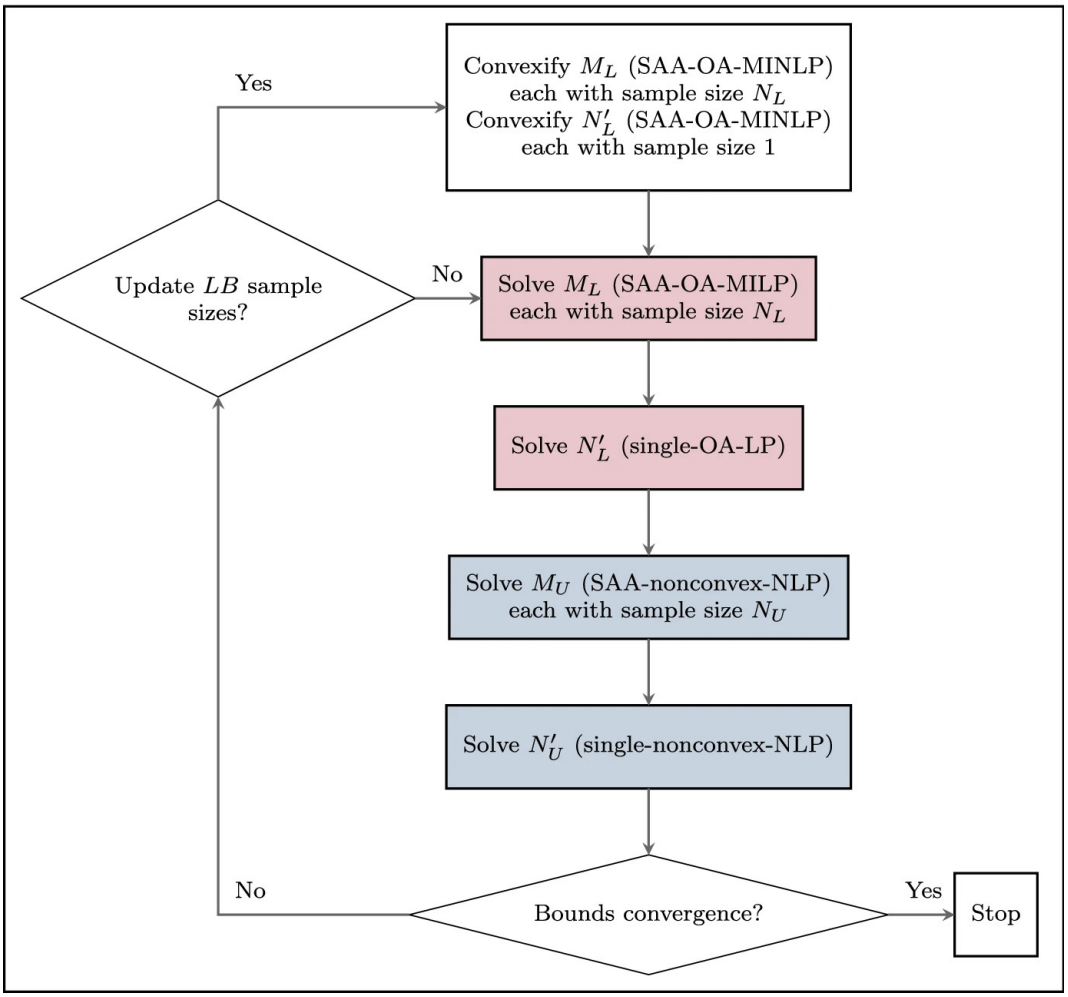


Figure 6.1: Flowchart of the SAAOA algorithm with confidence interval estimators. Boxes filled in red and blue show the MILP phase of and the NLP phase of the algorithm, respectively

impact of increasing the value of the sample sizes $N_L$, $N_U$ is two-fold. First, the expectation of the lower estimators becomes tighter with the increase in the sample size, which is proved by Mak et al. (1999).

$$\mathbb{E}_{\theta_i \sim \mathbb{P}}\left[\hat{u}_{N_U}^{(m),k}\right] \leq \mathbb{E}_{\theta_i \sim \mathbb{P}}\left[\hat{u}_{N_U+1}^{(m),k}\right] \leq UB(\bar{y}^k), \quad \forall N_U \tag{6.27}$$

Therefore, increasing the sample size makes the lower estimator tighter and Algorithm 2 may converge in fewer iterations. Second, the variance of the lower estimator decreases with the increase in the number of samples. With the same number of batches, tighter confidence intervals can be obtained with a larger sample size.

We describe two update policies for choosing the sample and batch sizes.

*Policy 1 (P1): Increase if loose confidence interval policy* At each iteration, check if the confidence interval is tight enough at Step 9 of the algorithm. A confidence interval is considered tight if the relative gap between its upper and lower estimators is less than a certain threshold, e.g., 5%. If it is not tight, multiply the number of samples and batches by a fixed ratio. For example, if $t^{\alpha/2}_{M_L-1}\frac{\hat{S}^{w,k}_{N_L,M_L}}{\sqrt{M_L}}$ is large, increase $N_L$ by setting $N_L^{\text{new}} = \beta N_L^{\text{old}}$, increase $M_L$ by setting $M_L^{\text{new}} = \gamma M_L^{\text{old}}$, where $\beta$ and $\gamma$ are parameters greater than 1. Four cases can happen when the confidence intervals are checked at Step 9.

- The confidence intervals for both the lower bound and the upper bound are tight at iteration $k$. Go to Step 10.

- The confidence interval for the lower bound is tight. The confidence interval for the upper bound is not tight. Increase sample size $N_U$, $M_U$ if $t^{\alpha/2}_{M_U-1}\frac{\hat{S}^{u,k}_{M_U}}{\sqrt{M_U}}$ is large. Increase sample size $N'_U$ if $t^{\alpha/2}_{N'_U-1}\frac{\hat{S}^{u,k}_{N'_U}}{\sqrt{N'_U}}$ is large. If both cases are true or only the former case is true, go back to Step 7. Otherwise, go back to Step 8.

- The confidence interval for the upper bound is tight. The confidence interval for the lower bound is not tight. Increase sample size $M_L$, $N_L$ if $t^{\alpha/2}_{M_L-1}\frac{\hat{S}^{w,k}_{N_L,M_L}}{\sqrt{M_L}}$ is large. Increase sample size $N'_L$ if $t^{\alpha/2}_{N'_L-1}\frac{\hat{S}^{w,k}_{N'_L}}{\sqrt{N'_L}}$ is large. If both cases are true or only the former case is true, go back to Step 1. Otherwise, go back to Step 2.

- Both of the confidence intervals are loose. Update the sample sizes as in the two former cases. Go back to Step 1 or Step 2.

Note that the 'no-good' cut is only added at Step 9 when the confidence interval for the UB is tight. Upper limits for $N_U$, $M_U$, $N'_U$, $N_L$, $M_L$, $N'_L$ are set to avoid intractability, which could sacrifice the tightness of the confidence intervals.

*Policy 2 (P2): Increase until overlap policy* Similar to P1, P2 also checks the tightness of the confidence intervals at Step 9 of the algorithm. However, P2 does not increase the initial sample sizes until the confidence interval of the LB and the confidence interval of

the UB overlap. After the overlap occurs, increase the sample sizes of (SAA-OA-MILP) and/or (single-OA-LP) at the current iteration if the confidence interval of the LB is not tight and increase the sample sizes of (SAA-nonconvex-NLP) and/or (single-nonconvex-NLP) corresponding to the best UB found so far if the confidence interval for the UB is not tight. The increase strategy relies on multiplying fixed ratios similar to P1. Keep increasing the sample sizes until one of the three cases occur: 1) the confidence intervals of the lower and the upper bound become tight; 2) the confidence intervals no longer overlap but the LB is still lower than the current best UB; 3) the confidence intervals no longer overlap but the LB is greater than the best UB. In case 1), check if the $\overline{UB}$ is less than the $\underline{LB}$, if not, keep iterating. Otherwise, terminate. In case 2), keep iterating until the bounds overlap again. In case 3), terminate. The 'no-good' cut is added in a given iteration if the confidence intervals do not overlap or the UB confidence interval is tight. Note that if one keeps iterating, then the same strategy is used recursively until one of the termination criteria is satisfied. Since the confidence intervals of the UB are not necessarily tight when the algorithm terminates, reevaluating all the integer solutions $\bar{y}^k$s by solving (SAA-nonconvex-NLP) and (single-nonconvex-NLP) problems may be needed to find the best feasible solution.

We make the following remark to end this section.

**Remark.** The algorithm can be applied to two-stage SP with both continuous and binary first stage variables in the first stage. However, it works better for problems with pure binary variables in the first stage. In this case, we do not have to solve (SAA-nonconvex-NLP), which can be a large scale nonconvex NLP, because fixing the first stage binary variables leads to fully decomposable (single-nonconvex-NLP) subproblems. Therefore, we only need to solve (single-nonconvex-NLP), potentially in parallel to obtain the confidence interval for the upper bound, which is given by equation (6.22).

## 6.5 Computational Results

Algorithm 2 with the two proposed update policies is implemented in Pyomo / Python (Hart et al., 2017) as a package named `saaoa.py`, which takes a two-stage model in the data structure of PySP Watson et al. (2012).

The convex relaxation for (SAA-MINLP) is obtained using a special version of BARON v.19.4.4 (Kılınç and Sahinidis, 2018) which provides the root node polyhedral relaxation. (SAA-OA-MILP) and (single-OA-LP) are solved using CPLEX v.12.9. The (SAA-nonconvex-NLP) (single-nonconvex-NLP) problems are solved with BARON v.19.3.24. All the problems in this chapter are solved using one processor of an Intel Xeon (2.67GHz) machine with 64 GB RAM.

### 6.5.1 Illustrative example: a process network problem

To show how Algorithm 2 works, we present a small process network problem involving concave investment cost functions. Consider producing a chemical C which can be manufactured with either process 2 or process 3, both of which use chemical B as raw material. B can be purchased from another company and/or manufactured with process 1 which uses A as a raw material. The demand for chemical C, denoted as $d$, is the source of uncertainty. The superstructure of the process network is shown in Figure 6.2.

The problem is formulated as a two-stage stochastic program. The first stage decisions are investment decisions on the three processes, which include binary variables $Y_i$ to denote whether process $i$ is selected and continuous variables $CAP_i$ to denote the capacity of process $i$ for $i = 1, 2, 3$. The second stage decisions are the material flows. We denote the uncertainty scenarios as $\omega$ and explicitly state the dependency of the second stage decision on them by presenting them as functions of $\omega$. Variables $PA(\omega)$, $PB(\omega)$ represent the purchase amount of chemical A and B, respectively. Other material flows for chemical B and C are shown in

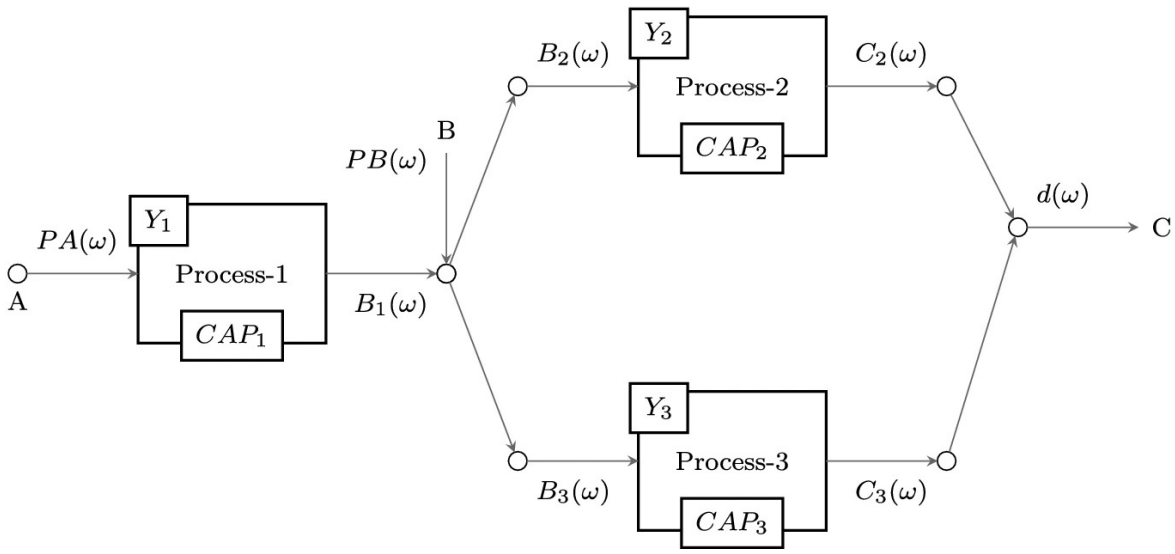


Figure 6.2: Superstructure for the process network problem.

the superstructure in Figure 6.2. The MINLP formulation is shown as follows.

$$\begin{aligned}\min \quad & 10Y_1 + 15Y_2 + 20Y_3 + 25CAP_1^{0.6} + 8CAP_2^{0.6} + 10CAP_3^{0.6} \\ & + \mathop{\mathbb{E}}_{d(\omega)\sim\mathbb{P}} 5PA(\omega) + 20PB(\omega) - 30(C_2(\omega) + C_3(\omega))\end{aligned} \tag{6.28a}$$

$$\text{s.t.} \quad CAP_1 \leq U \cdot Y_1, \quad CAP_2 \leq U \cdot Y_2, \quad CAP_3 \leq U \cdot Y_3, \quad Y_2 + Y_3 \leq 1 \tag{6.28b}$$

$$PA(\omega) \leq CAP_1, \quad B_2(\omega) \leq CAP_2, \quad B_3(\omega) \leq CAP_3 \quad \forall\omega \tag{6.28c}$$

$$B_1(\omega) = \frac{25}{1 + e^{-PA(\omega)+13}}, \quad C_2(\omega) = 0.82B_2(\omega), \quad C_3(\omega) = 0.95B_3(\omega) \quad \forall\omega \tag{6.28d}$$

$$B_1(\omega) + PB(\omega) = B_2(\omega) + B_3(\omega) \quad \forall\omega \tag{6.28e}$$

$$C_2(\omega) + C_3(\omega) \leq d(\omega) \quad \forall\omega \tag{6.28f}$$

$$Y_i \in \{0, 1\} \quad \forall i \in \{1, 2, 3\} \tag{6.28g}$$

In stage 1, (6.28b) forces the capacity of any given process to zero if that process is not selected and only one of process 2 and 3 can be selected. The stage 2 equations are defined for each scenario $\omega$. Equation (6.28c) limits the input to each process within the installed capacity. Equation (6.28d) describes the input-output relationship of each process. Note that the output of process 1, $B_1(\omega)$, is approximated by a sigmoidal function of the input, which is a nonlinear nonconvex function. Equation (6.28e) describes a material balance for chemical B. The production of chemical C is less than the demand $d(\omega)$. The objective is to maximize the expected profit. To be consistent with the notation of this chapter, we denote the objective as minimizing the negative of the expected profit. Note that the variable investment cost is a concave function of the capacity of each process. The demand is assumed to follow a truncated Gaussian distribution $\mathcal{N}(10, 0.5)$ where the values less than 9 or greater 11 are truncated. To relate this problem with the problem statement in equation (6.1), $x = CAP_i$, $y = \{Y_i\}$, $\forall i \in \{1, 2, 3\}$. $z = \{PA, PB, B_1, B_2, B_3, C_2, C_3\}, \theta = d$

Algorithm 2 is applied to solve this small problem. We fix the sample sizes: $N_L = 200$, $M_L = 10$, $N'_L = 100$, $N_U = 50$, $M_U = 10$, $N'_U = 100$. First, 10 batches of (SAA-MINLP) each with sample size 200 are convexified to obtain 10 batches of (SAA-OA-MILP) problems. Each (SAA-MINLP) has 3 binary variables, 1,403 continuous variables, and 1,604 constraints. 100 batches of (SAA-MINLP) each with sample size 1 are convexified to be solved as (single-OA-LP) problems.

At iteration $k = 1$, 10 batches of (SAA-OA-MILP) each with sample size 200 are solved. We examine the optimal solutions of the 10 batches and found that in most cases processes 1 and 3 are installed. The binary variables are fixed at $Y_1 = 1$, $Y_2 = 0$, $Y_3 = 1$, and the capacities of processes 1 and 3 are fixed at the optimal capacities of one batch of (SAA-OA-MILP) where processes 1 and 3 are selected. The 95% confidence interval for the lower bound is $(-115.33, -110.36)$ calculated using equation (6.26). We proceed to the NLP phase of the algorithm with the binary variables fixed at $Y_1 = 1$, $Y_2 = 0$, $Y_3 = 1$. 10 batches of (SAA-nonconvex-NLP) each with sample size 50 are solved to obtain the lower estimator of the lower bound. 100 (single-nonconvex-NLP) problems are solved with the capacity of each

process fixed at the mean of the optimal solutions of the 10 batches of (SAA-nonconvex-NLP). The 95% confidence interval for the upper bound is $(-50.59, -44.82)$ calculated by equation (6.23). Since the confidence intervals for the upper and the lower bound do not satisfy the termination criteria, $\underline{LB} \geq \overline{UB} - \epsilon$, we add an integer cut corresponding to the solution $Y_1 = 1$, $Y_2 = 0$, $Y_3 = 1$ to each batch of the (SAA-OA-MILP) problems and proceed to iteration 2.

At iteration $k = 2$, the confidence intervals for the lower bound becomes $(-102.1, -97.19)$ after the integer cut is added. The optimal binary variables found by (SAA-OA-MILP) problems are $Y_1 = 1$, $Y_2 = 1$, $Y_3 = 0$. The confidence interval for the upper bound corresponding to this binary variable combination is $(-58.07, -52.32)$. Since the new confidence interval is strictly lower than that at iteration 1, the best feasible solution is updated to $Y_1 = 1$, $Y_2 = 1$, $Y_3 = 0$. The termination criteria is still not satisfied. An integer cut corresponding to the solution $Y_1 = 1$, $Y_2 = 1$, $Y_3 = 0$ is added to each batch of the (SAA-OA-MILP) problems.

At iteration 3, the confidence interval for the lower bound is $(-41.31, -40.42)$. Since the lower bound is strictly greater than the upper bound, i.e., $\underline{LB} \geq \overline{UB} - \epsilon$, the algorithm terminates. We conclude that the optimal solution is $Y_1 = 1$, $Y_2 = 1$, $Y_3 = 0$ with a confidence interval of $(-58.07, -52.32)$.

### 6.5.2 Stochastic Pooling problem

The pooling problem arises in several engineering applications, such as petrochemical and agricultural industries. This nonconvex network flow optimization problem considers different flows that can be mixed and need to satisfy constraints concerning their quality attributes, which depend on their composition (Tawarmalani and Sahinidis, 2002). The stochastic pooling problem has been studied by Li et al. (2011b, 2012b). The first stage decisions are investment decisions on sources, pools, and pipelines, which are represented as binary variables. The second stage decisions are the mass flows and the split fractions, which are represented as continuous variables that give rise to nonconvex bilinear terms. The constraints include mass balance, investment capacity, and quality specifications. The

objective is to minimize the expected cost. This problem is an example of an engineering design problem posed as a stochastic mixed-integer nonlinear problem.

#### 6.5.2.1 Small pooling instance

We first consider a small pooling instance with 4 sources, 1 pool, 2 terminals, and 2 quality attributes. This instance has 16 binary variables and 26 constraints in the first stage; 21 continuous variables and 56 constraints per scenario in the second stage. The uncertainty is assumed to arise from the quality of one source whose deviation from the nominal value follows a truncated distribution $\mathcal{N}(0, \sigma)$ where the parameter values less than $-2\sigma$ or greater $2\sigma$ are truncated.

We apply the SAAOA Algorithm 2 to solve the problem for $\sigma = 0.0031$ and $\sigma = 0.004$ using both update policies P1 and P2. Since the problem only has pure binary first stage variables, we do not need to solve (SAA-nonconvex-NLP). The sample sizes and batch sizes that need to be updated are $N_L$, $M_L$, $N'_L$, $N'_U$. We multiply the sizes by fixed ratios, i.e., $N_L^{\text{new}} = \beta N_L^{\text{old}}$, $M_L^{\text{new}} = \gamma M_L^{\text{old}}$, $N'^{\text{new}}_L = \beta N'^{\text{old}}_L$, $N'^{\text{new}}_U = \beta N'^{\text{old}}_U$, where $\beta$ and $\gamma$ are constants greater than 1. In Algorithm 2, we start with $N_L = 50$, $M_L = 10$, $N'_L = 50$, $N'_U = 50$ and update the sample sizes using P1 and P2 respectively. We calculate the 95% confidence intervals for both the upper and lower bound, i.e., $\alpha = 5\%$. A confidence interval is considered tight if the relative gap between its upper and lower estimators is less than 5%.

To compare with the SAAOA algorithm, we use BARON v.19.3.24 to solve 10 batches of (SAA-MINLP) problems each with a sample size of 50. Each of these MINLP problems has 1,066 variables and 2,826 constraints. The time limit for each (SAA-MINLP) problem is set to 10,000 seconds. The expectation of (SAA-MINLP) provides a lower bound for the original (SP-MINLP). The lower estimator of (SP-MINLP) can be obtained similar to Eq. (6.24). Once we fix the binary variables from the optimal solution of (SAA-MINLP), we solve 100 individual (single-nonconvex-NLP) to estimate the expected value of the optimal solution. The best solution for the 10 batches is reported.

The computational results for $\sigma = 0.0031$ and $\sigma = 0.004$ are shown in Table 6.1. In

Table 6.1: Computational results for the small pooling instance

| Variance | $\sigma = 0.0031$ | | | | | $\sigma = 0.004$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Policy | P1 and P2 | | | | DE | P1 | | | | P2 | | | | DE |
| $\beta$ | 2 | 2 | 1.5 | 1.5 | - | 2 | 2 | 1.5 | 1.5 | 2 | 2 | 1.5 | 1.5 | - |
| $\gamma$ | 2 | 1.5 | 2 | 1.5 | - | 2 | 1.5 | 2 | 1.5 | 2 | 1.5 | 2 | 1.5 | - |
| Iterations | 9 | 9 | 9 | **9** | - | 22 | 24 | 20 | 20 | 21 | 22 | **10** | 12 | - |
| Wall time (s) | 784 | 675 | 584 | **529** | 100,412 | 19,845 | 16,376 | 16,708 | 11,094 | 13,098 | 14,364 | **7,177** | 10,190 | 100,471 |
| (SAA-OA-MILP) | 110 | 101 | 97 | 89 | - | 3,273 | 1,576 | 2,309 | 751 | 2,288 | 1,319 | 566 | 754 | - |
| (single-OA-LP) | 1 | 1 | 1 | 1 | - | 45 | 45 | 25 | 29 | 22 | 43 | 1 | 1 | - |
| (single-nonconvex-NLP) | 167 | 172 | 175 | 181 | - | 10,044 | 10,094 | 5,839 | 5,738 | 6,109 | 8,685 | 206 | 299 | - |
| Convexification | 496 | 390 | 300 | 248 | - | 5,946 | 4,086 | 8,098 | 4,219 | 4,207 | 3,772 | 6,384 | 9,108 | - |
| $\overline{UB}$ | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 |
| $\underline{UB}$ | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 | -115.0 |
| $\overline{LB}$ | -105.0 | -115.0 | -105.0 | -105.0 | - | -107.2 | -105.5 | -109.6 | -111.4 | -108.7 | -105.0 | -103.5 | -109.5 | - |
| $\underline{LB}$ | -113.5 | -109.8 | -110.9 | -109.1 | -172.4 | -114.5 | -114.8 | -114.6 | -114.5 | -114.2 | -113.8 | -114.8 | -113.9 | -171.3 |
| $N_L$ | 100 | 100 | **75** | **75** | 50 | 200 | 200 | **168** | **168** | 200 | 200 | **168** | 252 | 50 |
| $M_L$ | 20 | **15** | 20 | **15** | 10 | 40 | **22** | 80 | 33 | 40 | **22** | 80 | 49 | 10 |
| $N'_L$ | 100 | 100 | **75** | **75** | - | 3,200 | 2,000 | 2,868 | 2,868 | 1,600 | 3,200 | **75** | 168 | - |
| $N'_U$ | 50 | 50 | 50 | 50 | 100 | 2,000 | 3,200 | 2,000 | 2,000 | 2,000 | 2,000 | **50** | 75 | 100 |

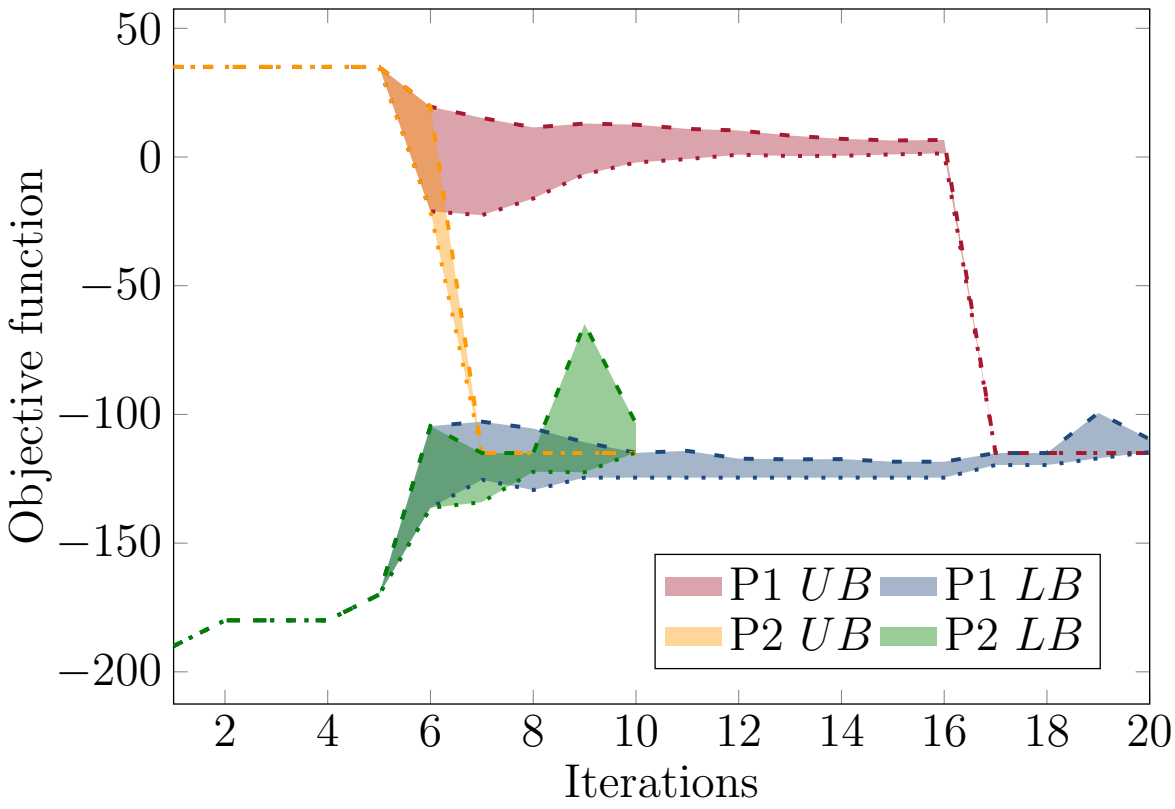


Figure 6.3: Bounds convergence for test case with $\sigma = 0.004$, $\beta = 1.5$, $\gamma = 2$

the case of $\sigma = 0.0031$, P1 and P2 give the same results. We use different constants $\beta$ and $\gamma$ for updating the sample sizes. The upper and lower estimators when the SAAOA algorithm terminates are shown in the table. Note that we allow the algorithm to terminate after the confidence interval of the LB is strictly greater than the confidence interval of the UB, i.e., $\underline{LB} \geq \overline{UB} - \epsilon$. In all the four cases, the SAAOA algorithm returns the same optimal solution, which gives an optimal value of -115.0. In both policies, the parameters $M_L$, $N_L$, $N'_L$ are updated once at the last iteration to tighten the LB. After the update, the algorithm terminates. Therefore, the smallest update ratio, $\beta = 1.5$, $\gamma = 1.5$ gives the least computational time. The column 'DE' (deterministic equivalent) represents using BARON to solve (SAA-MINLP) directly. Here, we slightly abuse notation to report the UB estimator and the LB estimator returned by external sampling. It is easy to see that the SAAOA algorithm outperforms external sampling given that it avoids solving the large scale nonconvex MINLP problem directly.

Compared with the low variance case, in the case with $\sigma = 0.004$ the algorithm needs more iterations to converge especially for P1. Recall that in P1, the sample sizes are updated whenever the confidence intervals are not tight. Therefore, if the confidence interval of the UB in a given iteration is not tight enough, we need to run another iteration with an increased sample size without adding a 'no-good' cut to cut off the current binary solution. In P2, the algorithm keeps adding one 'no-good' cut to the master problem at each iteration until the confidence intervals of the upper and the lower bound overlap. In general, P2 takes less computational time and fewer iterations to converge than P1.

To show how P1 and P2 perform differently, we show the convergence of the confidence intervals of $\sigma = 0.004$, $\beta = 1.5$, $\gamma = 2$ in Figure 6.3. In the beginning, the confidence intervals for both the upper and lower bounds are tight. When the confidence intervals become loose, P1 increases the sample sizes immediately to get tighter confidence intervals. As a result, P1 spends several iterations just to tighten the confidence intervals while the gap between bounds does not reduce significantly. On the other hand, P2 keeps adding 'no-good' cuts until the upper and the lower confidence intervals overlap. Therefore, P2 converges in a fewer number of iterations and saves computational time.

We also show a breakdown of the computational time of SAAOA Algorithm 2 in solving (SAA-OA-MILP), (single-OA-LP), (single-nonconvex-NLP), and convexification. In most cases, the majority of computational time is spent on solving (single-nonconvex-NLP) and generating convex relaxations.

#### 6.5.2.2 Large pooling instance

A large pooling instance with 5 sources, 2 pools, 4 terminals, and 7 quality attributes is solved. This instance has 24 binary variables and 39 constraints in the first stage; 89 continuous variables and 184 constraints per scenario in the second stage. The uncertainty is assumed to come from three different sources, the qualities of two feed streams and the upper bound of one terminal demand. The deviations of the feed qualities from the nominal value follow a truncated $\mathcal{N}(0, 0.02)$ distribution; the deviation from the upper bound of the

demand follows a $\mathcal{N}(0, 3)$ distribution, where the parameter values that are more than two standard deviations away from the mean values are truncated.

Table 6.2: Computational results for the large pooling instance

| Policy | P1 | | P2 | |
|---|---|---|---|---|
| $\beta$ | 1.5 | 2 | 1.5 | 2 |
| Iterations | 32 | **28** | 32 | **28** |
| Wall time (s) | 48,462 | 48,230 | 45,721 | **45,133** |
| (SAA-OA-MILP) | 1,950 | 1,731 | 1,963 | 1,750 |
| (single-OA-LP) | 9 | 7 | 7 | 8 |
| (single-nonconvex-NLP) | 45,958 | 45,959 | 43,223 | 42,837 |
| Convexification | 327 | 307 | 307 | 317 |
| $\overline{UB}$ | -400.5 | -400.2 | -399.8 | -400.2 |
| $\underline{UB}$ | -384.7 | -385.8 | -383.8 | -385.8 |
| $\overline{LB}$ | -383.8 | -385.5 | -383.8 | -385.5 |
| $\underline{LB}$ | -373.7 | -373.9 | -370.6 | -373.9 |
| $N_L$ | 50 | 50 | 50 | 50 |
| $M_L$ | 10 | 10 | 10 | 10 |
| $N'_L$ | 112 | 200 | **75** | 100 |
| $N'_U$ | 168 | **100** | 168 | 200 |

We apply Algorithm 2 to solve this problem using the same setting (initial sample sizes, probability of confidence intervals, relative gap, solvers, etc.) as in the small pooling instance. The computational results are shown in Table 6.2. Different values of the fixed ratio parameters $\beta$ and $\gamma$ are tested. In this case, the computational results do not depend on the value of $\gamma$, i.e., the value $M_L$ is never updated in all the cases. Therefore, only the $\beta$ values are reported in Table 6.2. Compared with the small pooling instance, the computational time increases significantly due to the larger NLP subproblems. However, Algorithm 2 is

still able to solve the problem within one day of computational time. P2 still takes less wall time than P2 but the difference is not significant. Most of the computational time is spent on (single-nonconvex-NLP) problems. The convexification time is small because the sizes of $N_L$ have not increased.

**Remark.** In general, P2 outperforms P1 in terms of the number of iterations and computational time. In the test cases, they yield the same optimal solution. P1 is more conservative in adding 'no-good' cuts to the master problem, i.e., a 'no-good' cut can only be added if the confidence interval corresponding to the integer solution is tight. Computational time can be wasted in constructing a tight estimate for suboptimal solutions. However, if one prefers to have a tight estimate for each solution so that there is more certainty that one solution is suboptimal, P1 is preferable.

## 6.6 Conclusions

In this chapter, we propose a sample average approximation based outer-approximation (SAAOA) algorithm for solving two-stage nonconvex stochastic MINLPs. The SAAOA algorithm iterates between an MILP master problem (SAA-OA-MILP) and nonconvex NLP subproblems (SAA-nonconvex-NLP). We prove that the SAAOA algorithm converges as the sample size tends to infinity and provides a theoretical estimate for the sample size. Since the sample size estimates are too conservative in practice, we design an SAAOA algorithm with confidence interval estimates for the upper bound and the lower bound at each iteration of the OA algorithm. To construct the confidence intervals, we define (single-OA-LP) and (single-nonconvex-NLP), which are proved to provide upper estimators for the problem's lower bound and upper bound, respectively. The sample sizes are updated dynamically using some update polices. We propose two update policies, namely, *P1: Increase if loose confidence interval policy* and *P2: Increase until overlap policy*. The algorithm is suitable for solving two-stage stochastic MINLPs with pure binary variables where the nonconvex NLP subproblems can be solved for each scenario separately. Computational results are shown

for a stochastic pooling problem. The SAAOA algorithm with confidence interval estimates is shown to perform better than solving the deterministic equivalent (SAA-MINLP) directly in terms of computational time and optimality gap. We provide some criteria for selecting between update Policy 1 and Policy 2 in Remark 6.5.2.2.

Future work can be focused on improving the update policies. To find an update policy that works well in general, we need to have more test cases to benchmark different update policies and tune the parameters like the fixed ratio in the two update policies proposed in this chapter. Although the algorithm can be used to address stochastic nonconvex MINLPs with mixed-binary first stage variables in theory, it is only computationally efficient for stochastic nonconvex MINLPs with pure binary first stage variables. Therefore, another possible future direction would be making the algorithm more efficient for problems with mixed-binary first stage variables.

# Chapter 7

# Shale Gas Pad Development Planning under Price Uncertainty

## 7.1 Problem Statement for the Deterministic Model

The deterministic problem addressed in this article is similar to the one proposed by Ondeck et al. (2019). We consider the shale gas development problem for a single wellpad represented in Figure 7.1. The potential wells are identified *a priori*. The productivity profile for each well is also given. In order to complete a shale gas well, four operations, i.e., top setting, horizontal drilling, fracturing, turning in line, need to be done sequentially. The corresponding equipment needs to be disassembled once the operation on the wellpad is changed and there is a mobilization cost for the change of operation.

We consider gas curtailment in the model, i.e., the the amount of gas produced can be less than the amount determined by the production curve. The curtailed gas is stored in the well and can be released in future time periods. Instead of modeling the storage of curtailed gas in each well, we simplify the problem by considering virtual storage of the whole wellpad. It should be noted that the storage considered in the model is virtual storage, i.e., the curtailed gas is stored under the wellpad itself. We are not considering "real" storage like storage tanks.

The decisions in the model are: 1) when and which wells should be developed to completion, 2) the timing of the operations performed on the well, 3) when and how much gas should be released from the completed wells. 4) the amount of gas coming in and out of storage at each time.

The major assumptions in this work are:

1. The locations and the information related to the development and the production of the prospective wells are known *a priori*. This includes the lengths of the wells as well as the production curves, which are functions of the wells' horizontal drilling lengths. The wells are optimally placed laterally such that well interference does not occur, i.e., the production curve of any given well is not affected by its neighboring wells.

2. Well shut-in is not allowed, i.e., once a well is turned in line, it cannot be shut-in.

3. The development cost for all four operations and the time to complete each operation are known for each well.

4. Every operation is performed for the entire well. A well cannot be "partially" completed. Note that this requirement is removed in the stochastic programming models because in some of the scenarios in the stochastic model, it is no longer profitable to finish all the operations.

5. Every operation can be performed at most once at a well. There are no refracturing of wells at a later time.

6. The development resource mobilization cost is a one time fee that includes transportation, assembly, and disassembly. The value is an assumed estimate, calculated based on past experience.

7. The optimization is solved using a discrete time model with time intervals of one week.

Since the mathematical formulation of the deterministic model is similar to Ondeck et al. (2019), we include the description of the MILP model in Appendix A. The focus of this

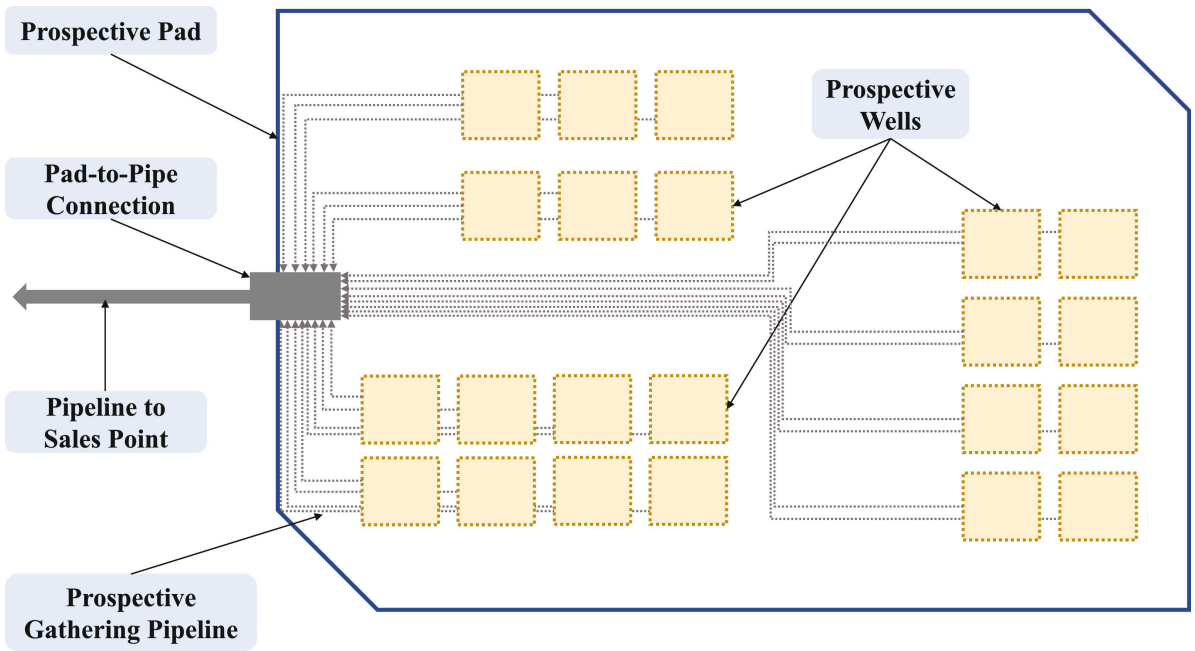


Figure 7.1: Prospective pad

chapter is on the extensions of the deterministic MILP model, which consider uncertainty in natural gas price.

## 7.2 Motivation for a Stochastic Programming Model

Although the deterministic model for the single pad shale gas development problem has been studied by Ondeck et al. (2019), there is no model for single pad planning that considers uncertainty in the parameters to the planning model.

Natural gas price is a major source of uncertainty in the planning problem. Henry Hub natural gas spot prices from 1998-2018 (U.S. Energy Information Administration, 2019b) are shown in Figure 7.2. It is easy to observe that the natural gas price has fluctuated significantly in the past 20 years. Natural gas price has reached as high as over 15 dollars per Btu in 2006. Hydraulic fracturing has currently reduced the Henry Hub spot price of natural gas to about 3 dollars per Btu. On the one hand, upstream operators would like to develop more wells when the price is high. On the other hand, there are also scenarios where

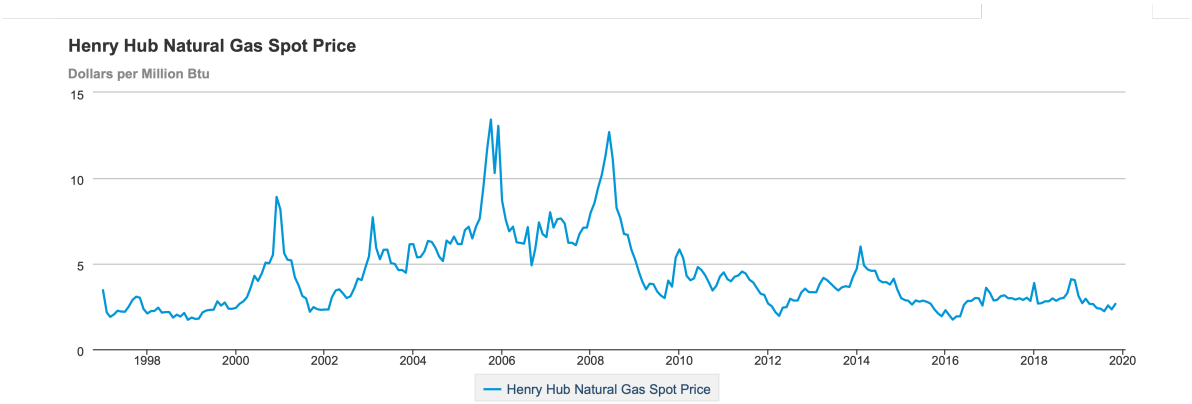


Figure 7.2: Historical natural gas price

natural gas is no longer a profitable business.

Therefore, in this chapter, we investigate the effect of the uncertainties from natural gas price on the optimal decision-making in the pad development problem. Since the decisions involved in the pad development are long-term decisions, usually ranging from a few months to years, we use stochastic programming to maximize the expected net present value. A brief introduction to stochastic programming is given in the next section.

## 7.3 Introduction to Stochastic Programming

Stochastic programming is an optimization framework that deals with decision making under uncertainty (Birge and Louveaux, 2011). In stochastic programming, it is assumed that the probability distributions of the uncertain parameters are known *a priori*. The uncertainties are usually characterized by some discrete realizations of the uncertain parameters as an approximation to the continuous probability distribution. For example, the realizations of the demand of a product can have three different values which represent high, medium, and low demand, respectively. Each realization is defined as a *scenario*. The objective of stochastic programming is to optimize the expected value of an objective function (e.g., the expected cost) over all the scenarios.

### 7.3.1 Two-stage stochastic programming

A special case of stochastic programming is two-stage stochastic programming (Figure 7.3). Specifically, stage 1 decisions are made 'here and now' at the beginning of the period, and are then followed by the resolution of uncertainty. Stage 2 'wait and see' decisions, or recourse decisions, are taken as corrective actions at the end of the period. One common type of a two-stage stochastic program is the mixed-integer linear program presented in equation (7.1). Set $\Omega$ is the set of scenarios. Parameter $\tau(\omega)$ is the probability of scenario $\omega$. The $n$-dimensional vector $x$ represents the first stage decisions, while the $m$-dimensional vector $y(\omega)$ represents the second stage decisions in scenario $\omega$. Both $x$ and $y(\omega)$ variables can be mixed-integer. Without loss of generality, we assume that the first $n_1$ ($n_1 \leq n$) variables of the first stage decisions and the first $m_1$ ($m_1 \leq m$) variables of the second stage decisions are binary. The uncertainties are reflected in the matrices (vectors), $W(\omega)$, $T(\omega)$, $h(\omega)$. Equation (7.1) is often referred to as the *deterministic equivalent* of the two-stage stochastic program. This problem can be solved directly if the number of scenarios is modest; if the number of scenarios is large, special decomposition algorithms, such as Benders decomposition (Benders, 1962), can be used. However, if we have (mixed) integer stage two variables, Benders decomposition is not applicable.

$$\min \quad c^T x + \sum_{\omega \in \Omega} \tau(\omega) d^T(\omega) y(\omega) \tag{7.1a}$$

$$\text{s.t.} \quad Ax \leq b \tag{7.1b}$$

$$W(\omega)y(\omega) \leq h(\omega) - T(\omega)x \quad \forall \omega \in \Omega \tag{7.1c}$$

$$x \in \left\{x = (x_1, x_2) : x_1 \in \{0,1\}^{n_1}, x_2 \geq 0\right\} \tag{7.1d}$$

$$y(\omega) \in \left\{y = (y_1, y_2) : y_1 \in \{0,1\}^{m_1}, y_2 \geq 0\right\} \tag{7.1e}$$

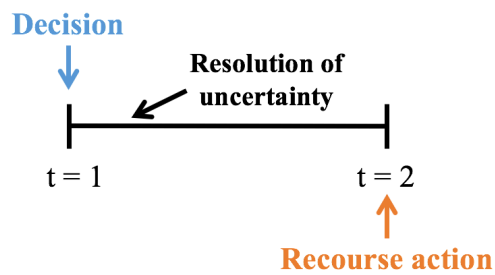


Figure 7.3: Two-stage problem

### 7.3.2 The value of stochastic solution

The value of stochastic solution (VSS) (Birge and Louveaux, 2011) is used to quantify the value that stochastic programming yields compared with the deterministic model. We need to define some notation before we present the mathematical expression for VSS. Let $\xi(\omega)$ be the vector that represents the random parameters involved in the two-stage stochastic programming problem for scenario $\omega$. Define $Q\left(x, \xi(\omega)\right)$

$$Q\left(x, \xi(\omega)\right) = c^T x + \min_y d^T(\omega) y \tag{7.2a}$$

$$\text{s.t.} \quad W(\omega)y \leq h(\omega) - T(\omega)x, \quad y \in \left\{y = (y_1, y_2) : y_1 \in \{0,1\}^{m_1} \quad y_2 \geq 0\right\} \tag{7.2b}$$

as the optimization problem associated with one particular realization of random parameter $\xi$. The expected value of $\xi$ is defined as $\bar{\xi}$. The *expected value solution* is defined as

$$\bar{x}(\bar{\xi}) = \arg\min_x Q\left(x, \bar{\xi}\right) \tag{7.3}$$

where the parameter $\xi$ is fixed at its expected value $\bar{\xi}$. In order to quantify how the expected value solution performs in different scenarios, we define *the expected results of using the expected value solution* (EEV) as

$$EEV = \mathbb{E}_\xi\left[Q\left(\bar{x}(\bar{\xi}), \xi\right)\right] \tag{7.4}$$

The *recourse problem* (RP), i.e., the two-stage stochastic program, is defined as

$$RP = \min_x \mathbb{E}_\xi Q\left(x, \xi\right) \tag{7.5}$$

which chooses the first stage decisions $x$ that minimizes the expected value of $Q\left(x, \xi\right)$. The difference of EEV and RP can quantify the difference of the expected cost between the

expected value solution and the stochastic solution. Therefore, it is reasonable to define $EEV - RP$ as the *value of stochastic solution* (VSS),

$$VSS = EEV - RP \tag{7.6}$$

VSS quantifies the difference between the expected outcome of the optimal stochastic solution and optimal expected value solution. Therefore, VSS is a metric that can evaluate the additional value that stochastic programming can create compared with the deterministic model. In practice, it is only worth solving the stochastic programming model if VSS is significant for the decision-maker.

### 7.3.3 Multistage stochastic programming

The two-stage decision-making process can be generalized to account for multiple stages. Multistage stochastic programming models allow recourse decisions in each stage coming after stage one. Hence, they are also fully adaptive to the uncertainty realization. An example of a multistage scenario tree is shown in Figure 7.4, where we have 3 different realizations of the uncertainty parameters for stage two and three, and end up with $3^2 = 9$ scenarios in total. In stage two, the decision-maker is only aware of the uncertainties realized at stage two; the parameters in stage three cannot be realized until the time goes to stage three.

The general multistage stochastic programming formulation is given as follows (Birge and Louveaux, 2011):

$$\min\ c_1x_1 + \mathbb{E}_{\xi_2}[\min c_2(\omega_2)x_2(\omega_2) + ... + \mathbb{E}_{\xi_H}[\min c_H(\omega_H)x_H(\omega_H)]...] \tag{7.7a}$$

$$\text{s.t.}\quad W_1x_1 = h_1 \tag{7.7b}$$

$$T_1(\omega_2)x_1 + W_2x_2(\omega_2) = h_2(\omega) \tag{7.7c}$$

$$... \tag{7.7d}$$

$$T_{H-1}(\omega_H)x_{H-1}(\omega_{H-1}) + W_Hx_H(\omega_H) = h_H(\omega) \tag{7.7e}$$

$$x_1 \in X_1; x_t(\omega_t) \in X_t, t = 2, ..., H; \tag{7.7f}$$

where we have $H$ stages. The set of scenarios in stage $t$ is represented as $\omega_t$. The

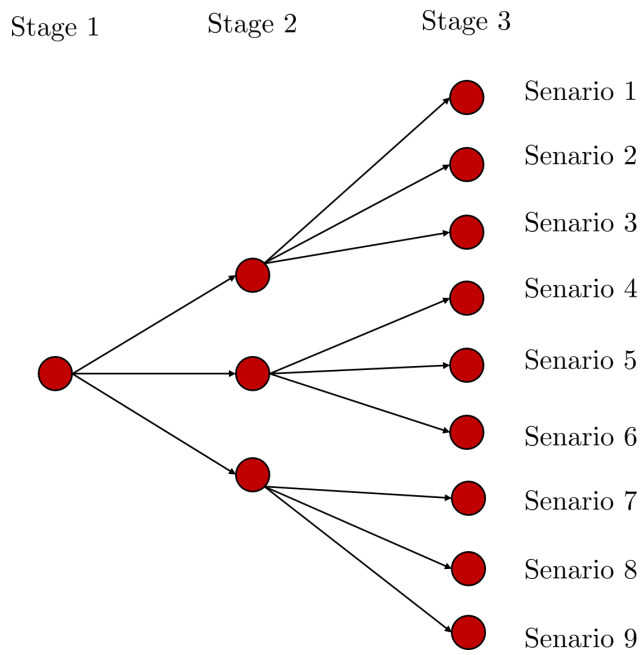


Figure 7.4: Illustration of a scenario tree with 3 stages and 3 realizations per stage

*deterministic equivalent* of the general multistage stochastic programming is then defined as follows (Birge and Louveaux, 2011):

$$\min_{x \in X_1} \{c_1 x_1 + \mathcal{Q}_2(x_1) : W_1 x_1 = h_1\} \tag{7.8}$$

where the expected value function for stage $t+1$ is given by:

$$\mathcal{Q}_{t+1}(x_t) = \mathbb{E}_{\xi_{t+1}}[Q_{t+1}(x_t, \xi_{t+1}(\omega))] \tag{7.9}$$

for all $t$ to obtain the recursion for $t = 2, ..., H-1$,

$$Q_t(x_{t-1}, \xi_t(\omega)) = \left\{ \min_{x(\omega) \in X_t} c_t(\omega) x_t(\omega) + \mathcal{Q}_{t+1}(x_t) : W_t x_t(\omega) = h_t(\omega) - T_{t-1}(\omega) x_{t-1} \right\} \tag{7.10}$$

For the terminal condition $t = H$, we have:

$$Q_H(x_{H-1}, \xi_H(\omega)) = \left\{ \min_{x(\omega) \in X_H} c_H(\omega) x_H(\omega) : W_H x_H(\omega) = h_H(\omega) - T_{H-1}(\omega) x_{H-1} \right\} \tag{7.11}$$

Solving multistage stochastic programming problem is computationally challenging since the size of the problem grows exponentially with the number of stages. One option is to solve problem (7.7), i.e., the deterministic equivalent problem with a finite number of scenarios. Decomposition algorithms can also be used to solve multistage stochastic programs under some assumptions. For example, nested Benders decomposition (Birge, 1985b) can

solve linear multistage stochastic programming problem. For more details about solution procedures, we refer the readers to the review paper (Torres et al., 2019).

## 7.4 Case study

In the case study, we use a data set that has 9 wells in a single wellpad. We assume that none of the operations have been performed on the wells. All the wells are allowed to be developed at any time in the planning horizon. Each of the four operations takes one to two weeks to finish. If all the operations for the 9 wells are performed, 41 weeks are needed. Therefore, we assume that the whole planning horizon is slightly over 41 weeks in the case studies. The details of the length of the planning horizon can be found in the subsections below. We assume that a well can be "partially" completed, which is different from the assumption in the problem statement for the deterministic model in section 7.1 because when the price becomes low, it can be unprofitable to complete the wells.

To showcase whether using stochastic programming can create value, we apply both two-stage and multistage stochastic programming to the single pad development problem. In order to come up with the mathematical formulation of the stochastic programming models, we first need to have a scenario tree that describes the realization of uncertain parameters. We need to duplicate the decisions in the deterministic model described in Appendix A for each node in the scenario tree, i.e., different recourse decisions can be made for each realization of the uncertain parameters. We also evaluate the impact of the variance of the prices in the scenarios on the optimal solution of the two-stage stochastic programs. The details can be found in the following subsections.

### 7.4.1 Two-stage stochastic programming with high price variance

We investigate price uncertainty by formulating a two-stage stochastic programming problem. The whole planning horizon is 45 weeks with the decisions of the first 20 weeks corresponding to the first stage decisions and the decisions of week 21 - week 45 corresponding

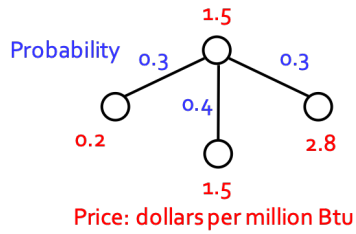


Figure 7.5: The scenario tree for two-stage stochastic programming with high price variance

to the second stage decisions. The scenario tree for this problem is shown in Figure 7.5. We assume that the prices are 0.2, 1.5, 2.8 dollars per million Btu for the three scenarios, which are shown in red in Figure 7.5. The probabilities for each scenario are shown in blue in Figure 7.5, being 0.3, 0.4, 0.3, respectively. It should be noted that the prices here are lower than the natural gas spot price shown in Figure 7.2. This is because in the proposed MILP model we do not consider all the costs that are needed to deliver natural gas to the customers, for example, transportation costs are not considered in the model. Therefore, the natural gas prices should be discounted in order to correctly characterize whether developing the wells is profitable or not. We also assume that the prices are constant after the first stage, i.e., the price outside the planning horizon is the same as the price in the second stage of the stochastic programming problem.

All the problems are solved using CPLEX 12.7 (CPLEX, 2016) on the 12 processors of an Intel Xeon (2.67GHz) machine with 64 GB RAM. The walltime limit is set to 12 hours. The computational statistics of the deterministic problem and the stochastic programming problem including the number of binary and continuous variables, the number of constraints, walltime, optimality gap, are shown in Table 7.1. The stochastic model cannot be solved to optimality within our time limit.

Table 7.1: Computational statistics of the deterministic and the stochastic model with high price variance

| | Binary Var | Continuous Var | Constraints | Walltime | gap |
|---|---|---|---|---|---|
| Deterministic | 3655 | 867 | 4524 | 1,112 secs | 0.01% |
| Stochastic | 9,963 | 1,859 | 12,828 | 12 hrs | 2.41% |

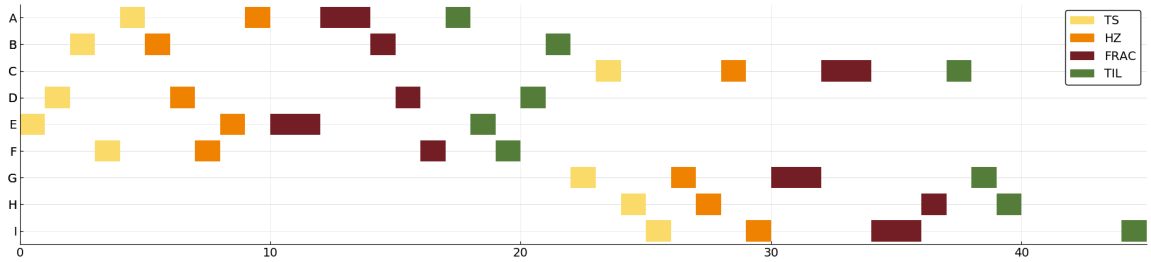


Figure 7.6: Gantt Chart of the deterministic problem

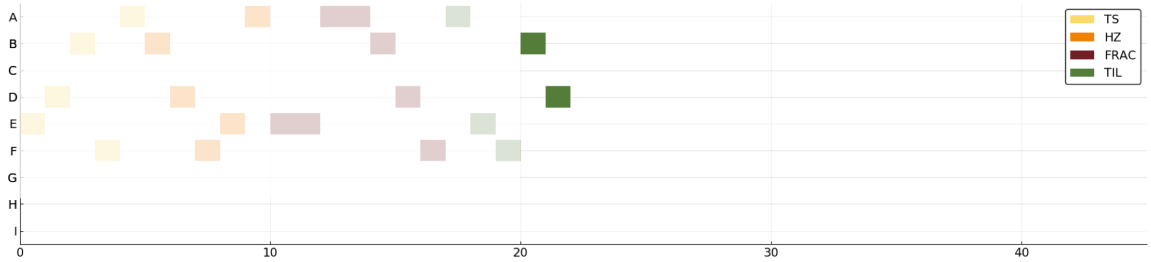


Figure 7.7: Gantt Chart of the expected value solution for scenario price=0.2

To compare the difference between the deterministic model and the stochastic programming model, we first solve the deterministic model with the price of natural gas fixed at 1.5 dollars per Btu. The Gantt chart for the deterministic problem is shown in Figure 7.6. All the 9 wells are completed in the deterministic solution. Five wells are turned in line around week 20. The other four wells are turned in line at the end of the planning horizon.

Then we fix the first stage decisions and check how it performs in different scenarios, i.e., the decisions from week 1 to week 20 are fixed at the corresponding optimal solution of the deterministic model and we only optimize the decisions from week 21 to week 45 for the three scenarios, respectively. The Gantt charts for prices equal to 0.2, 1.5, 2.8 dollars per Btu, are shown in Figures 7.7, 7.8, and 7.9, where the first stage decisions are obscured to denote that they are fixed based on the deterministic model.

In the scenario when the price equals to 0.2 dollar per Btu, the optimal solution only turns in line the two wells that have not been completed in the first stage. No new wells are developed in stage two when the price is low. However, the scenarios when the price is 1.5 or 2.8 dollars per Btu, all the 9 wells are completed within the planning horizon.

The stochastic solutions for the three scenarios are shown in Figures 7.10, 7.11, and 7.12.

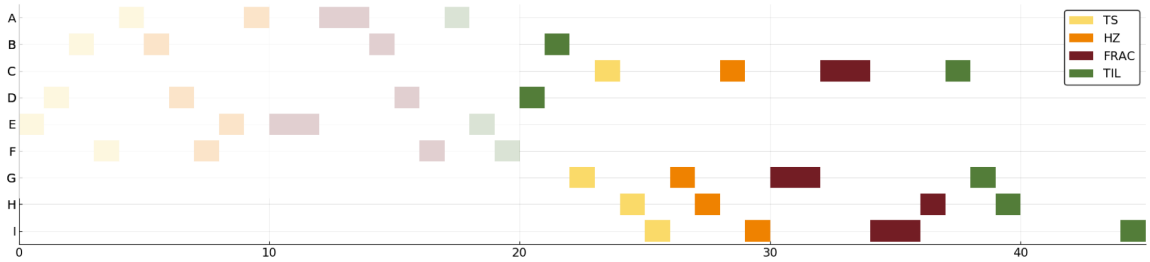


Figure 7.8: Gantt Chart of the expected value solution for scenario price=1.5

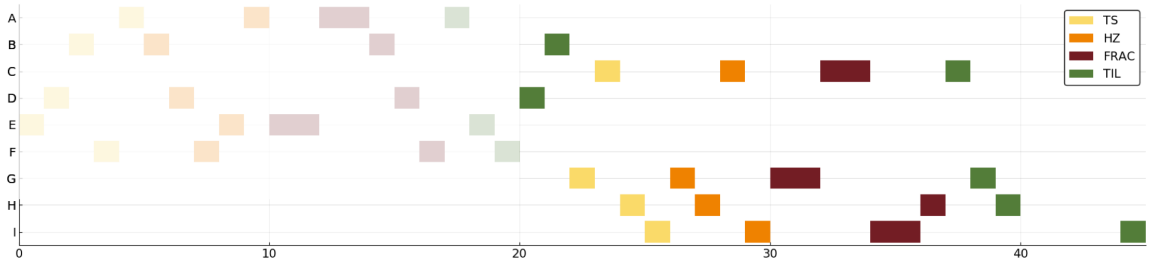


Figure 7.9: Gantt Chart of the expected value solution for scenario price=2.8

Compared with the deterministic solution that fractures five wells in the first stage, the stochastic solution only fractures three wells in the first stage. The stochastic solution tries to wait until the second stage to decide if more wells should be fractured. In the scenario when the price is 0.2 dollars per Btu, no more wells are fractured in the second stage. When the prices are greater than or equal to 1.5 dollars per Btu, all the wells are completed in stage 2. Note that two wells are top set at the end of stage 1 so that the wells can be turned in line faster once the operators realize that the price is going up. Moreover, although the two top set wells are turned in line in the low price scenario, this does not sacrifice the overall expected NPV significantly since top setting is the cheapest operation among the four operations.

In order to demonstrate the value that stochastic programming can create, we show the NPVs of the expected solution and the stochastic solution in all the three scenarios in Table 7.2. While the expected value solution has slightly larger NPVs when the prices are medium or high, it has a highly negative NPV when the price is 0.2 dollars per Btu. On the other hand, the stochastic solution tries to delay development decisions to stage two as much as possible and has higher NPV than the expected value solution in the low price scenario.

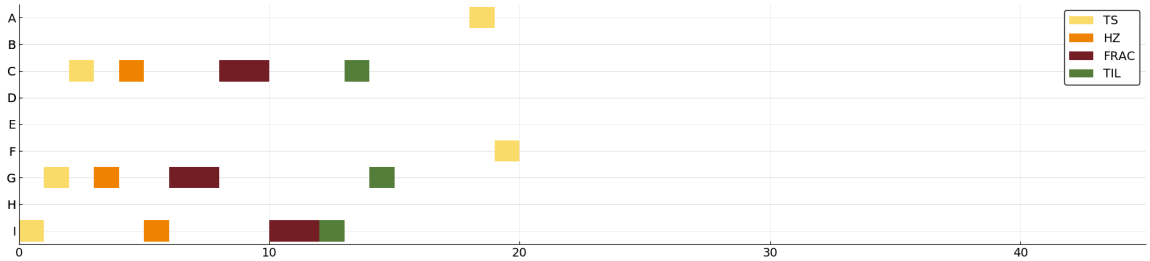


Figure 7.10: Gantt Chart of the stochastic solution for scenario price=0.2

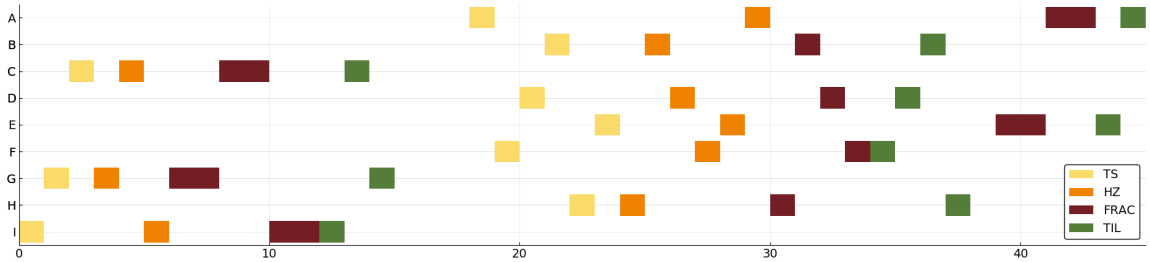


Figure 7.11: Gantt Chart of the stochastic solution for scenario price=1.5

By taking the expected NPV, we can calculate that the value of stochastic solution is 3.50 million dollars.

### 7.4.2 Two-stage stochastic programming with low price variance

In this subsection, we change the assumption of the scenario tree slightly compared with subsection 7.4.1. The scenario tree is shown in Figure 7.13 when the price has smaller variance compared with the scenario tree in Figure 7.5. The prices in the three scenarios are 0.75, 1.5, and 2.25 dollars per Btu.

We solve the same two-stage stochastic programming problem with week 1 to week 20

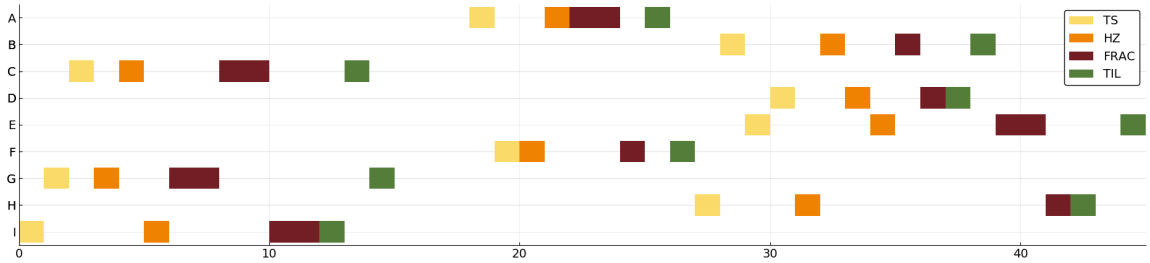


Figure 7.12: Gantt Chart of the stochastic solution for scenario price=2.8

Table 7.2: NPVs of the expected value and the stochastic solution (million dollars)

| Scenario | Price | NPV(Expected value) | NPV(Stochastic Solution) |
|---|---|---|---|
| 1 | 0.2 | -24.28 | -7.74 |
| 2 | 1.5 | 71.45 | 70.86 |
| 3 | 2.8 | 182.58 | 178.48 |

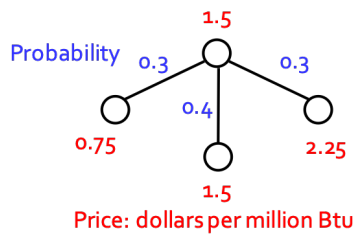


Figure 7.13: The scenario tree for two-stage stochastic programming with low price variance

as the first stage, and week 21 to week 45 as the second stage. It turns out that the optimal solutions of the three scenarios are all similar to the expected value solution shown in Figure 7.6. Hence, there is no value in solving the stochastic programming model. The main reason is that in all the three scenarios natural gas is profitable. Therefore, all the wells are completed as in the deterministic model for all the three scenarios.

### 7.4.3 Multistage stochastic programming with high price variance

Multistage stochastic programming is a more accurate way to model the single pad development problem since it considers a sequence of decisions that react to outcomes that evolve over time. We apply multistage stochastic programming to our single pad development problem with high price variance. We assume that the whole planning horizon is 48 weeks. The planning horizon is equally divided into three stages. In each stage, the realizations of price can be 0.2, 1.5, and 2.8 dollars per Btu with probabilities 0.3, 0.4, and 0.3, respectively. The scenario tree for the multistage problem is shown in Figure 7.14. We also assume that the prices outside the planning horizon remain equal to the prices in the third stage.

The stochastic programming model has 31,213 binary variables, 4,153 continuous variables, and 49,801 constraints. It is solved using CPLEX 12.7 (CPLEX, 2016) on the 12

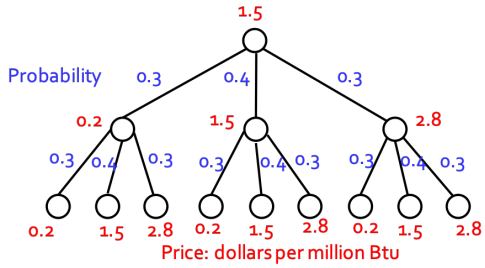


Figure 7.14: The scenario tree for multistage stochastic programming with high price variance

processors of an Intel Xeon (2.67GHz) machine with 64 GB RAM for 12 hours. The CPLEX solver can obtain an optimality gap of 3.76% within the time limit. The best expected NPV from CPLEX is 79.94 million dollars. To quantify the value of using stochastic programming, we use the concept of the value of stochastic solution for multistage stochastic programming from Escudero et al. (2007). In their definition, the expected results of using the expected value solution at stage $t$ ($EEV_t$) is obtained by fixing the first $t-1$ stage decisions to the optimal solution of the expected value problem and solve the rest of the stochastic programming problem. The value of stochastic solution at stage $t$ ($VSS_t$) is defined as $VSS_t = EEV_t - RP$. For the detailed definition, we refer the readers to Escudero et al. (2007). In this case, $EEV_2 = 74.62$ million dollars, $EEV_3 = 73.40$ million dollars, $VSS_2 = 5.33$ million dollars, $VSS_3 = 6.55$ million dollars.

In Figures 7.15 to 7.18, we show the Gantt charts of some scenarios. In both Figure 7.15 and Figure 7.17 when the price in the third stage and outside the planning horizon is only 0.2 dollar per Btu, not all the wells are completed in stage three because the natural gas price is not high enough to make the business profitable. On the other hand, when the price is greater than or equal to 1.5 dollars, it is preferable to complete all the 9 wells, which corresponds to the scenarios shown in Figures 7.16 and 7.18. Since the price in the third stage determines whether natural gas is profitable outside the planning horizon, all the optimal solutions try to delay the decisions to the third stage, i.e., no operation is performed in the last few days of stage two.

We can also observe that the price in stage two affects the number of wells that are turned in line. When the price is 0.2 dollar per Btu in stage two, only one well is turned

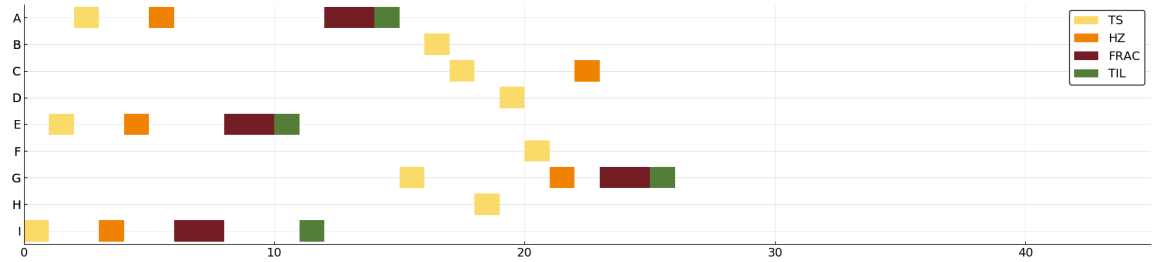


Figure 7.15: Gantt Chart of the stochastic solution for scenario price=0.2, 0.2, at stage two and three, respectively

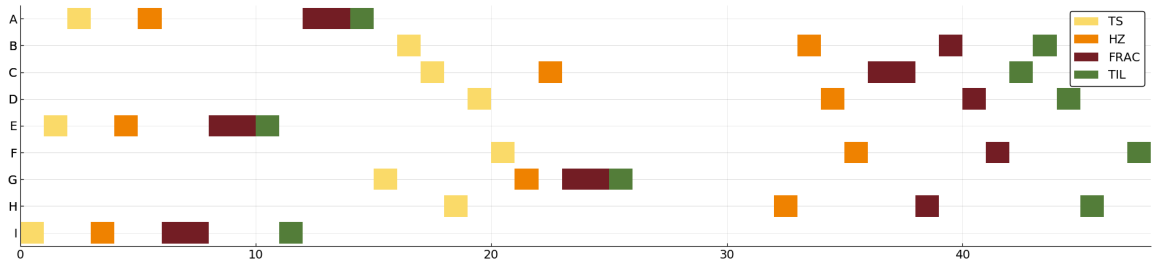


Figure 7.16: Gantt Chart of the stochastic solution for scenario price=0.2, 1.5, at stage two and three, respectively

in line (see Figures 7.15 and 7.16). When the price is greater than or equal to 1.5 dollars per Btu in stage two (see Figures 7.17 and 7.18), two wells are turned in line in stage two. Because of the higher price in stage two in Figures 7.17 and 7.18, one more well is turned in line to obtain the revenue of natural gas early in stage two.

The similarity of the multistage and the two-stage stochastic programming solution is that both of them try to delay the decisions to the last stage so that the uncertainty of the prices are fully realized. The major difference is that the multistage solution allows the uncertainties to evolve every quarter, which is a better approximation of what happens in

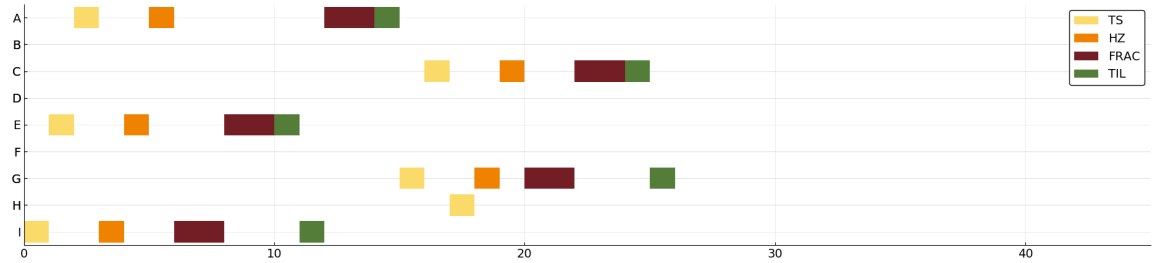


Figure 7.17: Gantt Chart of the stochastic solution for scenario price=1.5, 0.2, at stage two and three, respectively

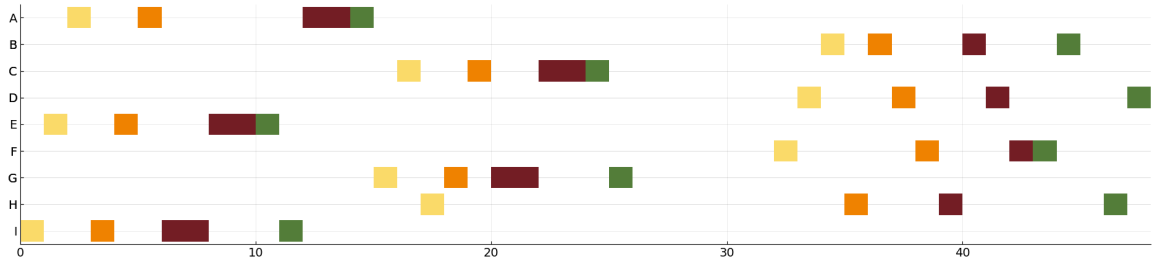


Figure 7.18: Gantt Chart of the stochastic solution for scenario price=1.5, 1.5, at stage two and three, respectively

practice.

## 7.5 Conclusions

In this chapter, we study the shale gas pad development problem under price uncertainty. The deterministic model is similar to the MILP model proposed by Ondeck et al. (2019) where a wellpad is given with several prospective wells and the upstream operator needs to determine the sequence of operations on the wellpad. To extend the work of Ondeck et al. (2019), we investigate how price uncertainties can affect the decision-making in this context. The mathematical framework that we use in this chapter is stochastic programming, which is regarded as a risk-neutral approach to hedge against uncertainty. We introduce some concepts to quantify the value that stochastic programming can create, such as the value of stochastic solution (VSS). We apply both two-stage and multistage stochastic programming to the development of a wellpad with 9 wells under price uncertainty. When the variance of price is high, we can obtain values using stochastic programming on the order of million dollars. The stochastic solutions in both two-stage and multistage try to delay the development decisions to the last stage when the uncertainty of the prices are fully realized. We also demonstrate that multistage stochastic programming is a better approximation of what happens in the real world than two-stage stochastic programming since it allows the uncertainties to evolve every quarterly.

However, when the price variance is low, there is no value of using stochastic programming since all the scenarios have the same recourse decisions. This gives us some insights on when

to use stochastic programming. There may not be any value of using stochastic programming even if there are uncertain parameters in the model. Stochastic programming can only add value if there is something that we can do differently than the deterministic solution.

It should be noted that the generation of scenario trees can be improved by using more realistic forecast models. The aim of using the high and low variance scenarios trees is to evaluate the stochastic programming approach. We will leave the incorporation and better forecast models and expert knowledge in building the scenario as one future direction.

Futhermore, future work can also concentrate on extending the model to multiple pads development problem, and considering the mobilization of crew and equipment between the wellpads. We expect the optimization problem to grow even larger once more wellpads are taken into account. Therefore, better solution techniques need to be developed to solve the large-scale stochastic programming problem with higher efficiency.

# Part II

# Long Term Optimization of Electric Power Systems

# Chapter 8

# Mixed-integer Linear Programming Models and Algorithms for Generation and Transmission Expansion Planning of Power Systems

## 8.1 Problem Statement and Assumptions

Given is a geographical region with existing and potential generating units and transmissions lines. The problem consists in making capacity expansion decisions for both generation and transmission while considering the unit commitment and power flow constraints at the operational level.

### 8.1.1 Generation representation

The existing and potential generation technologies are similar to the ones used in Lara et al. (2018), i.e.,

- For the existing generators we consider: (a) coal: steam turbine (coal-st-old); (b) natural gas: boiler plants with steam turbine (ng-st-old), combustion turbine (ng-ct-

old), and combined-cycle (ng-cc-old); (c) nuclear: steam turbine (nuc-st-old); (d) solar: photo-voltaic (pv-old); (e) wind: wind turbine (wind-old).

- For the potential generators we consider: (a) coal: without (coal-new) and with carbon capture (coal-ccs-new); (b) natural gas: combustion turbine (ng-ct-new), combined-cycle without (ng-cc-new) and with carbon capture (ng-cc-ccs-new); (c) nuclear: steam turbine (nuc-st-new); (d) solar: photo-voltaic (pv- new) and concentrated solar power (csp-new); (e) wind: wind turbine (wind-new).

Also known are: the generating units' nameplate (maximum) capacity; expected lifetime; fixed and variable operating costs; fixed and variable start-up cost; cost for extending their lifetimes; $CO_2$ emission factor and carbon tax, if applicable; fuel price, if applicable; and operating characteristics such as ramp-up/ramp-down rates, operating limits, contribution to spinning and quick start fraction for thermal generators, and capacity factor for renewable generators.

For the case of existing generators, their age at the beginning of the study horizon and location are also known. For the case of potential generators, the capital cost and the maximum yearly installation of each generation technology are also given. Also given is a set of potential storage units, with specified technology (e.g., lithium ion, lead-acid, and flow batteries), capital cost, power rating, rated energy capacity, charge and discharge efficiency, and storage lifetime. Additionally, the projected load demand is given for each location.

We assume that the generators using the same type of technology are homogeneous, i.e., their design parameters are identical. For example, all the coal-st-old generators have the same parameters, which can be obtained by performing aggregation on the existing generators that use coal steam turbines. Note that although the renewable generators of the same technology have the same design parameters under our assumption, they can have different capacity factors depending on the weather conditions of the region in which they are installed.

### 8.1.2 Transmission representation

Given are existing and candidate transmission lines between any of the two neighboring buses. The susceptance, distance, and capacity of each transmission line are known. For the existing transmission lines, we assume that they will not reach their life expectancy during the planning horizon, i.e., we do not consider the retirement of transmission lines. For the candidate transmission lines, the capital cost of each transmission line is known.

We use DC power flow equations to calculate the power flow in each transmission line. These equations are built based on Kirchhoff's voltage and current laws which differ from the network flow model used in the work of Lara et al. (2018). In the network flow model, the transmission network is represented similarly to pipelines where the flows only observe energy balance at each node while ignoring Kirchhoff's laws.

### 8.1.3 Temporal representation

The GTEP model integrates unit commitment decisions to evaluate the hourly operation requirements. Given that the planning horizon of the GTEP problem can be as long as 10 to 30 years, solving the long-term planning problem with operating decisions in every hour of the planning horizon is intractable. Therefore, a simplification is needed to make the problem solvable, while representing the hourly fluctuations of the load and renewable profiles.

Several works propose to select a few representative days (Mallapragada et al., 2018; Teichgraeber and Brandt, 2019; Scott et al., 2019) from the full data set to represent the hourly fluctuations. To keep the chronology of the hourly historical data, the time series for the loads and the capacity factors corresponding to the same day are concatenated as a single vector, which will be used as the input to some clustering algorithms, such as $k$-means, and hierarchical clustering. After performing the clustering on the full time series data set, the time series corresponding to the representative days are the centroids or medoids of the clusters. The details can be found in subsection 5.1.

### 8.1.4 Spatial representation

GTEP is typically performed on large scale power systems which consists of thousands of buses, such as ERCOT, SPP, PJM, MISO, etc. In most cases, it is intractable for GTEP to model each bus. Therefore, we adopt a similar approach as in Lara et al. (2018) to reduce the spatial complexity of the problem. The area of interest is divided into several regions that have similar climate (e.g., wind speed and solar incidence over time), and load profiles. As we describe in the generation representation subsection, all the generators using the same technology have the same parameters. On the other hand, for the renewable generators, the capacity factors are dependent on the location at which they are installed. We assume that the capacity factors of the renewable generators in the same region are the same.

We assume that all the generators and loads are located at the center of each region. Since each region is treated as one bus in the power flow model, we only consider the tielines between two neighboring regions. We assume that the two ends of each tieline are the centers of the two regions it connects. All the tielines are assumed to have the same voltage, susceptance, and capacity. An example of the proposed spatial representation approach is shown in Figure 8.1. The ERCOT region is divided into five regions, Panhandle, Northeast, West, South, and Coast. The center of each region is specified as one of the cities in the region. The existing transmission lines are represented as solid lines while the candidate transmission lines are represented as dashed lines. Each region has generator clusters corresponding to different technologies.

The aggregation of the generating units is a simplification of the problem that may yield suboptimal solution compared with modeling each generator individually. Such simplification is necessary to make the problem tractable. In order to obtain a feasible solution to the real physical system, i.e., the unit commitment decisions of each generator, one could perform a disaggregation heuristics on the aggregated solution. We will leave developing these heuristics as future work.

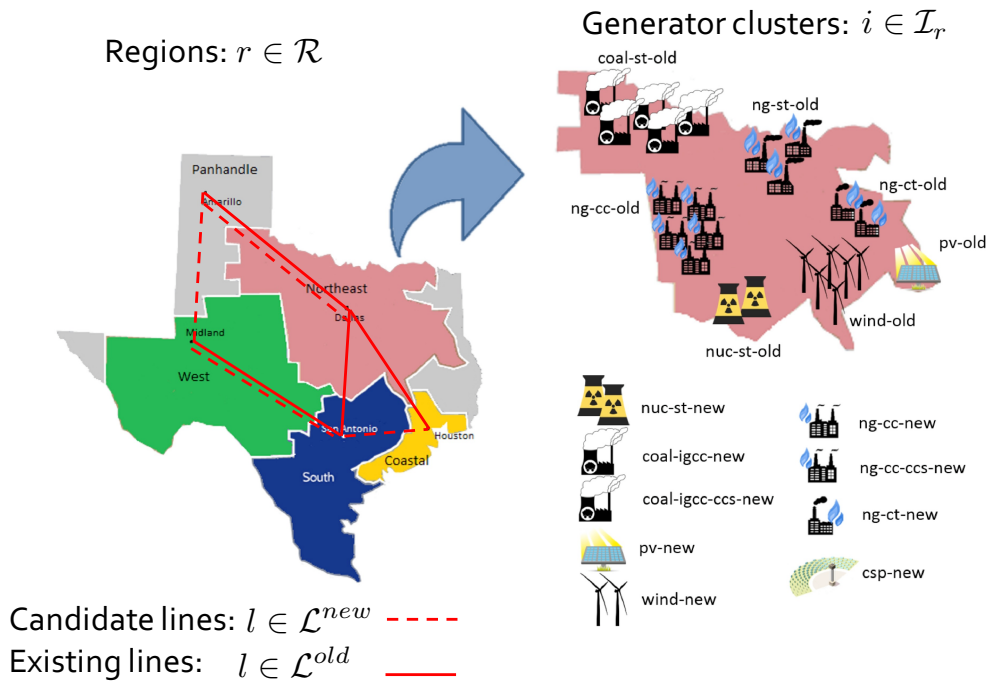


Figure 8.1: Spatial representation of the five ERCOT regions' generator clusters and transmission lines

### 8.1.5 Decisions and objective

With the above input data, spatial and temporal representations, the proposed GTEP model is to decide: a) when and where to install new generators, storage units and transmission lines; b) when to retire generators and storage units; c) whether or not to extend the life of the generators that reached their expected lifetime; d) unit commitment of the thermal generators during the representative days; e) power generations of the generator clusters and power flows through the transmission lines. The objective is to minimize the overall cost including operating, investment, and environmental costs (e.g., carbon tax and renewable generation quota).

## 8.2 MILP Formulation

This section presents a deterministic MILP formulation for the GTEP problem. Most of the MILP formulation is similar to that in Lara et al. (2018). Here, we emphasize the transmission expansion formulation that is added. Note that if an index appears in a summation or next to a $\forall$ symbol without a set, all elements in the corresponding set should be considered. The nomenclature for sets, parameters, and variables used in the MILP formulation are provided in Appendix A in supplementary material.

### 8.2.1 Transmission expansion constraints

Transmission line balance constraints. A succinct version of Appendix A is attached at the end of this chapter. Variable $ntb_{l,t}$ denotes whether or not candidate transmission line $l$ is built in year $t$. Variable $nte_{l,t}$ denotes whether transmission line $l$ has been installed in year $t$. Equation (8.1) represents the balance of transmission lines.

$$nte_{l,t} = nte_{l,t-1} + ntb_{l,t} \quad \forall l \in \mathcal{L}^{\text{new}}, t \tag{8.1}$$

The DC transmission constraints calculate and limit the power flows through the transmission lines. Parameter $B_l$ represents the susceptance of line $l$. $\theta_{s(l),t,d,s}$, $\theta_{r(l),t,d,s}$ are the phase angles of the buses that are the sending-end and the receiving-end of line $l$, respectively, in year $t$, representative day $d$, and sub-period (hour) $s$. The existing transmission lines have to satisfy the DC power flow equation (8.2).

$$p^{\text{flow}}_{l,t,d,s} = B_l\big(\theta_{s(l),t,d,s} - \theta_{r(l),t,d,s}\big) \quad \forall l \in \mathcal{L}^{\text{old}}, t, d, s \tag{8.2}$$

The power flow through each transmission line is bounded. Parameter $F^{\max}_l$ represents the capacity of transmission line $l$. Thus:

$$-F^{\max}_l \leq p^{\text{flow}}_{l,t,d,s} \leq F^{\max}_l \quad \forall l \in \mathcal{L}^{\text{old}}, t, d, s \tag{8.3}$$

For the candidate transmission lines, we can write the following disjunction, where $NTE_{l,t}$ is a logic variable whose value can be True or False indicating whether or not transmission line $l$ is installed in year $t$. If line $l$ already exists in year $t$, the corresponding power flow has to

satisfy DC power flow equation and upper and lower bounds. Otherwise, the corresponding power flow is zero.

$$\begin{bmatrix} NTE_{l,t} \\ p^{\text{flow}}_{l,t,d,s} = B_l(\theta_{s(l),t,d,s} - \theta_{r(l),t,d,s}) \\ -F^{\max}_l \leq p^{\text{flow}}_{l,t,d,s} \leq F^{\max}_l \end{bmatrix} \vee \begin{bmatrix} \neg NTE_{l,t} \\ p^{\text{flow}}_{l,t,d,s} = 0 \end{bmatrix} \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.4}$$

Standard approaches, i.e., big-M reformulation and hull reformulation (Grossmann and Trespalacios, 2013), are available to reformulate disjunctions (8.4) into mixed integer constraints.

The big-M formulation of the disjunction is,

$$-M_l(1 - nte_{l,t}) \leq p^{\text{flow}}_{l,t,d,s} - B_l(\theta_{s(l),t,d,s} - \theta_{r(l),t,d,s}) \leq M_l(1 - nte_{l,t}) \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.5}$$

$$-F^{\max}_l nte_{l,t} \leq p^{\text{flow}}_{l,t,d,s} \leq F^{\max}_l nte_{l,t} \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.6}$$

This big-M formulation is most commonly used in the literature (Conejo et al., 2016b) for TEP.

The hull formulation is,

$$p^{\text{flow}}_{l,t,d,s} = B_l \Delta\theta^1_{l,t,d,s} \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.7}$$

$$\theta_{s(l),t,d,s} - \theta_{r(l),t,d,s} = \Delta\theta^1_{l,t,d,s} + \Delta\theta^2_{l,t,d,s} \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.8}$$

$$-\pi \cdot nte_{l,t} \leq \Delta\theta^1_{l,t,d,s} \leq \pi \cdot nte_{l,t} \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.9}$$

$$-\pi(1 - nte_{l,t}) \leq \Delta\theta^2_{l,t,d,s} \leq \pi(1 - nte_{l,t}) \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.10}$$

where $\Delta\theta^1_{l,t,d,s}$ and $\Delta\theta^2_{l,t,d,s}$ are disaggregated variables for the angle difference of transmission line $l$. Variable $\Delta\theta^1_{l,t,d,s}$ is equal to the angle difference if transmission line $l$ has been installed in year $t$. Otherwise, $\Delta\theta^2_{l,t,d,s}$ equals to the angle difference. In addition to equations (8.7)-(8.10), equation (8.6) needs to be included in the hull formulation.

The hull formulation has more continuous variables than the big-M formulation but it

avoids using the big-M parameters of equations (8.5). The hull formulation provides the intersection of the convex hull of each disjunction in (8.4). Therefore, the hull formulation can provide a tighter LP relaxation at the expense of solving larger LPs at each node of a branch-and-bound algorithm.

Alternative big-M formulation: Besides the big-M and hull formulations, an alternative big-M formulation is proposed by Bahiense et al. (2001). In this formulation, additional continuous variables $p^{\text{flow}+}_{l,t,d,s}$, $p^{\text{flow}-}_{l,t,d,s}$, $\Delta\theta^{+}_{l,t,d,s}$, $\Delta\theta^{-}_{l,t,d,s}$, are introduced, where the superscript '+' means that the flow is in the same direction as the nominal direction of transmission line $l$, i.e., from the sending-end node $s(l)$ to the receiving-end node $r(l)$; superscript '-' means the opposite direction. By defining these new continuous variables, equation (8.5) is replaced by equations (8.11) to (8.14) and equation (8.6) is replaced by equations (8.17) and (8.18). Bahiense et al. (2001) claim that the alternative big-M formulation is tighter than the big-M formulation. However, we prove that they have the same feasible region if we project the feasible region of the alternative big-M formulation onto the space of $(p^{\text{flow}}_{l,t,d,s}, \theta_{s(l),t,d,s}, \theta_{r(l),t,d,s}, nte_{l,t})$ in Theorem 1. The proof of Theorem 1 can be found in Appendix B of supplementary material

$$p^{\text{flow}+}_{l,t,d,s} - B_l \Delta\theta^{+}_{l,t,d,s} \leq 0 \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.11}$$

$$p^{\text{flow}-}_{l,t,d,s} - B_l \Delta\theta^{-}_{l,t,d,s} \leq 0 \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.12}$$

$$p^{\text{flow}+}_{l,t,d,s} - B_l \Delta\theta^{+}_{l,t,d,s} \geq -M_l(1 - nte_{l,t}) \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.13}$$

$$p^{\text{flow}-}_{l,t,d,s} - B_l \Delta\theta^{-}_{l,t,d,s} \geq -M_l(1 - nte_{l,t}) \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.14}$$

$$p^{\text{flow}}_{l,t,d,s} = p^{\text{flow}+}_{l,t,d,s} - p^{\text{flow}-}_{l,t,d,s} \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.15}$$

$$\theta_{s(l),t,d,s} - \theta_{r(l),t,d,s} = \Delta\theta^{+}_{l,t,d,s} - \Delta\theta^{-}_{l,t,d,s} \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.16}$$

$$p^{\text{flow}+}_{l,t,d,s} \leq F^{\max}_{l} nte_{l,t} \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.17}$$

$$p^{\text{flow}-}_{l,t,d,s} \leq F^{\max}_{l} nte_{l,t} \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.18}$$

$$p^{\text{flow}+}_{l,t,d,s}, p^{\text{flow}-}_{l,t,d,s}, \Delta\theta^{+}_{l,t,d,s}, \Delta\theta^{-}_{l,t,d,s} \geq 0 \quad \forall l \in \mathcal{L}^{\text{new}}, t, d, s \tag{8.19}$$

**Theorem 5.** The alternative big-M formulation (ABM) has the same feasible region as the big-M (BM) formulation if the feasible region of ABM is projected to the space of $\big\{ \bigoplus_{l\in\mathcal{L}^{\text{new}}, t\in\mathcal{T}, d\in\mathcal{D}, s\in\mathcal{S}} (p^{\text{flow}}_{l,t,d,s}, \theta_{s(l),t,d,s}, \theta_{r(l),t,d,s}, nte_{l,t}) \big\}$, where the symbol '$\oplus$' means the concatenation of all the variables $(p^{\text{flow}}_{l,t,d,s}, \theta_{s(l),t,d,s}, \theta_{r(l),t,d,s}, nte_{l,t})$ over the set $\mathcal{L}^{\text{new}}, \mathcal{T}, \mathcal{D}, \mathcal{S}$.

### 8.2.2 Other constraints

All other constraints including operational constraints, investment-related constraints, generator balance constraints, storage constraints, are similar to those of the MILP formulation proposed by Lara et al. (2018). The details of these constraints and the nomenclature can be found in Appendix A in supplementary material. A succinct version of Appendix A is attached at the end of this chapter.

## 8.3 Solution techniques

Given the large size of the proposed GTEP problem, tailored solution approaches need to be developed. In this section, we describe two solution algorithms: a) nested Benders decomposition adapted from (Birge, 1985a; Zou et al., 2019), which has been used by Lara et al. (2018) to solve the GEP model. b) a tailored Benders decomposition. Both of the two algorithms exploit the structure of the GTEP problem.

### 8.3.1 Nested Benders decomposition

Lara et al. (2018) apply a nested Benders decomposition algorithm to solve their GEP model. Like in the GEP model, the nested Benders decomposition algorithm decomposes the fullspace of the GTEP problem by year. Note that the linking constraints for two consecutive years are the investment related constraints. For the investment decisions in transmission lines, the linking constraints are described by equation (8.1), i.e., the balance of candidate transmission lines. Similarly, there are linking constraints corresponding to the number of thermal generators $ngo^{\text{th}}_{i,r,t}$, the number of renewable generators $ngo^{\text{rn}}_{i,r,t}$, the number of storage units $nso_{j,r,t}$ per region $r$ and year $t$. These linking constraints can be found in equations (8.1), and (E.17), (E.20), (E.24) in Appendix A in supplementary material.

From the above observation, variables $nte_{l,t}$, $ngo^{\text{th}}_{i,r,t}$, $ngo^{\text{rn}}_{i,r,t}$, $nso_{j,r,t}$ can be treated as complicating variables. Once these variables are fixed, the GTEP problem can be decomposed by year. The nested Benders decomposition consists of two phases, i.e., forward pass and backward pass. In the forward pass, the problem is solved sequentially year after year. In each year $t$, the problem is solved in a myopic way, with the complicating variables of year $t-1$ fixed, and the cutting planes generated from the backward pass. The optimal solution is obtained for year $t$. Then the complicating variables are fixed for year $t$ and the problem for year $t+1$ is solved, until we reach the end of the planning horizon.

In the backward pass, cutting planes can be generated by solving the LP relaxations of the planning problem with the complicating variables fixed at the values of the forward pass. The backward pass starts from the last year and sequentially adds cutting planes to the previous year. Since the nested Benders decomposition is developed by Lara et al. (2018) for the GEP model, we do not provide the details of the algorithms. The steps of the nested Benders algorithms are similar to those in Lara et al. (2018), except that in the GTEP problem we introduce new complicating variables $nte_{l,t}$ pertaining to transmission expansion. An additional difference is that while in Lara et al. (2018) three different types of cutting planes are implemented in the backward pass, i.e., Benders cuts, strengthened Benders cuts, and Lagrangean cuts, we only implement Benders cuts to solve the GTEP

problem because this type of cut is computationally cheap.

### 8.3.2 Tailored Benders decomposition algorithm

Instead of solving the GTEP problem sequentially by year as in the nested Benders decomposition, we treat all the investment-related variables as complicating variables and include all these variables in a single Benders master problem.

More specifically, the proposed GTEP model can be represented using the succinct form (8.20) below, where $x_t$ represents all the investment decisions in year $t$, $y_t$ represents all the operating decisions in the representative days for year $t$. Note that the investment decisions are made on a yearly basis indexed by $t$. The operating decisions not only have the index $t$ but also have indices $d$ and $s$, which represent the $d$th representative day in the $s$th hour, respectively. Since we will decompose the problem by year, we omit the indices $d$ and $s$ and simply use $y_t$ to represent all the operating decisions corresponding to year $t$. Equations (8.20c) and (8.20d) are investment related constraints, which correspond to equations (8.1), (E.14)-(E.21), (E.23), (E.24). Equations in (8.20b) describe the operational decisions of each year, such as the power flow equations (8.2) and (8.3). Note that equation (8.20b) can be decomposed by year. Equation (8.20e) represents the integrality constraints and variable bounds that $x_t$ and $y_t$ have to satisfy.

$$\min \sum_{t \in \mathcal{T}} c_t^\top x_t + d_t^\top y_t \tag{8.20a}$$

$$\text{s.t.} \quad A_t x_t + B_t y_t \leq b_t \quad \forall t \in \mathcal{T} \tag{8.20b}$$

$$C_1 x_1 \leq f_1 \tag{8.20c}$$

$$C_{t-1} x_{t-1} + D_t x_t \leq f_t \quad t = 2, 3, \ldots, |\mathcal{T}| \tag{8.20d}$$

$$x_t \in X_t, y_t \in Y_t \quad \forall t \in \mathcal{T} \tag{8.20e}$$

The GTEP problem has a decomposable structure in the sense that if we treat all the investment decisions $x_t$ for all $t \in \mathcal{T}$ as *complicating variables*, the problem can be decomposed by

year. Benders decomposition (Rahmaniani et al., 2016) can be applied to solve this type of problem. We can assign all the investment variables to the Benders master problem and the operating variables $y_t$ to the $t$th subproblem. After solving the Benders master problem, the investment decisions are fixed and each Benders subproblem can be solved independently. Note that there are some integer variables in the operating decisions, such as the number of generators that are on/off. In order to generate valid Benders cuts, we solve the LP relaxation of each Benders subproblem and add the cuts to the Benders master problem. A high level description of the algorithm is provided in Figure 8.2. The formulation of the Benders

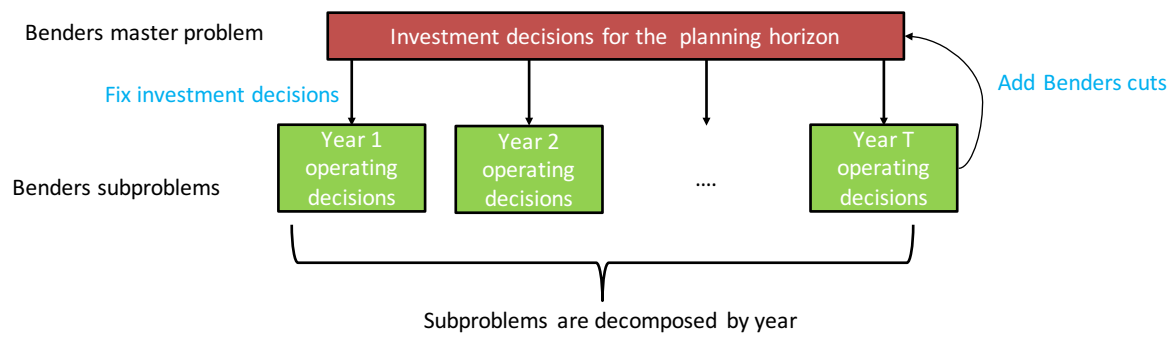


Figure 8.2: Tailored Benders decomposition algorithm applied to the GTEP problem

master problem solved at iteration $k$ is:

$$\min \sum_{t \in \mathcal{T}} c_t^\top x_t + \eta_t \tag{8.21a}$$

$$\text{s.t.} \quad C_1 x_1 \leq f_1 \tag{8.21b}$$

$$C_{t-1} x_{t-1} + D_t x_t \leq f_t \quad t = 2, 3, \ldots, |\mathcal{T}| \tag{8.21c}$$

$$\eta_t \geq \tilde{\eta}_t^{k'} + \left(\mu_t^{k'}\right)^\top \left(\tilde{x}_t^{k'} - x_t\right) \quad t \in \mathcal{T}, k' < k \tag{8.21d}$$

$$x_t \in X_t \quad \forall t \in \mathcal{T} \tag{8.21e}$$

where equation (8.21d) are the Benders cuts generated by solving the Benders subproblems. We denote the optimal solution of the Benders master problem at iteration $k$ as $\tilde{x}_t^k$, $\forall t \in \mathcal{T}$.

Fixing the values of the investment decision variables to the values obtained at the master

problem, i.e., $x_t = \tilde{x}_t^k$, $\forall t \in \mathcal{T}$, we can solve each Benders subproblem independently for each year $t \in \mathcal{T}$:

$$\tilde{\eta}_t^k = \min d_t^\top y_t \tag{8.22a}$$

$$\text{s.t.} \quad x_t = \tilde{x}_t^k \tag{8.22b}$$

$$A_t x_t + B_t y_t \leq b_t \tag{8.22c}$$

$$y_t \in \tilde{Y}_t \tag{8.22d}$$

where all the integer variables in $y_t$ are relaxed and set $\tilde{Y}_t$ represents set $Y_t$ without the integrality constraints, i.e, $\tilde{Y}_t$ only represents variables bounds for $y_t$. Let $\mu_t^k$ be the optimal dual multiplier for equation (8.22b). A valid Benders cut,

$$\eta_t \geq \tilde{\eta}_t^k + \left(\mu_t^k\right)^\top \left(\tilde{x}_t^k - x_t\right)$$

can be generated by solving the $t$th subproblem. The cuts from the subproblems are then added to the master problem via equation (8.21d). Note that the Benders subproblem (8.22) can be infeasible. In this case, a feasibility subproblem should be solved to generate feasibility cuts. Interested readers can refer to (Rahmaniani et al., 2016) for the definitions of feasibility cuts. To simply notation, we assume that the subproblems are feasible here.

At each iteration $k$, the Benders master problem provides a lower bound of the optimal objective function value with relaxed $y_t$ variables, while an upper bound can be calculated as $\sum_{t\in\mathcal{T}} c_t^\top \tilde{x}_t^k + d_t^\top \tilde{y}_t^k$ where $\tilde{x}_t^k$ and $\tilde{y}_t^k$ are the optimal solutions to the master problem and the subproblems, respectively. We keep iterating between the Benders master problem (8.21) and the subproblems (8.22) until the upper bound and the lower bound lie within certain optimality tolerance.

For our computational study, we use the Benders implementation from CPLEX (Bonami et al., 2020b), which is a branch-and-Benders-cut algorithm. We only need to specify the variables in the master problem and the variables in each subproblem and CPLEX automatically solves the GTEP problem using Benders decomposition. Note that the implementation in CPLEX uses a single branch-and-bound tree where Benders cuts are added as lazy con-

straints. The Benders subproblems are solved whenever a feasible solution is found in the branch-and-bound tree of the master problem. The corresponding optimality or feasibility cuts will be added to the master problem dynamically in the single branch-and-bound tree.

Since the integrality constraints of the $y_t$ variables are relaxed within the Benders decomposition algorithm, we can only obtain a lower bound to the original GTEP problem (8.20) through this algorithm. In order to obtain a feasible solution to the original problem, i.e., an upper bound, we can fix the investments decisions $x_t$ to the optimal solution of the Benders decomposition algorithm and solve the operating problem with the integrality constraints of the $y_t$ variables for each year independently. Moreover, as a result of the relaxation of the integrality constraints corresponding to the $y_t$ variables, there will be a gap between the lower bound and the upper bound. However, our computational results in section 8.4 show that this gap is small. The reason is that all the integer variables in $y_t$ are general integer variables instead of binary variables. Typically, mixed-integer programs with general integer variables have good LP relaxations.

## 8.4 Case studies

### 8.4.1 Input data

We carry out a GTEP case study for ERCOT. The spatial representation of the ERCOT region has been discussed in subsection 8.1.4. It is divided into four geographical regions: Northeast, West, Coast and South. Besides these regions, a fifth region, Panhandle, is also included, which is technically outside the ERCOT region, but due to its renewable generation potential, it supplies electricity to the ERCOT regions. Note that in our model, Panhandle is treated as a pure supplier, i.e., it has zero load. The map of the five regions is shown in Figure 8.1.

Each of the five regions are treated as a bus and a DC power flow model is considered. We specify a city for each region as the location of the bus. The center for Northeast, West, Coast, South and Panhandle are Dallas, Midland, Houston, San Antonio, and Amarillo,

respectively. The lengths of the transmission lines are determined by the distance between the centers of any of the two neighboring regions. To test the GTEP model, we assume that no transmission lines are available, i.e., the model will identify the transmission lines to be built. We assume that for each pair of the two neighboring regions, at most 10 candidate transmission lines can be built. The susceptance and capacity of the transmission lines are all the same, which are obtained from a synthetic grid of Texas (Birchfield et al., 2016). The unit capital cost of transmission lines is $1,919,450 per mile obtained from (Andrade and Baldick, 2016).

Old and new generation technologies have been described in subsection 8.1.1. The investment cost, fixed and variable operating costs for different generation technologies are obtained from the National Renewable Energy Laboratory (NREL), available in the 2016 Annual Technology Baseline (ATB) Spreadsheet (Cole et al., 2016). The capital cost, power rating, rated energy capacity, charge and discharge efficiency and storage lifetime of the storage units are from (Schmidt et al., 2017). We consider a 20 year time horizon, in which the first year is 2019. The fuel price data for coal, natural gas and uranium correspond to the reference scenario in U.S. Energy Information Administration (2019a). A discount rate of 5.7% as chosen in Short et al. (2011) is used. We assume that the undiscounted carbon tax is zero in the first year and grows linearly across years to $325/tonne $CO_2$, which is on the high side compared with most forecast scenarios reported in McFarland et al. (2018). The curtailment cost is assumed to be $5,000/MWh.

The hourly solar capacity factor profiles including photo-voltaic (pv) and concentrated solar power (csp), are calculated based on the national solar radiation data base (NSRDB) (Sengupta et al., 2018) in 2012 via the System Advisor Model (SAM) (Blair et al., 2014). The hourly wind capacity factor profiles are calculated based on the wind speed from the wind integration national dataset (wind) toolkit (Draxl et al., 2015) in 2012 using one power curve from SAM. Since load data are correlated with solar and wind capacity factors, to generate the hourly load profiles we take load data from ERCOT in 2012 and scale them so that the annual load for each ERCOT region is equal to the annual load in 2019. To sum

up, we have 365 days (the leap day is excluded) of 24 hour solar and wind capacity factor and load data. The capacity factors are assumed to be unchanged over the planning horizon. The annual load growth rate is assumed to be 1.4% calculated based on the historical load data from 2011 to 2018 (ERCOT, 2106). To select the representative days for the GTEP model, we use the software package, TimeSeriesClustering.jl developed by Teichgraeber and Brandt (Teichgraeber and Brandt, 2019). By using this package, we are able to apply the $k$-means clustering algorithm to the time series and select the centroid of each cluster as the representative day. The weight of each representative day is proportional to the number of data points in that cluster. The details of the clustering algorithms are described in Teichgraeber and Brandt (2019). There is a trade-off between computational complexity and model fidelity in selecting the number of representative days. Here, we adopt a trial-and-error approach and gradually increase the number of representative days until the investment decisions do not change significantly. Due to the length constraint of the paper, the sensitivity analysis with respect to the number of representative days is given in Appendix D of the supplementary material. The results with 15 representative days are reported in subsection 8.4.3 because the solutions stabilize after the number of representative days is increased to 15.

### 8.4.2 Comparison of formulations and algorithms

All the MILP formulations are implemented in Pyomo/Python (Hart et al., 2011). We first solve the GTEP model directly with CPLEX 12.9.0.0. We compare the three transmission expansion formulations proposed in subsection 8.2.1. All the problems in this chapter are solved using one processor of an Intel Xeon (2.67GHz) machine with 64 GB RAM. The time limit is set to 10 hours. The number of general integer variables, binary variables, continuous variables, and constraints of the fullspace GTEP problem with the three different formulations are given in Table 8.1. The upper bound (UB), lower bound (LB) of the optimal value of the objective function in billion dollars and the wall time in seconds are also shown in Table 8.1. All the three formulations have the same number of general

integer variables and binary variables but differ in the number of continuous variables. The standard big-M formulation uses the fewest number of continuous variables and constraints. CPLEX is not able to find a feasible solution (UB) for any of the three formulations within the prespecified time limit. The lower bound column (LB) provides the bound that CPLEX returns at termination. In fact, we direct CPLEX to solve the LP relaxation for each of the three formulations, but CPLEX was not able to return a solution for any of the formulations within the 10-hour time limit, regardless of the LP algorithm chosen.

Table 8.1: Computational statistics for the fullspace problem with 4 representative days

| Formulation | Int Var | Bin Var | Cont Var | Constraints | UB ($\$10^9$) | LB ($\$10^9$) | Wall time (sec) |
|---|---|---|---|---|---|---|---|
| big-M | 274,920 | 2,800 | 564,826 | 1,543,966 | - | 21.13 | 36,000 |
| alternative big-M | 274,920 | 2,800 | 1,102,426 | 2,081,566 | - | 21.13 | 36,000 |
| hull | 274,920 | 2,800 | 833,626 | 2,081,566 | - | 281.73 | 36,000 |

We also test the two decomposition algorithms described in section 8.3. The nested Benders decomposition is implemented in Pyomo/Python (Hart et al., 2011). The tailored Benders decomposition implementation is from CPLEX (Bonami et al., 2020b), which is called using the Pyomo persistent solver interface (Siirola, 2017). The computational results of the two proposed decomposition algorithms are shown in Table 8.2.

The tailored Benders decomposition algorithm is able to solve all the three formulations to within 1% optimality gap within 10,000 seconds.

For the nested Benders decomposition, we observe that the forward pass with integrality constraints is expensive to solve. Therefore, we make a change in the implementation so that we first use the nested Benders decomposition algorithm to solve the LP relaxation of the problem until the LP relaxation is solved to optimality or we reach the time limit of 10 hours or the iteration limit of 100. Then we perform one single forward pass with the integrality constraints to obtain a feasible solution. Although the nested Benders decomposition can obtain an upper bound and a lower bound to all the three formulations, the optimality gaps are large compared to the results from the tailored Benders decomposition. In fact, in none of the three formulations is the nested Benders decomposition able to solve the LP relaxation of the problems to optimality within the time limit. Note that the performance of

the nested Benders decomposition is quite different from the numerical experiments on the GEP model performed by Lara et al. (2018) for the GEP model where the nested Benders decomposition performs well. The reason for this difference could be due to the complication brought by the transmission expansion constraints and the DC power flow equations of the GTEP model. As a result, the subproblems become larger and more dual degenerate, which makes nested Benders decomposition take not only more time to solve each iteration but also more iterations to converge.

Table 8.2: Computational results of the two proposed decomposition algorithms using different formulations

| Algorithm | Formulation | UB ($\$10^9$) | LB ($\$10^9$) | Gap | Wall time (secs) |
|---|---|---|---|---|---|
| tailored Benders | big-M | 283.7 | 282.6 | 0.38% | 5,115 |
| tailored Benders | alternative big-M | 283.9 | 281.6 | 0.82% | 3,693 |
| tailored Benders | hull | 282.6 | 280.6 | 0.71% | 8,418 |
| nested Benders | big-M | 295.7 | 268.9 | 9.98% | 53,682 |
| nested Benders | alternative big-M | 294.2 | 265.5 | 10.81% | 43,389 |
| nested Benders | hull | 288.0 | 269.3 | 6.97% | 37,577 |

From this numerical experiment, the tailored Benders decomposition algorithm with the alternative big-M formulation proves to be the fastest. We adopt this algorithm-formulation combination for the rest of the experiments in this chapter.

Some additional computational statistics on this problem with 4 representative days including the problem sizes after presolve, the performance curves for the nested Benders decomposition, the sizes of the Benders master problem and subproblem are shown in Appendix C of the supplementary material.

### 8.4.3 Results with 15 representative days

We improve the fidelity of the model by increasing the number of representative days to 15. The 15 representative day model with the alternative big-M formulation has 2,800 binary variables, 1,024,680 general integer variables, 4,120,606 continuous variables, and 7,787,266 constraints. The proposed tailored Benders decomposition algorithm is able to solve the

problem in 33,207 seconds with an upper bound of 301.1 ($\$10^9$), a lower bound of 299.9 ($\$10^9$) and an optimality gap of 0.4%.

The capacities of different generation technologies from 2019 to 2038 are shown in Figure 8.3. The results include high capacities of solar and wind. The aggregated natural gas capacity of the five regions increases in the first few years, reaches its peak in 2024 and gradually decreases afterwards due to the retirement of old generators and the increase in carbon tax, which makes the natural gas generators less competitive compared with solar and wind generators. The nuclear capacities are unchanged throughout the planning horizon. The coal capacities are unchanged in the first few years and start decreasing in 2029 because of reaching their nominal lifetimes. No storage unit is installed. Therefore, the renewable generation when the net load is negative has to be curtailed. The total discounted renewable curtailment cost is 1.64 billion in 20 years.

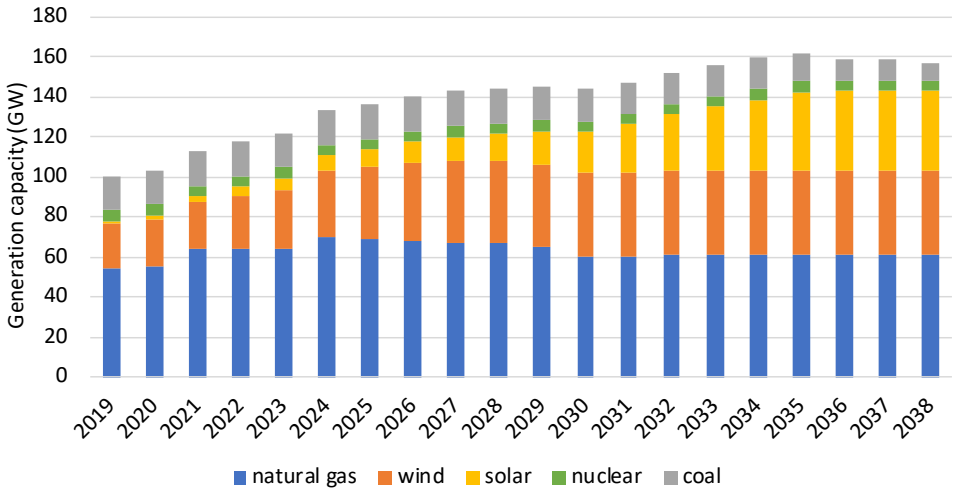


Figure 8.3: Aggregated generation expansion results

Geographically, most of the solar and wind capacity additions are projected to take place in the West and Panhandle regions because the capacity factors for solar and wind are higher in these two regions. The projected capacity for natural gas in the four regions, i.e., Coast, Northeast, South, and West, are shown in Figure 8.4. It can be seen that most natural gas expansions are expected to take place in the Northeast and Coast regions where the absolute increase in load is high and capacity factors for renewables are relatively low.

In the West region, where the absolute load increase is low and the capacity factors for renewable generation are high, we observe very marginal changes in natural gas capacity. In the South region, natural gas capacity increases in the first few years and reaches a peak in 2025. After 2025, natural gas capacity decreases over the years due to the retirement of old generators. The load growth in South is satisfied by power transfers from West region, which we analyze below.

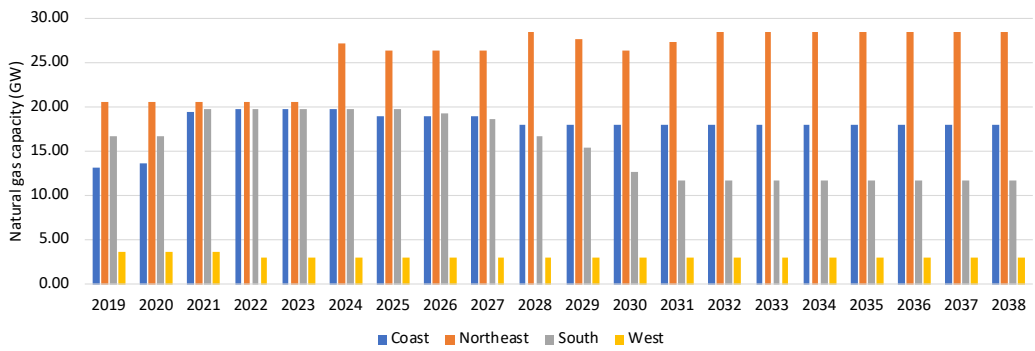


Figure 8.4: Projected capacities for natural gas in Coast, Northeast, South, and West

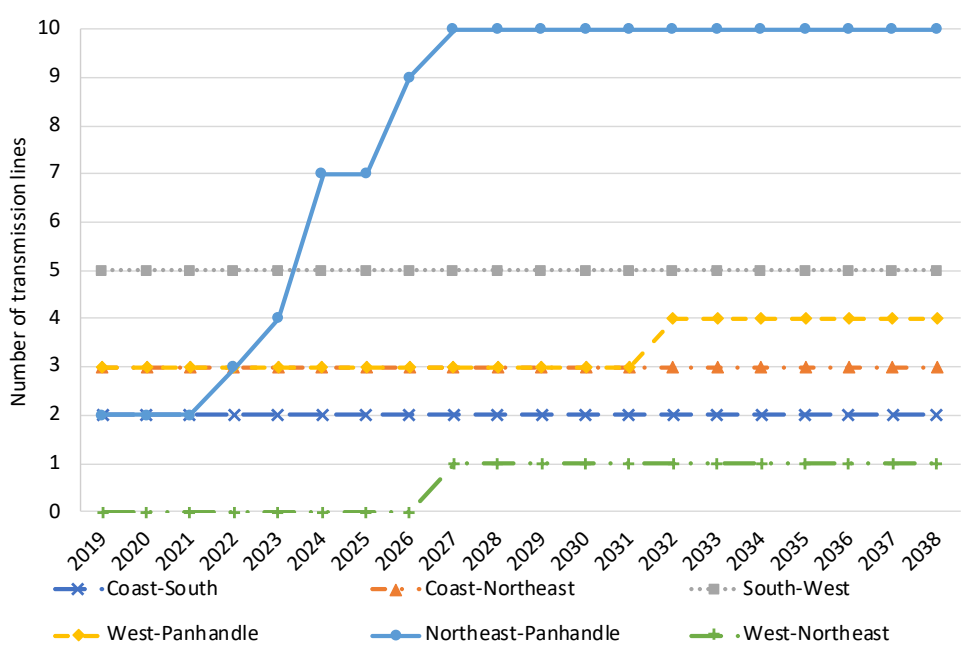


Figure 8.5: Transmission expansion results

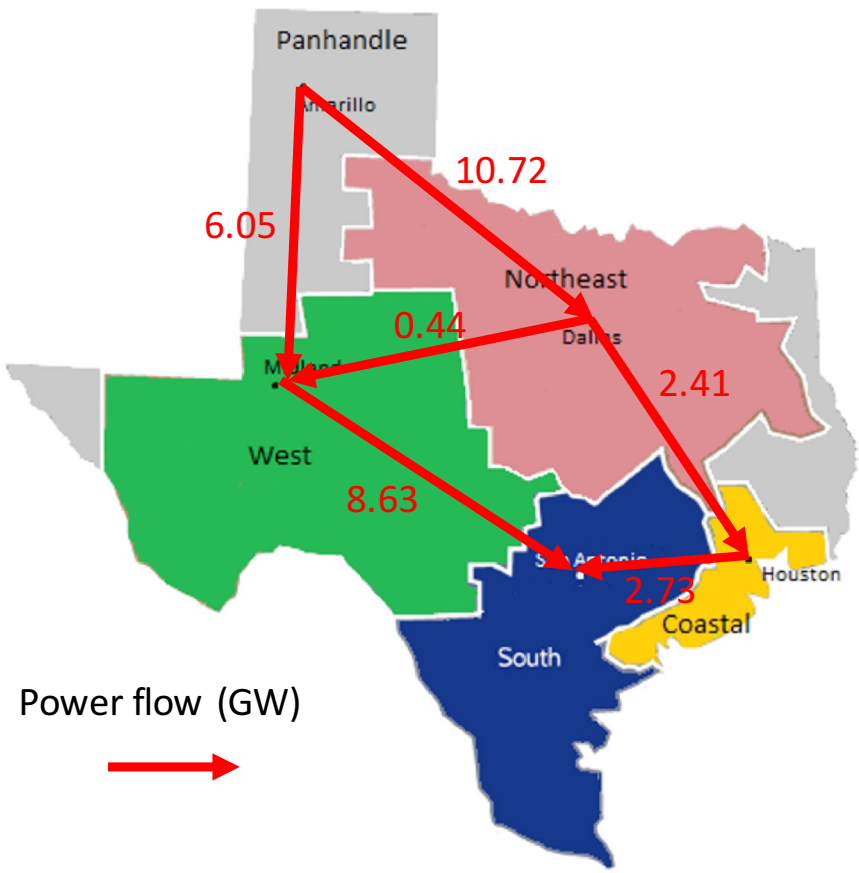


Figure 8.6: Aggregated power flow directions and magnitudes for all the transmission lines at $t = 20$, $d = 15$, $s = 24$

### 8.4.4 Sensitivity analysis of input generator data

The number of transmission lines built over the planning horizon are shown in Figure 8.5. Most of the transmission lines are built for Northeast-Panhandle and South-West in order to transfer the power generated by the renewable sources in West and Panhandle to other regions. Note that we assume that no transmission lines are built *a priori*. It is clear that there are correlations between the geographical locations of the generation technologies and the transmission expansion decisions.

Figure 8.6 shows the aggregated power flow through all the installed transmission lines at a peak load time period ($t = 20$, $d = 15$, $s = 24$). The directions and the magnitudes of

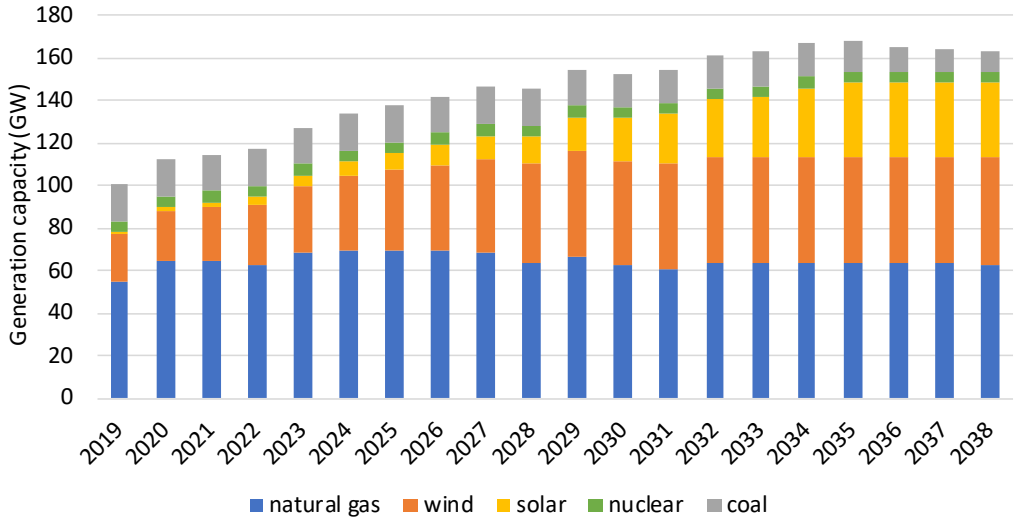


Figure 8.7: Generation expansion results using generating unit cost data from IHS Markit

the power flows are represented by the red arrows and numbers in this Figure. The most significant power flows are from Panhandle to Northeast and from West to South due the surplus of their renewable energy generation.

The generation capacity expansion planning results can be sensitive to the forecast of future capital cost and operating cost of the potential generating units. Recall that generator cost data for the previous analysis are from the National Renewable Energy Laboratory (NREL), available in the 2016 Annual Technology Baseline (ATB) Spreadsheet (Cole et al., 2016). We switch the capital cost and operating cost data for all the generators to internal data from IHS Markit and re-run the Benders decomposition experiment with 15 representative days. The proposed tailored Benders decomposition algorithm is able to solve the problem in 51,445 seconds with an upper bound of 294.4 ($\$10^9$), a lower bound of 293.7 ($\$10^9$) and an optimality gap of 0.2%. The increase in computational time compared with using NREL data is 67%.

The generation expansion results are shown in Figure 8.7. The major change is that the IHS Markit results favors more wind and less solar compared with the results shown in Figure 8.3. To give some insight on why this happens, we find that the overnight cost of the wind generating units in the IHS Markit dataset is 25% lower than that in the NREL

dataset in the first year of the planning horizon. In contrast, the overnight cost for the PV generating units is only 11% lower in the first year.

## 8.5 Conclusions

In this chapter, we address generation and transmission expansion planning (GTEP) problem by developing MILP formulations and solution techniques. We consider both thermal and renewable technologies as expansion candidates. Operating and transmission constraints are included in the model, which leads to large scale problems. To limit the size of the GTEP model, several simplifications are made. We aggregate the generators that use the same technology assuming that they have the same design parameters. We also spatially aggregate regions with similar climate and load profiles. For example, the ERCOT is divided into five regions. Each region represents one bus in the power flow model. Therefore, we only consider the expansion of tielines between regions. In terms of temporal representation, we select some representative days with hourly load and capacity factor data. The representative days are selected by a clustering algorithm such as $k$-means clustering.

The model is a multi-scale MILP model with both investment decisions and operating decisions. We compare three different formulations for transmission expansion, i.e., the big-M formulation, the hull formulation and the alternative big-M formulation. We prove that the alternative big-M (ABM) formulation has the same feasible region as the big-M formulation (BM) when projected onto the space of the variables involved in the big-M formulation. Computational experiments are performed as well for the three formulations, but it is hard to identify a clear winner among the three formulations.

Two solution techniques, a nested Benders decomposition algorithm and a tailored Benders decomposition algorithm, are proposed. Both algorithms decompose the planning problem by year. The nested Benders decomposition solves each year sequentially in a forward and backward pass manner. The Benders decomposition defines a master problem that deals with the investment decisions and a number of subproblems corresponding to representative operating decisions for a given year. The tailored Benders decomposition algorithm

outperforms the nested Benders decomposition one in our computational experiments.

An ERCOT case study is used to demonstrate the GTEP model and the solution techniques. The tailored Benders decomposition is able to solve the 20 year planning problem with 15 representative days. The capacity expansion mix for ERCOT will mainly include solar and wind capacities in the West and Panhandle regions. The transmission lines are mainly built to transfer power from solar and wind rich regions to the South and Northeast regions of ERCOT, which shows that the generation and transmission decisions are correlated. Co-optimization of generation and transmission has the potential of bring additional value to the system operator/regulator than solving the two planning problems independently.

Several future directions can be pursued. First, the deterministic GTEP model can be extended to a multi-stage stochastic programming framework by considering uncertainties in load growth rate, carbon tax, etc. Second, the benefit of using DC power flow compared with the network flow model is limited in a small scale problem. It is worth testing the model in a problem with larger number of nodes to test the scalibility of the proposed approach. Third, the computational performance of both decomposition algorithms can be improved using warm-start techniques where a number of cuts can be generated at the beginning based on the solutions obtained from some heuristics. Fourth, the reliability of the system can be improved by adding contingency constraints.

# Chapter 9

# On Representative Day Selection for Capacity Expansion Planning of Power Systems under Extreme Events

## 9.1 Discussion of issues in representative day selection

Capacity expansion planning (CEP) models (Conejo et al., 2016a; Koltsaklis and Dagoumas, 2018; Hemmati et al., 2013) that have been extensively used by power system operators and planners aim to determine the location, size, and type of the generating units and/or transmission lines that should be installed in order to meet the electricity demand within a given geographical region. These investment decisions in generating units and transmission lines are long-term decisions usually made on a yearly basis. However, due to the increase in the penetration of generation for renewable resources, CEP models developed recently (Lara et al., 2018; Palmintier and Webster, 2011) incorporate the hourly operating decisions in order to capture the high variations of renewable generation. Operating decisions including unit commitment and ramping decisions, and economic dispatch need to be included in CEP models, which leads to large-scale mixed-integer linear programming (MILP) problems.

One of the challenges in CEP models that include operating decisions is the temporal

complexity. Modeling each hour in a 10-30 year planning horizon will lead to MILP problems with billions of variables, which is intractable with the current commercial solvers. Different simplifications have been proposed to make the CEP models tractable. Mallapragada et al. (2018) proposes a time-slice model where the authors average the load and the capacity factor data of each of the four seasons (spring, summer, fall, winter) into time slices representing morning (7 am-2 pm), afternoon (2-6 pm), evening (6-11 pm), and night (11 pm-7 am). This approach does not link consecutive periods and thus fails to characterize the chronology of the operating decisions. In most of the literature, a "representative days" approach is used (Almaimouni et al., 2018; Kotzur et al., 2018; Bahl et al., 2018; García-Cerezo et al., 2020; Liu et al., 2017a; Mallapragada et al., 2018; Nahmmacher et al., 2016; Pfenninger, 2017; Scott et al., 2019; Seljom and Tomasgard, 2019; De Sisternes and Webster, 2013; Sun et al., 2019; Teichgraeber and Brandt, 2019; Teichgraeber et al., 2020; Yeganefar et al., 2020; Tso et al., 2020). A dataset is given that consists of the historical load data and capacity factor data for solar and wind generators. The historical data is then scaled to account for load growth. Based on this scaled dataset, $k$ representative days are selected in each year of the planning problem to represent the whole planning horizon where $k \ll 365$. A clustering algorithm, such as $k$-means clustering or $k$-medoids clustering, is used to divide the whole historical dataset into $k$ clusters based on some of the characteristics of the historical days. The centroid or the medoid of each cluster is selected as the representative day. Several issues could arise in the representative day selection procedure. We summarize these issues below and discuss how the existing literature addresses them.

I *What type of data should be used for clustering?*

Most papers perform clustering based on the input parameters to the CEP models (Almaimouni et al., 2018; Kotzur et al., 2018; Bahl et al., 2018; García-Cerezo et al., 2020; Liu et al., 2017a; Mallapragada et al., 2018; Nahmmacher et al., 2016; Pfenninger, 2017; Scott et al., 2019; Seljom and Tomasgard, 2019; De Sisternes and Webster, 2013; Teichgraeber and Brandt, 2019; Teichgraeber et al., 2020; Yeganefar et al., 2020; Tso et al., 2020), i.e., the load data and capacity factors corresponding to each historical day.

More specifically, each input data point to the clustering algorithm is a concatenation of the load and the capacity factor time series corresponding to a given historical day. As a result, the temporal correlations of different time series and the hourly chronology of each time series can be preserved. Since the numerical values of the load and the capacity factors can vary significantly, several normalization approaches have been proposed (Teichgraeber and Brandt, 2019). Besides clustering based on input parameters, Pineda and Conejo (2010); Sun et al. (2019) propose to perform clustering based on the optimal objective value or the optimal solution of each historical day. In these approaches, a CEP model has to be solved to optimality for each historical day. The case study in Sun et al. (2019) shows that this cost-based approach has generally a superior performance than the input-based approach.

II *Which clustering algorithm should be used?*

Different clustering algorithms have been used including $k$-means clustering (Almaimouni et al., 2018; Kotzur et al., 2018; García-Cerezo et al., 2020; Pfenninger, 2017; Teichgraeber and Brandt, 2019; Teichgraeber et al., 2020; Liu et al., 2017a; Mallapragada et al., 2018; Scott et al., 2019), $k$-medoids clustering (Kotzur et al., 2018; Teichgraeber and Brandt, 2019; Bahl et al., 2018; Scott et al., 2019), hierarchical clustering Kotzur et al. (2018); Nahmmacher et al. (2016); Pfenninger (2017); Sun et al. (2019); Teichgraeber and Brandt (2019); Liu et al. (2017a), DBA clustering (Teichgraeber and Brandt, 2019), $k$-shape clustering (Teichgraeber and Brandt, 2019), self-organizing map (SOM) clustering (Yeganefar et al., 2020). Most of the papers that we surveyed conclude that there is no clear winner among all the different clustering algorithms (Kotzur et al., 2018; Teichgraeber and Brandt, 2019).

III *How to choose the appropriate number of clusters?*

There is no standard method to choose the number of clusters *a priori*. Kotzur et al. (2018); Teichgraeber and Brandt (2019); Mallapragada et al. (2018); Sun et al. (2019); García-Cerezo et al. (2020); Nahmmacher et al. (2016) adopt a trial-and-error approach

to choose the appropriate number of representative days by gradually increasing the number of representative days and observe if the optimal objective value and the optimal solution stabilize. Other methods are solely based on the metrics on the input time series without solving any optimization problem. For example, Almaimouni et al. (2018) use an "elbow method" that determines how well the $k$ clusters can characterize the variance of the input data. Poncelet et al. (2016); Yeganefar et al. (2020) use metrics like average normalized root mean square error (NRMSE) for the time series data.

IV *Should additional "extreme days" be included and how to find these "extreme days"?*

Including $k$ representative days in the CEP model alone may not be enough since the representative days are the centroids or the medoids of the clusters, and therefore, fail to capture extreme events such as the day with the peak load. To guarantee the feasibility of the investment decisions, several works have been done on "extreme days" selection. Most of these works select extreme days based on the values of the input data (Kotzur et al., 2018; Scott et al., 2019; Pfenninger, 2017; Yeganefar et al., 2020). Scott et al. (2019); Pfenninger (2017); Yeganefar et al. (2020) propose to select extreme days that have extreme values in the input data such as the days with the peak load, the peak net load, and/or the peak ramp-up and append these extreme days as additional representative days. Kotzur et al. (2018); García-Cerezo et al. (2020) propose to modify the clustering process to include the extreme values of the clusters. Besides identifying extreme days based on input data, Teichgraeber et al. (2020) proposes to select extreme days based on the slack variables of the optimization problem itself.

V *How to estimate a bound for the error of the considered approach?*

A CEP model with representative days can be seen as a surrogate of the CEP model with all the days. It is relevant to know how far away the optimal solution of the surrogate is from the fullspace CEP model. Sun et al. (2019) assume that the "ground truth model", i.e., the model with a sufficiently large number of days, is solvable and

compare the investment decisions obtained from the representative day model with the solution of the "ground truth model". Seljom and Tomasgard (2019) proposes to use a sample average approximation approach (Shapiro et al., 2014) to calculate a statistical lower bound for the fullspace model. Teichgraeber and Brandt (2019) proves that for some linear programming energy system problem, the surrogate problem is a relaxation of the fullspace problem under some assumptions. However, the properties proved in Teichgraeber and Brandt (2019) are restricted to LP problems with uncertain data in the objective and right hand side coefficients, which does not apply to a general CEP model.

this chapter aims to provide additional developments on the five issues discussed above. The works that are closest to this chapter are Teichgraeber and Brandt (2019); Sun et al. (2019); Teichgraeber et al. (2020). The contributions of this chapter are outlined below.

- The procedures of the input-based approach and the cost-based approach are presented with extensions to address general CEP problems.
- The theoretical properties of the cost-based approach and the input-based approach are analyzed including a method to estimate the "optimality gap" of the representative day approach with respect to the full day approach. Several relationships between the clustering error and the "optimality gap" are provided.
- Two extreme day selection methods are proposed.
- A case study is used to compare the effectiveness of different algorithms. The effects of adding extreme days are also analyzed.

## 9.2 Background: clustering algorithms

As discussed in the previous section, different clustering algorithms have been used for representative day selection. There is no clear winner among all the clustering algorithms reported in the literature (Kotzur et al., 2018; Teichgraeber and Brandt, 2019). In this

chapter, $k$-means clustering and $k$ medoids clustering are considered. In order to make this chapter self-contained, the necessary background for these two clustering algorithms is provided in this section.

### 9.2.1 $k$-means clustering

Given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a $D$-dimensional real vector, $k$-means clustering aims to partition the $n$ observations into $k$ ($\leq n$) sets so as to minimize the within-cluster variances. Suppose $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$, denotes the partition of set $\{1, \ldots, n\}$ into $k$ subsets, $k$-means clustering is to solve the following optimization problem to find the optimal partition $\mathbf{S}^*$,

$$\mathbf{S}^* = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{9.1}$$

where $\mu_i$ is the mean of the points in $S_i$. The $k$-means clustering is an NP-hard problem. One approach to solve it is to use heuristics, such as the Lloyd–Forgy algorithm (Forgy, 1965). Implementation of this heuristic method is available in the well-known Python package scikit-learn (Pedregosa et al., 2011).

Although the heuristics can provide good feasible solutions to Problem (9.1), they cannot guarantee that the global optimal solution is found. An alternative approach to solve (9.1) is to formulate the problem as a mixed-integer nonlinear program (MINLP) shown in Problem (9.2). Binary variable $y_{il}$ represents whether the $i$th data point $x_i$ belongs to the $l$th cluster for all $i \in \{1, \ldots, n\}, l \in \{1, \ldots, k\}$. Continuous variable $c_{lj}$ denotes $j$th coordinate of the center of the $l$th cluster. Continuous variable $d_i$ denotes the square of the Euclidean distance from the point $x_i$ to the center of the cluster it belongs to. In Equation (9.2b), $d_i$ is greater than or equal to the sum of the squared distances of each coordinate if point $i$ belongs to cluster $l$, i.e., $y_{il} = 1$. The parameter $M_i$ is a big-M parameter such that the inequality holds when $y_{il} = 0$. Equation (9.2c) forces each point $i$ to be assigned to one of the clusters.

$$\min_{\mathbf{c},\mathbf{d},\mathbf{y}} \sum_{i=1}^{n} d_i \tag{9.2a}$$

$$d_i \geq \Big(\sum_{j=1}^{D}(x_{ij} - c_{lj})^2\Big) - M_i(1 - y_{il}) \quad \forall i \in \{1, \ldots, n\}, l \in \{1, \ldots, k\} \tag{9.2b}$$

$$\sum_{l=1}^{k} y_{il} = 1 \quad \forall i \in \{1, \ldots, n\} \tag{9.2c}$$

$$\mathbf{c}_l \in \mathbb{R}^D \quad \forall l \in \{1, \ldots, k\} \tag{9.2d}$$

$$d_i \in \mathbb{R}_+ \quad \forall i \in \{1, \ldots, n\} \tag{9.2e}$$

$$y_{il} \in \{0, 1\} \quad \forall i \in \{1, \ldots, n\}, l \in \{1, \ldots, k\} \tag{9.2f}$$

It is easy to observe that the size of (9.2) increases with the increase in the dimension of the data $D$, the number of data points $n$, and the number of clusters $k$. As a result, Problem (9.1) is difficult to solve for real-world CEP problems. In fact, we find that problem (9.2) is not solvable to global optimality for our case study. Therefore, the heuristic algorithm implemented in scikit-learn is used for $k$-means.

### 9.2.2 $k$-medoids clustering

The $k$-medoids problem is a clustering problem similar to $k$-means. Instead of minimizing the sum of within-cluster variances, $k$-medoids clustering seeks to minimize the distance from the "medoids" of each cluster. The difference between the medoid and the mean of the cluster is that the medoid has to be one of the actual data points within that cluster. In general, the $k$-medoids clustering problem is also NP-hard. There are heuristic algorithms for the $k$-medoids clustering (Park and Jun, 2009) whose implementations are available in scikit-learn.

Alternatively, $k$-medoids can be solved to global optimality using mixed-integer linear programming (MILP). The MILP formulation is described in Problem (9.3). Binary variable $y_i$ represents whether the $i$th data point is a medoid of a cluster ($y_i$=1) or not ($y_i$=0). Binary variable $z_{ij}$ denotes whether the $i$th data point belongs to the cluster with the $j$th data point as its medoid. The objective is to minimize the sum of the distances from each point to its

medoid. It is relevant to note that the distance between any two points $i$ and $j$ can be calculated *a priori*. Equation (9.3b) denotes that each point has to be assigned to exactly one point that is the medoid of the cluster it belongs to. Equation (9.3c) forces $z_{ij}$ be zero if point $j$ is not a medoid. Equation (9.3d) specify that the number of medoids is equal to $k$.

$$\min_{\mathbf{z},\mathbf{y}} \sum_{ij} d_{ij} z_{ij} \tag{9.3a}$$

$$\sum_{j=1}^{n} z_{ij} = 1 \quad \forall i = 1, 2, \ldots, n \tag{9.3b}$$

$$z_{ij} \leq y_j \quad \forall i = 1, 2, \ldots, n, j = 1, 2, \ldots, n \tag{9.3c}$$

$$\sum_{i=1}^{n} y_i = k \tag{9.3d}$$

The number variables in (9.3) increases with the number of data points, but is independent of the number of clusters and the dimension of the data. We have found that (9.3) can be solved efficiently in CEP applications.

## 9.3 Representative day selection algorithm

To guarantee the fidelity of the CEP model, an accurate forecast of the hourly load and capacity factors of renewable generating units is needed. We assume that some historical load and capacity factor data are available and that those data are sufficient to characterize the possible capacity factor and load variations. The historical loads are scaled to consider the future load growth, i.e., some annual load growth rate is assumed. For the capacity factor data, no scaling is needed since we assume that the weather condition change is negligible across years. We further assume that the inter-day (not the intra-day) ramping constraints can be neglected, i.e., there is no linking constraints between two consecutive days. While this assumption may decrease the fidelity of the model, it is necessary for our representative day approach. One can always increase the length of the time block, e.g., using representative weeks (Bahl et al., 2018) instead of representative days.

With these assumptions, a fullspace CEP problem with all the historical data used to represent each year of the planning horizon is defined in (9.4). Set $\mathcal{D}$ represents the set of days in the historical dataset. Each day has a weight of $\frac{365}{|\mathcal{D}|}$ in the objective function. Equation (9.4) is a succinct representation of the CEP problem. A detailed MILP formulation can be found in Li et al. (2020b).

$$(FD) \quad OBJ_{FD} = \min \sum_{t \in \mathcal{T}} \left( c_t^\top x_t + \sum_{d \in \mathcal{D}} \frac{365}{|\mathcal{D}|} f_t^\top y_{t,d} \right) \tag{9.4a}$$

$$\text{s.t.} \quad A_{t,d} x_t + B_t y_{t,d} \leq b_{t,d} \quad \forall t \in \mathcal{T}, d \in \mathcal{D} \tag{9.4b}$$

$$C_{t-1} x_{t-1} + D_t x_t \leq g_t \quad t = 2, 3, \ldots, |\mathcal{T}| \tag{9.4c}$$

$$x_t \in X_t, \quad \forall t \in \mathcal{T}, \quad y_{t,d} \in Y_t, \quad \forall t \in \mathcal{T}, d \in \mathcal{D} \tag{9.4d}$$

Variable $x_t$ represents the investment decisions at year $t$. Variable $y_{t,d}$ represents the operating decisions corresponding to day $d$ in year $t$. The objective (9.4a) is to minimize the total cost. Equation (9.4b) describes the operational decisions of each year $t$ and each day $d$, such as power flow equations, unit commitment, and economic dispatch. Equations (9.4c) are investment-related constraints, such as the installation and retirement of generating units. Equations (9.4d) specify the domain of the variables. We note that the days in our dataset differ in the load and the capacity factors of the renewable generators. The parameters corresponding to the load appear on the right hand side of equation (9.4b) represented by $b_{t,d}$. The parameters corresponding to the capacity factors are part of the constraint matrices $A_{t,d}$. All the other parameters in the model, including $c_t$, $f_t$, $B_t$, $C_t$, $D_t$, and $g_t$, are only indexed by year $t$ because they only change on a yearly basis.

Since problem (FD) includes all the historical data, it is best to solve (FD) directly to obtain planning decisions that are feasible for all the days in dataset $\mathcal{D}$. However, solving (FD) directly is prohibitive in practice since the number of variables in (FD) can easily exceed one billion (Li et al., 2020b) if we consider one or more years of historical data. Therefore, the model (RD) below is solved as a surrogate of the fullspace model where a set of representative days $\mathcal{K}$ is selected or constructed to approximate problem (FD). The

number of representative days is denoted by the cardinality of set $\mathcal{K}$, i.e., $|\mathcal{K}|$. The weight of the $k$th representative day is denoted by $w_k$. Variable $y_{t,k}$ represents the operating decisions corresponding to the $k$th representative day in year $t$. Set $\tilde{Y}_t$ represents the LP relaxation of set $Y_t$, i.e., set $\tilde{Y}_t$ is obtained from set $Y_t$ if all the integrality constraints regarding the $y_{t,k}$ variables are relaxed. The reason for relaxing the integrality constraints for $y_{t,k}$ is to make the model (RD) amenable to decomposition algorithms, such as the Benders decomposition in Li et al. (2020b) or the nested Benders decomposition in Lara et al. (2018). Since all the integer variables are general integer variables instead of binary variables, the relaxation provides a very tight bound (Li et al., 2020b).

$$(RD) \quad OBJ_{RD} = \min \sum_{t \in \mathcal{T}} \left( c_t^\top x_t + \sum_{k \in \mathcal{K}} w_k f_t^\top y_{t,k} \right) \tag{9.5a}$$

$$\text{s.t.} \quad A_{t,k} x_t + B_t y_{t,k} \leq b_{t,k} \quad \forall t \in \mathcal{T}, k \in \mathcal{K} \tag{9.5b}$$

$$C_{t-1} x_{t-1} + D_t x_t \leq g_t \quad t = 2, 3, \ldots, |\mathcal{T}| \tag{9.5c}$$

$$x_t \in X_t, \quad \forall t \in \mathcal{T}, \quad y_{t,k} \in \tilde{Y}_t, \quad \forall t \in \mathcal{T}, k \in \mathcal{K} \tag{9.5d}$$

The main challenge that we face is to find out how the representative days should be selected in order to approximate problem (FD) as well as possible. Before diving into the representative day selection approaches, we introduce a quantitative metric to evaluate the solution quality of (RD). Suppose that the optimal investment decision of (RD) is $\mathbf{x}^{RD}$, then the "actual cost" of $\mathbf{x}^{RD}$ can be obtained by fixing the investment decisions at $\mathbf{x}^{RD}$ and solving the problem corresponding to each day in the full dataset $\mathcal{D}$ individually, which is equivalent to fixing the $\mathbf{x}$ variables in (FD) and solving the rest of the fullspace problem. We denote this objective as $OBJ_{FD}(\mathbf{x}^{RD})$. It is easy to see that

$$OBJ_{FD}(\mathbf{x}^{RD}) \geq OBJ_{FD}(\mathbf{x}^{FD}) = OBJ_{FD} \tag{9.6}$$

where $\mathbf{x}^{FD}$ represents the optimal investment decisions obtained with (FD).

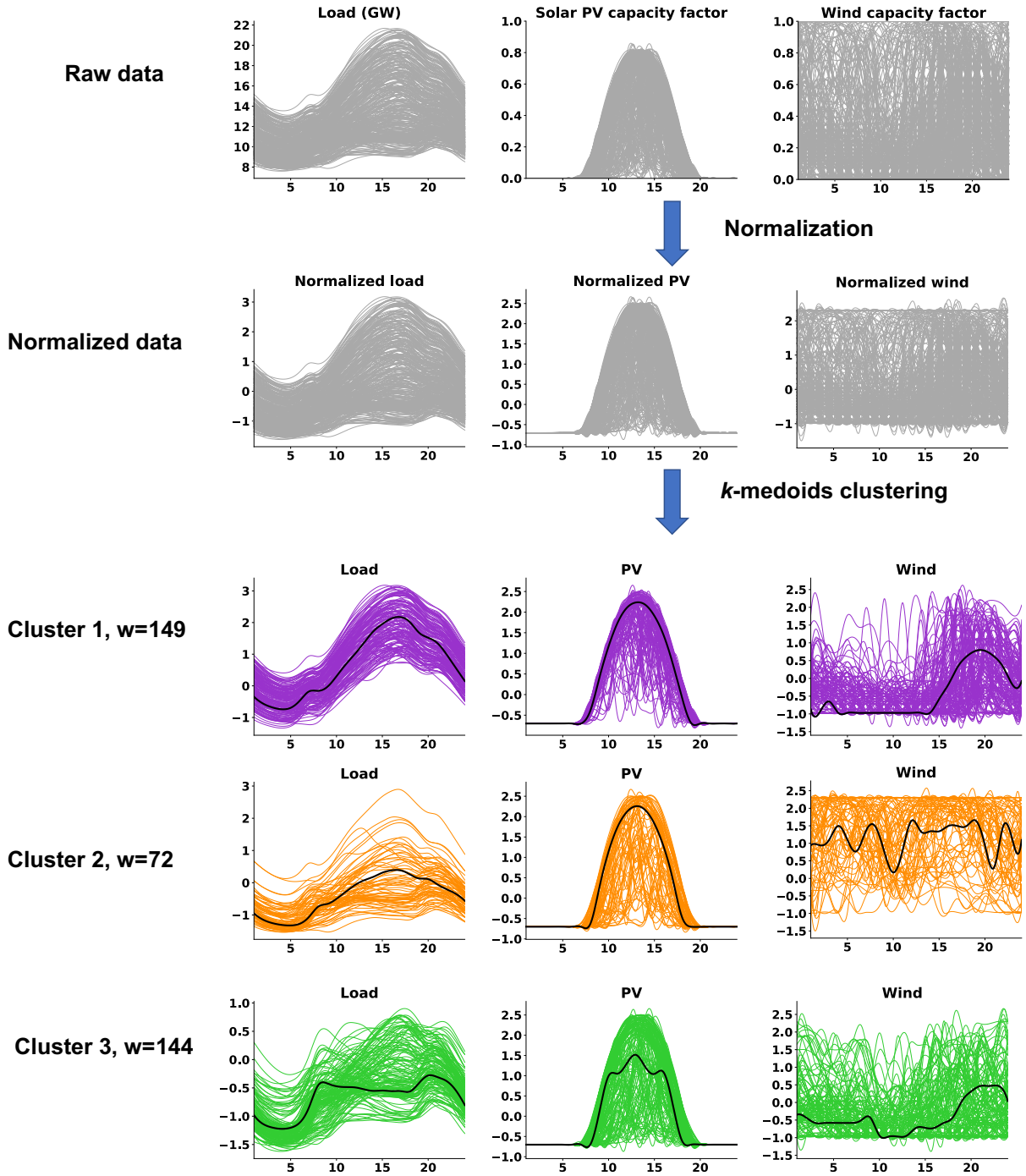


Figure 9.1: Illustration of the input-based approach

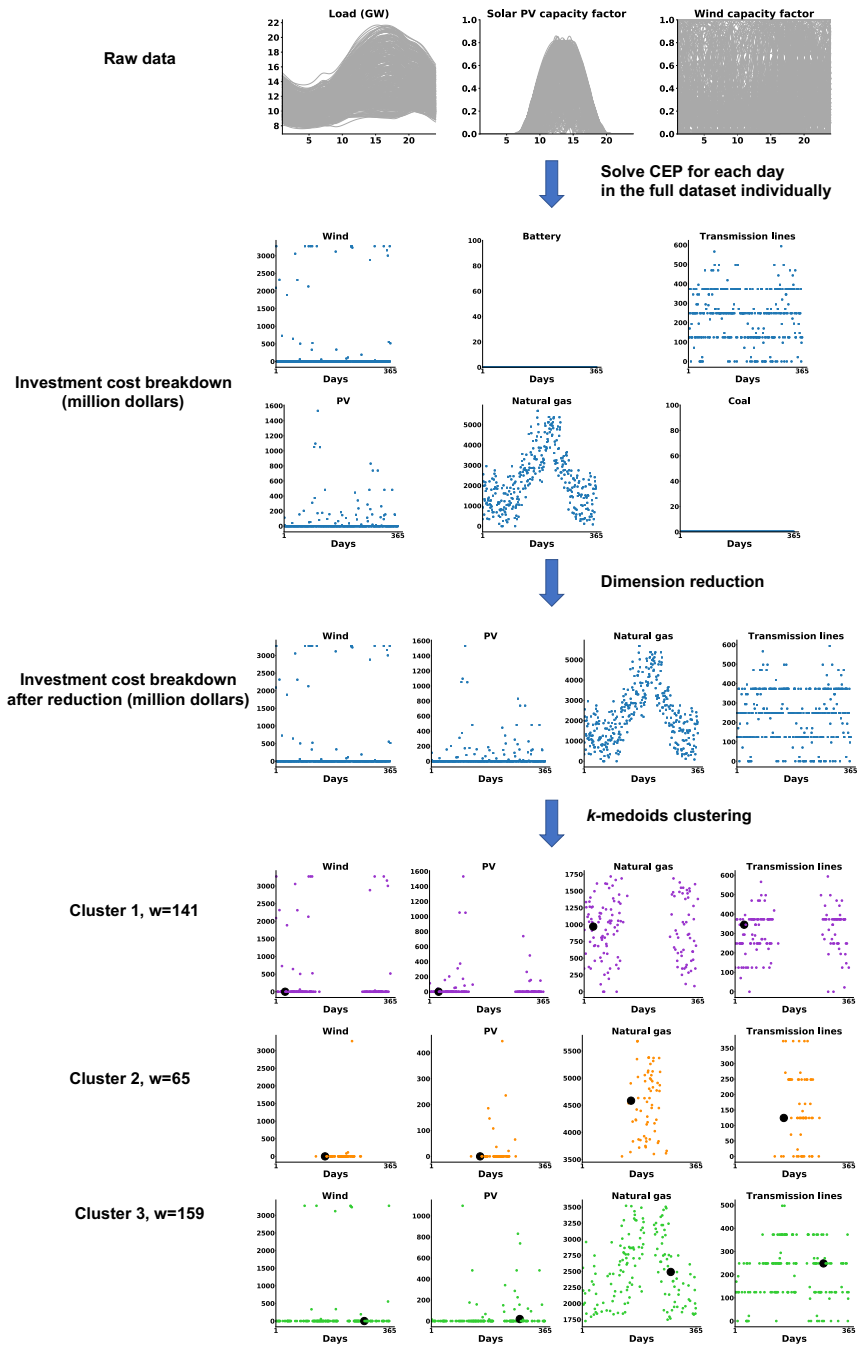


Figure 9.2: Illustration of the cost-based approach

### 9.3.1 Input-based approach

Most algorithms on representative day selection perform clustering directly on the input data to problem (FD). Let us first define the notations for this approach. Suppose we have

a dataset that contains the historical loads and the capacity factors of solar and wind for each node $n \in \mathcal{N}$ within the considered geographical region. The set of historical days in this dataset is represented by set $\mathcal{D}$. The data corresponding to day $d \in \mathcal{D}$ is represented by vector $H_d$. $H_d$ is a concatenation of the hourly time series data for the load, capacity factors of all the nodes $n \in \mathcal{N}$ in day $d$. This approach captures the correlations of the input data because the data corresponding to each day are concatenated. Before applying a clustering algorithm on $\{H_d, d \in \mathcal{D}\}$, the input data are usually normalized because they can be in very different magnitudes. For example, the capacity factors are from 0 to 1 while the load data are in GW. We normalize each type of data, e.g., the load data in a given node, using its mean and standard deviation over the whole dataset $\mathcal{D}$.

$$H_{d,p}^{\texttt{norm}} = \frac{H_{d,p} - \mu_p}{\sigma_p} \quad \forall p, d \tag{9.7}$$

where $H_{d,p}$ represents the entries in $H_d$ corresponding to the data type $p$. For example, $H_{d,p}$ can correspond to the time series for the load in a given node in day $d$. $\mu_p$ and $\sigma_p$ are scalars that represent the mean and standard deviation of $\{H_{d,p}, d \in \mathcal{D}\}$.

Other normalization schemes have been proposed in Teichgraeber and Brandt (2019), where instead of normalizing by the mean and standard deviation of each data type $p$ over the whole dataset, normalization can be performed based on the mean and variance of each day or each hour for a given data type $p$. In this chapter, we only apply the whole dataset normalization approach.

After the normalization, the clustering algorithms can be applied to $\{H_d^{\texttt{norm}}, d \in \mathcal{D}\}$. Then, the whole dataset is partitioned into $k$ clusters and the mean or the medoid of each cluster is selected as the input parameter corresponding to the representative day.

An illustrative example of the input-based approach is shown in Figure 9.1. The full dataset consists of 365 days of hourly load, hourly capacity factors of PV and wind corresponding to a single node, which is denoted as "raw data" in Figure 9.1. Each day is shown as a separate time series denoted by a gray line. After the normalization step, the time series become dimensionless and is shown as "normalized data". The $k$-medoids clustering is performed on the normalized data. For illustration, three clusters are shown in different colors

with weights, 149, 72, and 144, respectively. The medoid of each cluster is shown as a bold black line. The medoids will be used as the input data corresponding to the representative days.

### 9.3.2 Cost-based approach

Besides clustering the days based on input data, another approach is based on solving a CEP problem with operating decisions for only a single day $d$ in $\mathcal{D}$ at a time. The CEP problem with one single day is small and can be solved relatively fast. After solving the single-day CEP problem for each day $d \in \mathcal{D}$, the optimal investment decisions are obtained for all $d \in \mathcal{D}$. The hypothesis is that the days with similar optimal investment decisions, i.e., the days that need similar generators, transmission lines, and storage units, are similar and should be assigned to the same cluster. One option is to perform clustering based on the optimal solutions themselves. However, normalization is needed because the number of generators, transmission lines, and storage units are expressed in different magnitudes. One natural way to perform the normalization is to transform the number of units to the cost of different types of units as proposed in Sun et al. (2019) following the work reported in Pineda and Conejo (2010).

Our main contributions compared with Sun et al. (2019) in the cost-based approach are outlined below, which are discussed after an illustrative example is presented.

(i) We provide an approach that accommodate planning problems with a time horizon longer than one year.

(ii) Both $k$-means and $k$-medoids clustering are adapted to the cost-based approach, while Sun et al. (2019) only considers a $k$-medoids approach.

(iii) A simple dimension reduction method is proposed, and the effects of such method are analyzed combined with the two clustering algorithms.

To provide a conceptual overview of the proposed approach, an illustrative example is shown in Figure 9.2. The available raw data are hourly load, and hourly capacity factors of PV and

**Algorithm 1**

**Initialization:** A dataset $\mathcal{D}$ includes the load, and wind and solar capacity factor data. The data corresponding to the $d$th day in $\mathcal{D}$ is denoted as $H_d$.
**for** $d = 1$ to $|\mathcal{D}|$ **do**
  Solve a $T$ year CEP problem using $H_d$ as the single representative day data. Denote the optimal investment costs for each type of the generating and storage units in each node, and the transmission lines that connect any two nodes as a concatenated vector $c_d$. Denote the total cost of the CEP model as $tc_d$
**end for**
Considering matrix $C = [c_1, c_2, \ldots, c_{|\mathcal{D}|}]$, delete the all-zero rows in $C$ to derive matrix $\tilde{C}$.
Perform a $k$-clustering algorithm on the columns of matrix $\tilde{C}$. Select the representative days.

wind for 365 days. A CEP problem that has a single day is solved individually for each of the 365 days. The optimal investment costs in the wind, PV, natural gas, and coal generating units, batteries, and transmission lines are shown in the "investment cost breakdown" charts for each of the 365 days. Note that in practice, investment costs can be associated with each node of the CEP problem, i.e., each node has its associated costs in installing generating units and batteries. The transmission line costs are associated with any two nodes. For illustration purpose, we only show the investment costs of generating units and batteries corresponding to one selected node, and the transmission line costs connecting two selected nodes. Dimensional reduction is performed on the cost data by removing the units that are never installed in any of the 365 days. Specifically, the battery and the coal generators are never installed and therefore removed because they do not contribute to the clustering error. $k$-medoids clustering is performed on the investment costs breakdown after dimension reduction. Three clusters are obtained shown in different colors. The medoid of each cluster is highlighted using large black dots. The days corresponding to the medoids are selected as the representative days.

Next, we present our contributions regarding (i)-(iii). For (i), the proposed approach can be applied to solve a CEP model with $T$ years by considering the total discounted cost of each type of investment over the $T$ years. For (ii), it is not straightforward to apply $k$-

means clustering because the mean of the investment costs does not correspond to any days. However, it is possible to transform the data back to the input domain and take the mean of the input data for each cluster. For (iii), the dimension reduction technique that removes the all-zero entries can reduce the sizes of the MINLP problem for $k$-means clustering. On the other hand, the MILP formulation of the $k$-medoids clustering is independent of the dimension of the problem. Heuristic algorithms for clustering are generally computationally effective and do not suffer from the "curse of dimensionality". Since the MINLP formulation is too expensive to solve and will not be used in CEP problems, we conclude that the dimension reduction techniques have marginal effects on the clustering algorithms that are used.

The detailed steps of the cost-based algorithm are described in Algorithm 1.

### 9.3.3 Extreme days selection

As discussed in the introduction section, including only $|\mathcal{K}|$ representative days to solve the CEP model can lead to expansion decisions that are infeasible under some extreme days. To examine whether the optimal investment decisions found by the reduced problem (RD) is feasible for the fullspace problem (FD), the operating problem corresponding to each day $d \in \mathcal{D}$ is solved with the optimal investment decisions fixed evaluate the objective $OBJ_{FD}(\mathbf{x}^{RD})$. If there are no infeasible days in $\mathcal{D}$, i.e., $OBJ_{FD}(\mathbf{x}^{RD}) < +\infty$, the investment decisions found by solving (FD) are sufficient to satisfy the "extreme" scenarios. If not, suppose the set of infeasible days are represented by $\mathcal{I}$. In this case, we select the "extreme day" or "most infeasible day" in set $\mathcal{I}$ and add this extreme day to set $\mathcal{K}$ as a new cluster with only 1 day. Then, the weights of the representative days are adjusted accordingly. With the added extreme day, the reduced problem (RD) is solved again to find the new investment decisions. We repeat this procedure until all the days in the set $\mathcal{D}$ are feasible for our optimal investment decisions.

We next focus on how to choose the "extreme day" from the set of infeasible days $\mathcal{I}$. We propose two approaches to select the extreme days.

#### 9.3.3.1 Load shedding cost

The operating problems for the infeasible days are solved again with load shedding variables added to energy balance constraints for each node at each hour. The load shedding variables are used to balance the load when the power generation is not enough to satisfy the load. The objective function is changed to minimizing the total load shedding cost over the planning horizon. The day with the highest total load shedding cost is chosen as the extreme day.

This approach is similar to the one proposed in Teichgraeber et al. (2020). The difference is that Teichgraeber et al. (2020) proposes to choose the infeasible day with the highest load shedding cost for a single hour.

#### 9.3.3.2 Highest cost

Using the cost-based algorithm, the optimal total cost $tc_d$ (investment cost + operating cost) of each day in the dataset $\mathcal{D}$ is obtained by solving the optimization problem described in Algorithm 1. We choose the infeasible day with the highest total cost obtained in Algorithm 1 as the extreme day.

$$d^{\texttt{select}} = \arg\max_{d, d \in \mathcal{I}} tc_d \tag{9.8}$$

The rationale for this selection is that the day with the highest cost is likely to include the extreme events that trigger installing additional units and result in a high operating cost. The advantage of the highest cost approach is that it exploits information calculated already in Algorithm 1 and therefore there is no need to solve the infeasible CEP model again to find the extreme day.

### 9.3.4 Properties of the proposed algorithms

In this subsection, we analyze the proposed algorithms by characterizing their properties.

#### 9.3.4.1 Lower bound

An upper bound of the optimal objective value of the fullspace problem (FD) is available by evaluating $OBJ_{FD}(\mathbf{x}^{RD})$ as shown in Equation (9.6). Additionally, it is desirable to provide a lower bound of (FD) by solving (RD) to obtain an estimate of the suboptimality of the investment decisions obtained from (RD). The following theorem provides such a lower bound.

**Theorem 6.** For both cost-based and input-based approaches, if k-means clustering in (9.1) is used, (RD) provides a lower bound for the optimal objective value of (FD), i.e., $OBJ_{RD} \leq OBJ_{FD}$. This lower bound holds before and after adding extreme days.

*Proof.* The proof is provided in Appendix F.1.1. □

Since both the upper bound and the lower bound of problem (FD) are available, an optimality gap of the solution $\mathbf{x}^{RD}$ can be estimated using

$$\texttt{Gap} = \frac{OBJ_{FD}(\mathbf{x}^{RD}) - OBJ_{RD}}{OBJ_{FD}(\mathbf{x}^{RD})} \times 100\% \tag{9.9}$$

#### 9.3.4.2 Relationship between the clustering error and the optimality gap

The clustering error refers to the sum of deviations from the mean or the medoids in the $k$-means or $k$-medoids clustering, respectively. The mathematical expressions for the clustering errors are given in Equations (9.1) and (9.3a) for $k$-means and $k$-medoids, respectively. As the number of clusters increases, the clustering error is guaranteed to decrease. Intuitively, one would expect the optimality gap defined in Equation (9.9) to decrease as the clustering error diminishes, or at least, the upper bound, $OBJ_{FD}(\mathbf{x}^{RD})$, to improve as $k$ increases. Our computational results in section 9.4 show that this actually happens in most cases. However, no formal characterization is available in the literature. Here, we provide several mathematical properties.

A relatively simple question is whether the optimality gap is zero if the clustering error is zero. For the input-based approach, the following theorem holds.

**Theorem 7.** For the input-based approach, if the clustering error is zero and the integrality constraints on variables $y_{t,d}$ are not relaxed in (RD), then the optimality gap defined in (9.9) is zero, i,e., $\mathbf{x}^{RD}$ is optimal for (FD).

*Proof.* The proof is provided in Appendix F.1.2. □

For the cost-based approach, we can prove the following theorem.

**Theorem 8.** For the cost-based approach, if the $k$-medoids is used and the clustering error is zero, $|\mathcal{K}| = 1$, $|\mathcal{T}| = 1$, and the integrality constraints on variables $y_{t,k}$ are not relaxed in (RD), then the optimality gap defined in (9.9) is zero, i,e., $\mathbf{x}^{RD}$ is optimal for (FD).

*Proof.* The proof is provided in Appendix F.1.3. □

It is easy to observe that additional assumptions are needed in Theorem 3 than in Theorem 2, such as the conditions that the number of clusters and the number of years are both one. This is due to the fact that transforming the data to the cost domain entails loss of information. In other words, any two days $d$ and $d'$ with the same input data have the same optimal costs but not vice versa. When the assumptions in Theorem 3 are not satisfied, the cost-based approach is not guaranteed to find the optimal investment decisions and is, therefore, a rather empirical approach compared with the input-based approach. However, computational results in section 9.4 indicate the cost-based approach is empirically favorable under certain criteria.

An additional relevant question is whether the lower bound and the upper bound improve as $k$ increases. The following Theorem provides a sufficient condition to improve the lower bound. A similar Theorem can be found in Teichgraeber and Brandt (2019) for LP problems.

**Theorem 9.** Suppose that (RD) is solved with cluster number $|\mathcal{K}_1|$ and $|\mathcal{K}_2|$, $|\mathcal{K}_1| > |\mathcal{K}_2|$ that come from a $k$-means clustering algorithm. Denote the objective value of (RD) as $OBJ_{RD}^{\mathcal{K}_1}$, $OBJ_{RD}^{\mathcal{K}_2}$ for $|\mathcal{K}_1|$ and $|\mathcal{K}_2|$, respectively. If each cluster in the $|\mathcal{K}_1|$ clusters is contained in one of the clusters of the $|\mathcal{K}_2|$ clusters, then $OBJ_{RD}^{\mathcal{K}_1} \geq OBJ_{RD}^{\mathcal{K}_2}$.

*Proof.* The proof is provided in Appendix F.1.4. □

This theorem provides a condition for lower-bound improvement in $k$-means clustering. Following Theorem 4, it is easy to see that the lower bound provided by (RD) improves after the extreme days are added when $k$-means clustering are used because the added extreme days are contained in some of the original clusters.

## 9.4 Computational results

The case study of the Electric Reliability Council of Texas (ERCOT) reported in Li et al. (2020b) is used to test the representative day selection methods analyzed in this chapter. The problem is a generation transmission expansion planning problem with a planning horizon of 5 years. The ERCOT region is modeled using 5 nodes, South, Coast, Northeast, West, and Panhandle. The investment decisions include the number of coal, natural gas, nuclear, solar, and wind generating units, the number of storage units that are installed in each node, and the number of transmission lines that are installed connecting any two nodes each year. The operating decisions involve unit commitment, and must comply with the DC power flow equations. The historical dataset $\mathcal{D}$ consists of 365 days.

The data used for clustering can be the input data or cost data. Both $k$-means and $k$-medoids clustering algorithms are used. For the input-based approach, the extreme days are selected using the load shedding cost approach. For the cost-based approach, both the load shedding cost and the highest cost approach are used.

All the models and algorithms are implemented using Pyomo/Python (Hart et al., 2011). $k$-means clustering is solved using the heuristic provided by scikit-learn (Pedregosa et al., 2011). $k$-medoids clustering is solved with the MILP formulation shown in (9.3) using the solver CPLEX (IBM, 2015).

### 9.4.1 Comparison of different algorithms

To test the proposed representative day selection methods, the following six algorithm options (shown in Table 9.1) are tested. The computational results of the six algorithm

Table 9.1: The input data, clustering algorithm, and extreme day method used by the six cases.

| Algorithm option | Data | Clustering algorithm | Extreme day method |
|---|---|---|---|
| 1 | Input | $k$-means | load shedding cost |
| 2 | Input | $k$-medoids | load shedding cost |
| 3 | Cost | $k$-medoids | highest cost |
| 4 | Cost | $k$-medoids | load shedding cost |
| 5 | Cost | $k$-means | highest cost |
| 6 | Cost | $k$-means | load shedding cost |

Table 9.2: Computational results of the six algorithm options

| Option | $k$ | #infeasible day | $X$ | $OBJ_{FD}(\mathbf{x}^{RD})$ | $OBJ_{RD}{}^{*}$ | Gap$^{*}$ | Extreme days selected |
|---|---|---|---|---|---|---|---|
| 1 | 5 | 70 | 3 | 79.16 | 76.09 | 4.0% | 212,213,214 |
| | 10 | 63 | 2 | 79.04 | 76.29 | 3.6% | 212,213 |
| | 15 | 42 | 2 | 78.81 | 76.58 | 2.9% | 212,213 |
| 2 | 5 | 35 | 3 | 78.92 | *76.98* | - | 212,213,177 |
| | 10 | 21 | 2 | 78.72 | *73.10* | - | 212,213 |
| | 15 | 40 | 2 | 78.74 | *76.18* | - | 212,213 |
| 3 | 5 | 98 | 5 | 78.83 | *72.74* | - | 221,249,212,213,177 |
| | 10 | 13 | 3 | 78.67 | *77.39* | - | 221,212,213 |
| | 15 | 12 | 3 | 78.81 | *76.05* | - | 221,212,213 |
| 4 | 5 | 98 | 3 | 78.93 | *72.51* | - | 212,213,177 |
| | 10 | 13 | 2 | 78.79 | *77.32* | - | 212,213 |
| | 15 | 12 | 1 | 78.75 | *75.94* | - | 212 |
| 5 | 5 | 34 | 4 | 78.98 | 76.16 | 4.2% | 221,249,212,213 |
| | 10 | 30 | 6 | 79.09 | 76.64 | 3.7% | 221,249,212,213,177,214 |
| | 15 | 29 | 4 | 78.98 | 76.74 | 3.4% | 221,249,212,213 |
| 6 | 5 | 34 | 3 | 79.12 | 76.15 | 3.9% | 212,213,177 |
| | 10 | 30 | 4 | 78.93 | 76.63 | 3.0% | 212,214,213,177 |
| | 15 | 29 | 3 | 78.81 | 76.73 | 2.7% | 212,177,213 |

options are shown in Table 9.2 where $k$ represents the initial number of representative days for the problem (RD). "#infeasible day" represents the number of infeasible days resulting from the investment decisions of the $k$ representative days problem. $X$ is the number of extreme days added in addition to the $k$ representative days to make $\mathbf{x}^{RD}$ feasible for all $d \in \mathcal{D}$, i.e., to attain $OBJ_{FD}(\mathbf{x}^{RD}) < +\infty$. The objective value of the (RD), $OBJ_{RD}$, after the $X$ extreme days are added, are reported. Note that $OBJ_{RD}$ is only a valid lower bound when the $k$-means clustering is used. The values of $OBJ_{RD}$ for algorithm options 2,3,4 that use the $k$-medoids clustering are shown in italics in Table 9.2 since they do not provide valid lower bounds. $OBJ_{FD}(\mathbf{x}^{RD})$ is the "actual cost" when evaluating the investment decision of (RD) on the full dataset and therefore provides an upper bound. The "Gap" is calculated using Equation (9.9) for algorithm option 1,5,6, where $k$-means clustering is used. The days in the historical dataset $\mathcal{D}$ are numbered from 1 to 365. The "extreme days selected" column shows the indices of the extreme days.

In most of the algorithm options, $OBJ_{FD}(\mathbf{x}^{RD})$ (upper bound) and $OBJ_{RD}$ (provided that it's a valid lower bound) improve as $k$ increases. If $k$-means clustering is used, the optimality gap improves as $k$ increases. However, the upper bound is usually better when $k$-medoids clustering is used. We conclude that there is no clear winner between the $k$-medoids and the $k$-means clustering. When $k$-means clustering is used, the lowest optimality gap is achieved in algorithm option 6 with $k = 15$. There is a trend of decrease in the optimality gap as $k$ increases in option 1,5, and 6 if the $k$-means is used. If one wishes to achieve an even lower optimality gap, $k$ has to be increased. For our analysis, we consider that the optimality gap of 2.7% is sufficient to show the capability of the algorithms. It should be noted that the suboptimality comes from both the use of representative days instead of the full dataset and the relaxation of the integer variables in the operating subproblems. The relaxation of the integer variables result in an gap of around 1% in our computational experiments (Li et al., 2020b).

The number of infeasible days if using only the initial representative days is a good

* The values of $OBJ_{RD}$ for algorithm option 2,3,4 are not valid lower bounds and are shown in italics. Therefore, the gaps for these options are omitted.

indicator of the effectiveness of the algorithm. One would expect the algorithms with better performance to have fewer infeasible days. Overall, as expected, the number of infeasible days decreases as $k$ increases. The cost-based approach with $k$-medoids clustering and 15 representative days has the smallest number of initial infeasible days.

We note that $X$, the number of extreme days added to make $\mathbf{x}^{RD}$ feasible for the whole dataset $\mathcal{D}$, is an indicator of whether the extreme day selection approach is effective. The load shedding cost approach needs a smaller number of extreme days to achieve feasibility than the highest cost approach. Among all the algorithm options, there is an overlap in the extreme days selected. Days number 212,214,221, 177, are most frequently selected. The two extreme days selection approaches sometimes select the same days as extreme days. It should be noted that day 177 happens to be the day with the peak load. The highest ramp occurs in day 12, which is not selected by our proposed extreme day selection algorithms. The results show that our methods select sometimes different extreme days than just focusing on the input data, such as peak load and peak ramp. Selecting extreme days solely based on peak load and peak ramp does not guarantee feasibility.

Next, we analyze the computational time of different algorithm options. All the problems are solved using CPLEX version 12.9.0.0 (IBM, 2015) using one processor of an Intel Xeon (2.67GHz) machine with 64 GB RAM. For the cost-based approach, a CEP problem is solved for each day in the dataset $\mathcal{D}$ to obtain the optimal investment costs. To save computational time, the LP relaxations of these CEP models are solved. As a matter of fact, we also tested solving these CEP models with the integrality constraints and find that the optimal investment costs vary very little with and without the integrality constraints. The total computational time to solve the LP relaxations of these 365 CEP models is 2,091 seconds. Additionally, the solution time of the Benders decomposition algorithm for different algorithm options with and without extreme days are shown in Table 9.3. All these CEP models can be solved within 10 hours. Although the sizes of the CEP models does not depend on the algorithm option if they have the same number of representative days, their solution time can vary significantly.

Table 9.3: Solution time of the Benders decomposition algorithm for different algorithm options with and without adding $X$ extreme days (secs)

| Option | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $k$=5 | 938 | 1,460 | 2,078 | 2,392 | 1,004 | 1,436 |
| $k$=5+$X$ | 3,079 | 6,769 | 16,961 | 15,240 | 3,567 | 2,705 |
| $k$=10 | 1,520 | 9,175 | 4,404 | 4,557 | 2,897 | 3,367 |
| $k$=10+$X$ | 3,190 | 12,722 | 6,223 | 17,896 | 6,905 | 6,994 |
| $k$=15 | 4,647 | 17,914 | 26,514 | 27,019 | 6,235 | 7,433 |
| $k$=15+$X$ | 5,095 | 9,126 | 25,320 | 31,899 | 10,568 | 13,395 |

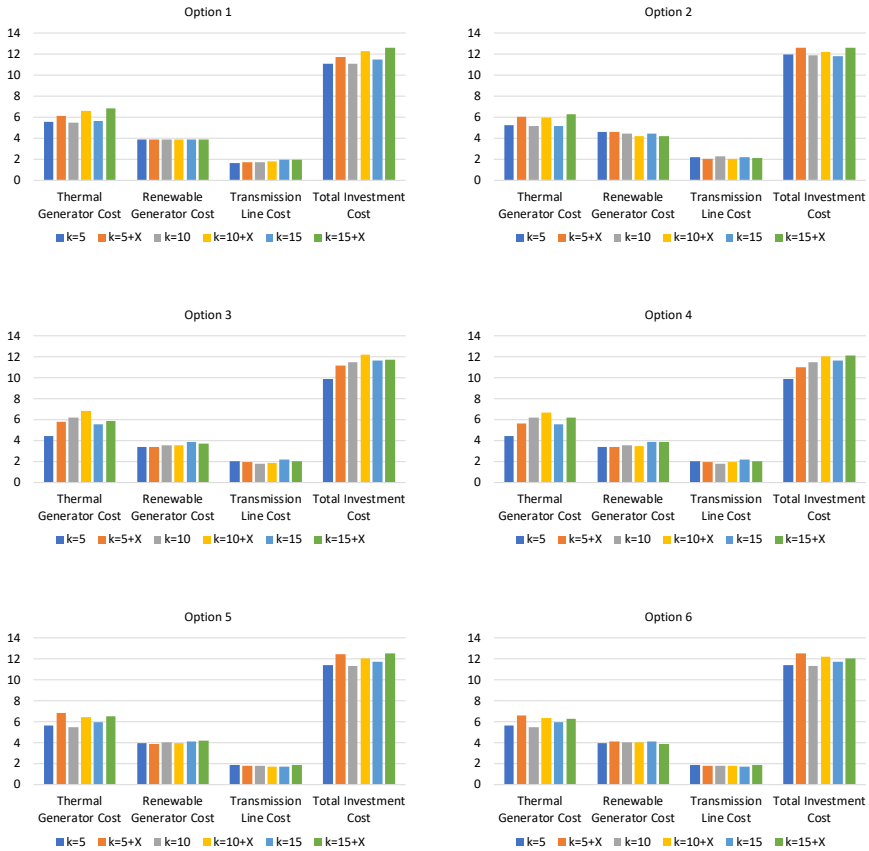


Figure 9.3: Thermal generating unit cost, renewable generating unit cost, transmission line cost and total investment cost change with and without the extreme days

### 9.4.2 Comparison of the capacity expansion results with and without adding the extreme days

Once the extreme days are added, the investment decisions $\mathbf{x}^{RD}$ change compared with the investment decisions that are optimal for the initial $k$ representative days. We compare in Figure 9.3 the thermal generator cost, the renewable generator cost, the transmission line cost, and the total investment cost in the 5 years of the planning horizon with and without adding the extreme days. In this Figure, $k = 5$ denotes the costs associated with the 5 representative day problem. $k = 5 + X$ denotes the problems with the initial 5 days plus the $X$ extreme days. In all the algorithm options, the total investment cost increases after adding the $X$ extreme days. The majority of this increase comes from the increase in the thermal generator costs while there is no consistent trend in the change in the costs of the renewable generators and the transmission lines. This can be explained by the fact that more dispatchable generating units are needed after the extreme days are included.

To provide a more quantitative view of the change in the costs, we select option 6, where we use cost-based $k$-means clustering, start with 15 representative days, and add extreme days according to the load shedding cost. After adding 3 extreme days to make all the days feasible, the total investment cost increases from 11.74 trillion dollars to 12.06 trillion dollars (2.7% increase). The number of transmission lines increases from 15 to 16. There is a small increase as well in the storage investment cost (0.2 million dollars). Additionally, there is an increase in total thermal generator cost (346 million dollars) and a decrease in renewable generator cost (-212 million dollars). In this case, the portfolio of the dispatchable and nondispatchable generating units is adjusted after the extreme days are included in order to make the planning decisions suitable for the extreme events.

## 9.5 Conclusions

In the context of power system expansion planning, we have presented an input-based approach and an cost-based approach for the selection of representative days. Two novel

extreme day selection algorithms, known as load shedding cost and highest cost, are proposed. The properties of the proposed algorithms are theoretically analyzed. In particular, an upper bound and an lower bound of the optimal objective value of the fullspace problem (FD) are obtained. We also identify the conditions under which zero clustering error implies zero optimality gap for both input-based and cost-based approaches. It turns out that the conditions for the cost-based approach are more restrictive than the input-based approach because there is an information loss in projecting the data from the input space to the cost space.

A case study of the ERCOT region is used to compare the different proposed algorithms. There is no clear winner between the $k$-means clustering and the $k$-medoids clustering algorithms in terms of the upper bound. However, the $k$-means clustering is recommended since it provides a valid lower bound. The cost-based approach and the input-based approach have varying performances in combination with different clustering and extreme day selection methods. The load shedding cost approach outperforms the highest cost approach in extreme day selection in the sense that it needs fewer extreme days to achieve feasibility. We also compare the change in the investment decisions with and without adding the extreme days. The major change is that additional dispatchable generating units are added to make the planning decisions feasible under extreme events.

# Chapter 10

# Conclusions

## 10.1 Summary and limitations of this thesis

In this section, we summarize the major findings and accomplishments in each chapter. We also discuss some limitations on chapters 3-9.

### 10.1.1 A Review of Stochastic Programming Methods for Optimization of Process Systems under Uncertainty

#### Summary

In chapter 2, we provide an overview of stochastic programming in process systems engineering. The review is intended for an engineering audience without an extensive background in mathematical optimization. We first provide a historical perspective of how the concept of stochastic programming was developed by the mathematical programming community. The applications of stochastic programming in process systems engineering are widespread, from the traditional petrochemical, pharmaceutical industry to carbon capture, energy storage.

There are two major types of uncertainties that can be modeled using stochastic programming, exogenous uncertainty, which is the most commonly considered, and endogenous uncertainty where realizations of the uncertainties depend on the decisions taken.

In this review chapter, we first review the stochastic programs with exogenous uncertainty. The mathematical formulation of two-stage stochastic programming is introduced and illustrated with a process network problem under demand uncertainty. We also review the classical decomposition algorithms, including Benders decomposition and Lagrangean decomposition. The works that have been done by the OR community on stochastic mixed-integer programs with integer recourse are not reviewed in detail, for which we have referred to the paper by Küçükyavuz and Sen (2017). Recent advances have been made by the PSE community on the algorithms for stochastic mixed-integer nonlinear programming (SMINLP). A brief review is conducted for SMINLP algorithms together with an illustrative example of the stochastic pooling problem. A generalization of two-stage stochastic programming to the sequential realization of uncertainties is called "multi-stage stochastic programming". Both the node-based formulation and the scenario-based formulation are reviewed. We also provide an illustrative example of the process network problem. Additionally, we review the algorithms to solve multi-stage stochastic programs including the nested Benders decomposition, SDDP/SDDiP, and Lagrangean decomposition. The details of these algorithms are not provided.

We further introduce the mathematical formulation for multistage stochastic programs with endogenous uncertainty. Both heuristic algorithms and rigorous Lagrangean decomposition algorithms are reviewed for solving this type of problem. A process network problem is given as an illustrative example.

Last but not least, we review the property matching methods and sampling-based methods for generating the scenario trees. The mathematical formulation of the distribution matching problem is given. The review of the scenario generation method is by no means exhaustive. Theoretical properties of the sampling-based methods can be found in Shapiro et al. (2014).

We provide several future research directions in this chapter as well. The details will be discussed in the next section on future research directions for this thesis.

### 10.1.2 An Improved L-shaped Method for Two-stage Convex 0-1 Mixed-Integer Nonlinear Stochastic Programs

#### Summary

In chapter 3, we have proposed an improved L-shaped method that can solve two-stage convex 0-1 mixed-integer nonlinear stochastic programs with mixed-integer variables in both the first and the second stage effectively. The proposed algorithm is a Benders-like decomposition algorithm that includes (strengthened) Benders cuts and Lagrangean cuts in the Benders master problem. The Benders cuts are derived by solving the Benders subproblems, which are the NLP relaxations of the subproblems for each scenario after the first stage decisions are fixed at the optimal solution of the Benders master problem. The strengthened Benders cuts are derived by adding rank-one lift-and-project cuts to the Benders subproblem. The lift-and-project cuts are generated by outer-approximating the convex nonlinear functions and solving the cut generating linear programs. The Lagrangean cuts are derived by solving the Lagrangean subproblems whose multipliers are updated by a new subgradient method. The validity and the tightness of the Lagrangean cuts are proved in this chapter.

Extensive computational studies are performed. Different decomposition algorithms including the proposed algorithms with and without the Benders and Lagrangean cuts, Lagrangean decomposition, and progressive hedging, are implemented and tested using the planning problem under demand and price uncertainty, and the batch plant design problem under demand uncertainty. It is shown that the combination of the Lagrangean cuts and the strengthened Benders cuts is able to provide the tightest lower bound among all the decomposition algorithms that we test, although it requires more time than algorithms like LD. We also compare the proposed algorithm with standard convex MINLP solvers. It is shown that the proposed algorithm is a more effective computational strategy to solve problems of the same type in PSE applications compared with directly solving the deterministic equivalent.

#### Limitations

The proposed algorithm is not guaranteed to close the optimality gap of the problems due to the duality gap of the Lagrangean cuts and the integrality gap of the strengthened Benders cuts. Therefore, a more rigorous algorithm that has finite convergence would also be desirable, which is reported in chapter 4.

Although we have shown from the computational results that the Benders cuts do not strictly dominate the Lagrangean cuts or vice versa, it would be desirable to theoretically prove that. The results from the primal characterization of Lagrangean relaxation are useful in such a proof.

Rank-one lift-and-project cuts require solving multiple cut generating linear programs (CGLP) that are computationally expensive. A better implementation of the cut-generating process would be desirable. Parametric Gormory cuts are cheaper to generate than lift-and-project cuts (Zhang and Küçükyavuz, 2014). Balas and Perregaard (2003) propose to generate lift-and-project cuts based on the simplex tableau, which is computationally more efficient than solving the CGLP.

### 10.1.3 A Finite $\epsilon$-convergence Algorithm for Two-Stage Stochastic Convex Nonlinear Programs with Mixed-binary First and Second-stage Variables

#### Summary

In chapter 4, a generalized Benders decomposition-based branch-and-bound algorithm, GBDBAB, is proposed to solve two-stage convex mixed-integer nonlinear stochastic programs with mixed-binary variables in both first and second-stage decisions. Due to the nonconvexity in the second stage problem, GBD cannot be applied to solve the problem directly. We propose to construct the convex hull of the second stage problem for each scenario. In order to obtain the convex hulls of the mixed-binary nonlinear subproblems in closed-form, each subproblem is first represented in conjunctive normal form (CNF), and we apply basic steps

to transform CNF to disjunctive normal form (DNF). The convex hull of each subproblem can be represented by the hull relaxation of the DNF. The convex hull formulation involves perspective functions that can be numerically unstable. To resolve the numerical issue, we use the approximation from Furman et al. (2016). The resulting convex problem is amenable to the GBD algorithm since the Benders subproblems are continuous and convex.

Although we have constructed the convex hull for each scenario subproblem, it is not equivalent to the convex hull of the deterministic equivalent problem. We have established several equivalence results in several special cases. For the problems with pure binary first-stage variables, we are able to prove that GBD has finite $\epsilon$-convergence if the convex hull of each subproblem is constructed in closed-form. For the problems with mixed-binary first-stage variables, we also prove that in order to have finite $\epsilon$-convergence, we may need to branch on the first stage continuous variables. A branching scheme and a node selection scheme are proposed for the branch and bound algorithm. To reduce the computational burden in solving the extensive form of the convex hull subproblem, a sequential convexification scheme is proposed to adaptively apply basic steps for the subproblems with a large number of binary variables. A sufficient condition that the solution is already in the convex hull is established in Proposition 3, i.e., if the current optimal solution already satisfies the condition described in Proposition 3, there is no need to apply more basic steps.

The planning problem under uncertainty, the constrained layout problem, and the illustrative example with quadratic constraints are studied. The problems with an increasing number of scenarios are solved with GBDBAB to within 0.1% optimality gap. The results show the advantage of the sequential convexification scheme in the sense that the convex hull representation can be avoided in some applications. Convex MINLP solvers including SBB, AlphaECP, and DICOPT are used to solve the corresponding DEFs to benchmark the proposed algorithm. It is shown that the proposed algorithm outperforms the solvers for the problems with a large number of scenarios. Especially, the algorithm is preferable for highly nonlinear problems with poor relaxations but with a small number of binary variables in the second-stage decisions like the constrained layout problem.

#### Limitations

We have proved that the algorithm can converge to an $\epsilon$-optimal solution in a finite number of iterations. It would also be interesting to consider the asymptotic convergence proof based on the spatial branch and bound proof in Horst and Tuy (2013).

Despite the sequential convexification scheme, solving the convexified subproblems remains the main computational burden of the proposed algorithm. More work needs to be done on improving the computational efficiency of the subproblems.

We have only proposed one branching scheme and one node selection scheme. To improve the computational efficiency of the algorithm, other branching and node selection schemes should be tested over larger case studies.

### 10.1.4 A Generalized Benders Decomposition-based Branch and Cut Algorithm for Two-stage Stochastic Programs with Nonconvex Constraints and Mixed-binary First and Second Stage Variables

#### Summary

In chapter 5, we have proposed a generalized-Benders decomposition-based branch and cut algorithm for two-stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables. The algorithm is based on a generalized Benders decomposition (GBD) framework. We include both Lagrangean cuts and Benders cuts in the Benders master problem. The Lagrangean cuts are derived by solving the Lagrangean subproblems whose multipliers are updated by a new subgradient method. The validity and the tightness of the Lagrangean cuts are proved similar as in chapter 3. Since the stage two problems are nonconvex nonlinear and have integer variables, GBD cannot be applied directly. We propose to convexify the stage two problems using convex envelopes. The problems are further strengthened using cutting planes such as lift-and-project cuts. With these convexification schemes, valid Benders cuts can be derived.

Similar to the convex case in chapter 3, the addition of two different types of cutting planes cannot guarantee global optimality. We propose a branch-and-bound algorithm where we only branch on the first stage variables. Different branching rules and a node selection rule are proposed. The convergence of the proposed algorithm relies on adding Lagrangean cuts to the Benders master problem, which follows from the convergence proof of general branch and bound by Horst and Tuy (2013). The properties that we have to establish include "exhaustiveness", "bound improving", "deletion by infeasibility is certain in the limit", "strongly consistent", which are proved in Lemmas 1-4, respectively.

The proposed algorithm is implemented in Julia/JuMP using the PlasmoGraph data structure. Three applications from PSE are solved using the proposed algorithm in comparison with using MINLP solvers to solve the deterministic equivalent problems. Our computational results show that adding cutting planes can potentially accelerate the convergence of the branch-and-bound procedure, especially in the stochastic pooling problem with contract selection. However, in the storage design for a multi-product plant under uncertainty problem, adding rank-one lift-project-cut does not help the branch-and-bound procedure.

### Limitations

For the problems with tight Lagrangean relaxation, the corresponding Benders cuts derived from the Benders subproblem convexified by cutting planes would be dominated by the Lagrangean cuts (storage design problem). On the other hand, for the problems with weak Lagrangean relaxation (stochastic pooling problem and crude selection problem), adding Benders cuts may be able to close the duality gap significantly. Therefore, in order to predict whether it is preferable to add cutting planes, heuristics need to be proposed to decide the cut generation.

In the spatial branch and bound process, a hybrid of the proposed branching rules is used. It would be desirable to have a large benchmark library of stochastic programming problems of the proposed category in order to test different hybrid branching rules.

While the proposed algorithm can solve the test cases to small optimality gaps, solving

stochastic nonconvex MINLPs is still challenging. It would be desirable to develop tighter convex relaxations for the nonconvex terms in stage 2 in order to generate tight Benders cuts. We have shown that in the crude selection problem, using the hull relaxation of piecewise McCormick envelope, yields smaller optimality gaps than using the McCormick envelope. Tighter convex relaxations are in general more difficult to solve. Therefore, there is a tradeoff between using tight convex relaxations and doing more branching. We propose to perform more computational studies in the future to learn the appropriate convex relaxations based on the features of each problem to make the proposed algorithm more efficient.

For our implementation of the algorithm, the convex relaxations of the nonconvex nonlinear functions have to be provided by the user. In order to make this step automatic, significant extra work has to be done, which is almost equivalent to implementing a global solver.

### 10.1.5 Sample Average Approximation for Stochastic Nonconvex Mixed Integer Nonlinear Programming via Outer-Approximation

#### Summary

In chapter 6, we propose a sample average approximation based outer-approximation (SAAOA) algorithm for solving two-stage nonconvex stochastic MINLPs. Like the classical outer approximation algorithm, the SAAOA algorithm iterates between an MILP master problem and an NLP subproblem. The master problem (SAA-OA-MILP) is constructed by generating N i.i.d. samples and apply convex relaxations to the nonconvex nonlinear functions. The NLP subproblem (SAA-nonconvex-NLP) restricts the binary variables at a given value and solves the nonconvex subproblems with N i.i.d. samples. The SAAOA algorithm iterates between an MILP master problem (SAA-OA-MILP) and nonconvex NLP subproblems (SAA-nonconvex-NLP). At the end of each iteration, a “no good” cut is added to the master problem to eliminate binary combinations that have been visited in the previous iterations.

Based on the results from Shapiro et al. (2014); Schultz (1995), we prove that the SAAOA algorithm converges as the sample size tends to infinity and provides a theoretical estimate for the sample size. Since the sample size estimates are too conservative in practice, we design an SAAOA algorithm with confidence interval estimates for the upper bound and the lower bound at each iteration of the OA algorithm. To construct the confidence intervals, we define (single-OA-LP) and (single-nonconvex-NLP), which are proved to provide upper estimators for the problem's lower bound and upper bound, respectively. The sample sizes are updated dynamically using some update policies. We propose two update policies, namely, *P1: Increase if loose confidence interval policy* and *P2: Increase until overlap policy*. The algorithm is suitable for solving two-stage stochastic MINLPs with pure binary variables where the nonconvex NLP subproblems can be solved for each scenario separately.

An illustrative example of the process network planning is provided. Computational results are shown for a stochastic pooling problem. The SAAOA algorithm with confidence interval estimates is shown to perform better than solving the deterministic equivalent (SAA-MINLP) directly in terms of computational time and optimality gap. We provide some criteria for selecting between update Policy 1 and Policy 2.

### Limitations

More work can be done in improving the update policies. To find an update policy that works well in general, we need to have more test cases to benchmark different update policies and tune the parameters like the fixed ratio in the two update policies proposed in this chapter.

Although the algorithm can be used to address stochastic nonconvex MINLPs with mixed-binary first stage variables in theory, it is only computationally efficient for stochastic nonconvex MINLPs with pure binary first stage variables. Therefore, another possible future direction would be making the algorithm more efficient for problems with mixed-binary first stage variables.

### 10.1.6 Shale Gas Pad Development Planning under Price Uncertainty

#### Summary

In chapter 7, we study the shale gas pad development problem under price uncertainty. The deterministic model is similar to the MILP model proposed by Ondeck et al. (2019). A wellpad is given with several prospective wells. The productivity profile for each well is also given. In order to complete a shale gas well, four operations, i.e., top setting, horizontal drilling, fracturing, turning in line, need to be done sequentially. The upstream operator needs to identify the optimal sequence of operations to maximize the total NPV.

To extend the work of Ondeck et al. (2019), we investigate how price uncertainties can affect decision-making in this context. The mathematical framework that we use in this chapter is stochastic programming, which is regarded as a risk-neutral approach to hedge against uncertainty. We introduce some concepts to quantify the value that stochastic programming can create, such as the value of the stochastic solution (VSS). We apply both two-stage and multistage stochastic programming to the development of a wellpad with 9 wells under price uncertainty. When the variance of price is high, we can obtain values using stochastic programming on the order of million dollars. The stochastic solutions in both two-stage and multistage try to delay the development decisions to the last stage when the uncertainty of the prices is fully realized. We also demonstrate that multistage stochastic programming is a better approximation of what happens in the real world than two-stage stochastic programming since it allows the uncertainties to evolve every quarterly.

However, when the price variance is low, there is no value in using stochastic programming since all the scenarios have the same recourse decisions. This gives us some insights on when to use stochastic programming. There may not be any value in using stochastic programming even if there are uncertain parameters in the model. Stochastic programming can only add value if there is something that we can do differently than the deterministic solution.

#### Limitations

The generation of scenario trees can be improved by using more realistic forecast models. The aim of using the high and low variance scenario trees is to evaluate the stochastic programming approach. We will leave the incorporation and better forecast models and expert knowledge in building the scenario as one future direction.

The choice between two-stage and multi-stage stochastic programming for this real-world application can be discussed in detail. Numerical experiments can be performed to compare the solutions from two-stage and multi-stage stochastic programs. As far as we know, there have been no general theoretical results on this topic.

Furthermore, future work can also concentrate on extending the model to multiple pads development problems, and considering the mobilization of crew and equipment between the wellpads. We expect the optimization problem to grow even larger once more wellpads are taken into account. Therefore, better solution techniques need to be developed to solve the large-scale stochastic programming problem with higher efficiency.

### 10.1.7 Mixed-integer Linear Programming Models and Algorithms for Generation and Transmission Expansion Planning of Power Systems

#### Summary

In chapter 8, we propose a generation and transmission expansion planning (GTEP) model that not only includes the investment decisions on the thermal and renewable generating units, transmission lines, and storage units, but also considers the detailed operating conditions of the generators to guarantee the feasibility of the investment decisions subject to volatile weather conditions. To limit the size of the GTEP model, both spatial and temporal simplifications are made. We aggregate the generators that use the same technology assuming that they have the same design parameters. We also spatially aggregate regions with similar climate and load profiles. We compare three different formulations for transmis-

sion expansion, i.e., the big-M formulation, the hull formulation, and the alternative big-M formulation. We prove that the alternative big-M (ABM) formulation has the same feasible region as the big-M formulation (BM) when projected onto the space of the variables involved in the big-M formulation. Computational experiments are performed as well for the three formulations. It is shown that none of the three formulations is superior as each has its strengths and weaknesses.

Despite the spatial and temporal simplifications, the MILP model still has tens of millions of variables, which cannot be solved directly by the state-of-the-art commercial solvers. Two solution techniques, a nested Benders decomposition algorithm and a tailored Benders decomposition algorithm, are proposed. The tailored Benders decomposition algorithm outperforms the nested Benders decomposition in our computational experiments and is able to solve the large MILP model within reasonable computational time.

An ERCOT case study is used to demonstrate the GTEP model and the solution techniques. The tailored Benders decomposition is able to solve the 20-year planning problem with up to 15 representative days. The capacity expansion mix for ERCOT will mainly include solar and wind capacities in the West and Panhandle regions. The transmission lines are mainly built to transfer power from solar and wind-rich regions to the South and Northeast regions of ERCOT, which shows that the generation and transmission decisions are correlated. Co-optimization of generation and transmission has the potential to bring additional value to the system operator/regulator than solving the two planning problems independently.

### Limitations

The aggregation of generating units and transmission lines in ERCOT is a simplification of the system in the real world, which can yield suboptimal solutions compared with modeling each generator individually. Such simplification is necessary to make the problem tractable. In order to obtain a feasible solution to the real physical system, i.e, the unit commitment decisions of each generator, one could perform a disaggregation heuristic on the aggregated

solution.

It would be desirable to report more computational results on transmission expansion planning to compare the tightness of the three formulations.

The Benders decomposition used is from CPLEX. Callbacks for the Benders decomposition implementation are not available for now. If they become available, warm-start strategies can be implemented to accelerate the Benders decomposition algorithm. Trial solutions can be generated for warm starting the algorithm. For example, we can solve the model with fewer representative days and obtain solutions to the master problem quickly.

### 10.1.8 On Representative Day Selection for Capacity Expansion Planning of Power Systems under Extreme Events

#### Summary

Chapter 9 is a continuation of chapter 8. We deal with one important issue in long-term capacity expansion planning models, i.e., the selection of representative days. In this chapter, we have presented an input-based approach and a cost-based approach for the selection of representative days. In both approaches, a dataset is given that consists of the historical load data, and capacity factor data for solar and wind generators. The historical data is then scaled to account for load growth. Based on this scaled dataset, $k$ representative days are selected in each year of the planning problem to represent the whole planning horizon where $k << 365$. A clustering algorithm, such as $k$-means clustering or $k$-medoids clustering, is used to divide the whole historical dataset into $k$ clusters based on some of the characteristics of the historical days. The centroid or the medoid of each cluster is selected as the representative day. In the input-based approach, clustering is performed directly on the historical data, i.e., load and capacity factor data. In the cost-based approach, a capacity expansion planning problem is solved for every single day in the historical dataset. The optimal investment decisions are obtained for all the historical days. Clustering is performed based on the cost of different types of investment decisions.

Using the medoid or centroid of each cluster fails to capture the extreme events. Two novel extreme day selection algorithms, known as load shedding cost and highest cost, are proposed. In the first approach, the operating problems for the infeasible days are solved again with load shedding variables added to energy balance constraints for each node at each hour. The load shedding variables are used to balance the load when the power generation is not enough to satisfy the load. The objective function is changed to minimizing the total load shedding cost over the planning horizon. The day with the highest total load shedding cost is selected as the extreme day. In the highest cost approach, the day with the highest cost in the cost-based approach is selected as the extreme day.

The properties of the proposed algorithms are theoretically analyzed. In particular, an upper bound and a lower bound of the optimal objective value of the fullspace problem (FD) are obtained. We also identify the conditions under which zero clustering error implies zero optimality gap for both input-based and cost-based approaches. It turns out that the conditions for the cost-based approach are more restrictive than the input-based approach because there is an information loss in projecting the data from the input space to the cost space.

A case study of the ERCOT region is used to compare the different proposed algorithms. There is no clear winner between the $k$-means clustering and the $k$-medoids clustering algorithms in terms of the upper bound. However, the $k$-means clustering is recommended since it provides a valid lower bound. The cost-based approach and the input-based approach have varying performances in combination with different clustering and extreme day selection methods. The load shedding cost approach outperforms the highest cost approach in extreme day selection in the sense that it needs fewer extreme days to achieve feasibility. We also compare the change in the investment decisions with and without adding the extreme days. The major change is that additional dispatchable generating units are added to make the planning decisions feasible under extreme events.

### Limitations

The optimality gap using the $k$-means clustering algorithm remains to be 3% for problems with 15 representative days. It is of interest to study problems with more representative days in order to make the optimality gap smaller.

Similar techniques can be used to study the effects of spatial aggregation in capacity expansion planning problems.

## 10.2 Research contributions

The major contributions of this thesis can be summarized as follows:

1. Provided a review of stochastic programming and its applications in the process systems engineering area.

2. Proposed an improved L-shaped algorithm for two-stage stochastic convex MINLP with mixed-binary first and second stage variables. This algorithm combines Lagrangean cuts and strengthened Benders cuts.

3. Proved the validity and the tightness properties of the Lagrangean cuts for any two-stage stochastic MINLPs.

4. Proposed a GBD-based algorithm for two-stage stochastic convex MINLP with mixed-binary first and second stage variables and prove its finite-$\epsilon$ convergence.

5. Proposed a sequential convexification scheme for the GBDBAB algorithm. Proved a sufficient condition for terminating the application of the basic steps.

6. Proposed a GBD-based branch-and-cut algorithm for two-stage stochastic nonconvex MINLP with mixed-binary first and second stage variables. Tested several applications using different convex relaxations.

7. Proposed a sample average approximation-based outer-approximation algorithm for two-stage stochastic nonconvex MINLP with continuous probability distribution, mixed-binary first stage variables, and continuous second stage variables.

8. Proposed two update policies for updating the sample size in the SAAOA algorithm with confidence interval estimates.

9. Developed two-stage and multi-stage stochastic programming models for shale gas pad development planning under price uncertainty.

10. Proposed mixed-integer linear programming models for long-term generation and transmission expansion planning problem. Theoretically compared three formulations for transmission expansion planning.

11. Adapted the Benders decomposition algorithm and the nested Benders decomposition algorithm to solve the GTEP problem using case studies of the ERCOT region.

12. Proposed an input-based approach and a cost-based approach for representative day selection in capacity expansion planning models. Proved mathematical properties, including the "optimality gap" for the representative day approach.

## 10.3 Papers produced from this dissertation

1. Li, C., A.J. Conejo, J.D. Siirola, I.E. Grossmann. On Representative Day Selection for Capacity Expansion Planning of Power Systems under Extreme Events. Under review in Energy.

2. Li, C., A.J. Conejo, P. Liu, B.P. Omell, J.D. Siirola, I.E. Grossmann. Mixed-integer Linear Programming Models and Algorithms for Generation and Transmission Expansion Planning of Power Systems. Under review in European Journal of Operations Research.

3. Li, C. and Grossmann, I. E. (2020). A Review of Stochastic Programming Methods for Optimization of Process Systems under Uncertainty. Frontiers in Chemical Engineering, 2, 34.

4. Li, C., Eason, J.P., Drouven, M.G. and Grossmann, I.E., 2020. Shale gas pad development planning under price uncertainty. AIChE Journal, 66(6), p.e16933.

5. Li, C., Bernal, D.E., Furman, K.C., Duran, M.A. and Grossmann, I.E., 2020. Sample average approximation for stochastic nonconvex mixed integer nonlinear programming via outer-approximation. Optimization and Engineering, pp.1-29.

6. Li, C. and Grossmann, I.E., 2019. A generalized Benders decomposition-based branch and cut algorithm for two-stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables. Journal of Global Optimization, 75(2), pp.247-272.

7. Li, C. and Grossmann, I.E., 2019. A finite $\epsilon$-convergence algorithm for two-stage stochastic convex nonlinear programs with mixed-binary first and second-stage variables. Journal of Global Optimization, 75(4), pp.921-947.

8. Li, C. and Grossmann, I.E., 2018. An improved L-shaped method for two-stage convex 0-1 mixed integer nonlinear stochastic programs. Computers & Chemical Engineering, 112, pp.165-179.

## 10.4 Future research directions

### 10.4.1 Stochastic Programming

#### 10.4.1.1 Algorithms for multi-stage stochastic MINLP

A natural extension of the two-stage stochastic MINLP algorithm is multi-stage stochastic MINLP. The SDDiP algorithm proposed by Zou et al. (2018a) can be adapted to multi-stage MINLP. The idea is to construct binary approximations of the continuous state variables to

apply integer cuts at each stage. Other types of cutting planes in this thesis can also be adapted to the multi-stage setting. However, the convergence of the algorithm still relies on the integer cuts. Therefore, the binarization scheme proposed by Zou et al. (2018a) does not scale well with the number of continuous state variables. How to design algorithms that are scalable for multi-stage stochastic programs remains clearly a major challenge.

#### 10.4.1.2 Tradeoffs between two-stage and multi-stage stochastic programs

Sequential decision-making under uncertainty can be modeled as multi-stage stochastic programming (MSSP) problems. However, MSSP is computationally expensive to solve for large-scale mixed-integer problems. Two-stage stochastic programming (TSSP) has been used as an approximation to the MSSP. For example Balasubramanian and Grossmann (2004) solve TSSP approximations in a rolling horizon fashion. The downside of the approximation is that the decisions in the two-stage approximations are not fully adaptive to the realizations of uncertainties. It would be desirable to quantify the difference between MSSP and its two-stage approximation in order to decide whether it is worth solving MSSP or the two-stage approximation is sufficient to obtain good solutions. Towards this goal, Huang and Ahmed (2009) provide an analytical bound for the difference between the objective of the two-stage and multi-stage capacity expansion planning problem. The analytical bound is derived based on the structure of the capacity expansion planning problem, i.e., application-specific. To the best of our knowledge, the results in Huang and Ahmed (2009) have not been generalized.

#### 10.4.1.3 Stochastic programs with endogenous uncertainty

For stochastic MINLPs with endogenous uncertainty, not many works have been done. Tarhan and Grossmann (2008) solve multi-stage stochastic nonconvex MINLP under endogenous uncertainty using an outer approximation algorithm. However, the application that can be addressed using this algorithm is limited to a simple process network problem. The algorithm fails to decompose the stochastic program by node or by scenario. Future

work can be focused on investigating new algorithms for the formulation proposed by Goel and Grossmann (2006) for stochastic programs under type 2 endogenous uncertainty as well as proposing novel formulations that are computational more effective.

#### 10.4.1.4 Feasibility issues in stochastic programming

All the decomposition algorithms proposed in this thesis assume relative complete recourse in the stochastic programs. One option to satisfy this assumption is to add slack variables in the second stage constraints and penalize them in the objective function. This strategy suffices to be a workaround for most applications in practice.

However, directly handling infeasibility is not a trivial task for nonconvex problems since a first stage solution that is infeasible for the original nonconvex second stage problem might be feasible for the convex relaxation of the second stage problem. This would make the GBD-based or OA-based algorithm fail to cut off the infeasible solution. More research is required for problems that do not satisfy the relatively complete recourse assumption.

#### 10.4.1.5 Integration of predictive/statistical learning models in stochastic programming

Traditional stochastic programming models assume that a probability distribution for the uncertain parameters is known. In the past few years, several works have been done to integrate the predictive models with the stochastic programming model (Liu et al., 2019; Sen and Deng, 2018; Bertsimas and Kallus, 2020). More research can be done in the future to extend the work to more general settings, e.g., multistage problems, heteroscedasticity.

#### 10.4.1.6 Parallel and decentralized implementation of the decomposition algorithms

Decomposition algorithms, such as Benders decomposition and dual decomposition, can be parallelized. However, a naive parallelization where all the subproblems are distributed to all the available processors and the root process waits until all the subproblems are

solved is not the optimal way because the subproblems can take very different time to solve. An asynchronous partitioned subgradient method for dual decomposition is proposed recently by Lim et al. (2021) where a subset of scenarios are considered and each iteration is asynchronous. There are opportunities to improve the parallelization of other decomposition algorithms as well.

### 10.4.2 Long term power systems planning

#### 10.4.2.1 Reliable design of power system

The expansion planning models in this thesis have not considered the potential failures in the generating units and transmission lines. The reliability of power systems has attracted increasing attention due to the increasing occurrences of extreme weather that causes blackouts. N-k contingency constraints have been studied for transmission expansion planning (Wu et al., 2018). Different contingency scenarios are considered that increase the size of the problem significantly. More general frameworks for reliable design of power system infrastructure can be developed as well as solution techniques.

#### 10.4.2.2 Analysis of spatial aggregation in the expansion planning model

In the GTEP model of the ERCOT region, we have aggregated the generating units in each subregion of ERCOT to simplify the problem. Such a simplification can lead to suboptimal solutions compared with the model that considers each generating unit separately. Mathematical analysis of the suboptimality resulting from the spatial aggregation can be part of future work. It would be desirable to derive an "optimality gap" for the spatial aggregation similar to what we have developed in chapter 9 for temporal simplification.

#### 10.4.2.3 Integrated energy systems

The power system expansion planning model can be extended to integrated energy systems where all the possible alternatives of integrating electrical generating units, natural gas networks, water, and industrial processes are considered through a superstructure. The

tightly coupled integrated energy systems may increase energy efficiency through the coordination of renewable generation and consumption. One of the challenges in developing planning models for the integrated energy system is to collect data from different sources.

#### 10.4.2.4 Including battery degradation model into the expansion planning model

We have not considered battery degradation in the expansion planning model. Instead, we assumed that the batteries have constant lifetimes that are independent of the charge and discharge. Including a model for battery degradation will increase the complexity of the GTEP model and can be part of future work.

# Bibliography

J. Acevedo and E. N. Pistikopoulos. Stochastic optimization based algorithms for process synthesis under uncertainty. *Computers & Chemical Engineering*, 22(4-5):647–671, 1998.

T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

J. Aghaei, N. Amjady, A. Baharvandi, and M.-A. Akbari. Generation and transmission expansion planning: MILP–based probabilistic model. *IEEE Transactions on Power Systems*, 29(4):1592–1601, 2014.

S. Ahmed and R. Garcia. Dynamic capacity acquisition and assignment under uncertainty. *Annals of Operations Research*, 124(1-4):267–283, 2003.

S. Ahmed, A. Shapiro, and E. Shapiro. The sample average approximation method for stochastic programs with integer recourse, 2002.

N. Alguacil, A. L. Motto, and A. J. Conejo. Transmission expansion planning: a mixed-integer LP approach. *IEEE Transactions on Power Systems*, 18(3):1070–1077, Aug 2003. ISSN 0885-8950.

A. Almaimouni, A. Ademola-Idowu, J. N. Kutz, A. Negash, and D. Kirschen. Selecting and evaluating representative days for generation expansion planning. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–7. IEEE, 2018.

A. Alonso-Ayuso, L. F. Escudero, and M. T. Ortuno. Bfc, a branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0–1 programs. *European Journal of Operational Research*, 151(3):503–519, 2003.

A.Morris. Optimizing Shale Gas Production from Well to Wire. `http://www.mccormick.northwestern.edu/news/articles/2015/06/optimizing-shale-gas-production-from-well-to-wire.html`, 2015.

J. Andrade and R. Baldick. Estimation of transmission costs for new generation. *White Paper UTEI/2016-08-1*, 2016.

G. Angulo, S. Ahmed, and S. S. Dey. Improving the integer L-shaped method. *INFORMS Journal on Computing*, 28(3):483–499, 2016.

R. M. Apap and I. E. Grossmann. Models and computational strategies for multistage stochastic programming under endogenous and exogenous uncertainties. *Computers & Chemical Engineering*, 103:233–274, 2017.

K. Arredondo-Ramírez, J. M. Ponce-Ortega, and M. M. El-Halwagi. Optimal planning and infrastructure development for shale gas production. *Energy Conversion and Management*, 119:91–100, 2016.

S. Atakan and S. Sen. A progressive hedging based branch-and-bound algorithm for mixed-integer stochastic programs. *Computational Management Science*, pages 1–40, 2018.

L. Bahiense, G. C. Oliveira, M. Pereira, and S. Granville. A mixed integer disjunctive model for transmission network expansion. *IEEE Transactions on Power Systems*, 16(3):560–565, 2001.

B. Bahl, T. Söhler, M. Hennen, and A. Bardow. Typical periods for two-stage synthesis by time-series aggregation with bounded error in objective function. *Frontiers in Energy Research*, 5:35, 2018.

E. Balas. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM Journal on Algebraic Discrete Methods*, 6(3):466–486, 1985.

E. Balas and M. Perregaard. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming. *Mathematical Programming*, 94(2):221–245, 2003.

E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming*, 58(1-3):295–324, 1993.

E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj. Gomory cuts revisited. *Operations Research Letters*, 19(1):1–9, 1996.

J. Balasubramanian and I. Grossmann. Approximation to multistage stochastic optimization in multiperiod batch plant scheduling under demand uncertainty. *Industrial & Engineering Chemistry Research*, 43(14):3695–3713, 2004.

L. Baringo and A. Baringo. A stochastic adaptive robust optimization approach for the generation and transmission expansion planning. *IEEE Transactions on Power Systems*, 33(1):792–802, 2017.

E. M. Beale. On minimizing a convex function subject to linear inequalities. *Journal of the Royal Statistical Society: Series B (Methodological)*, 17(2):173–184, 1955.

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252, 1962.

D. Bertsimas and N. Kallus. From predictive to prescriptive analytics. *Management Science*, 66(3):1025–1044, 2020.

A. B. Birchfield, T. Xu, K. M. Gegner, K. S. Shetye, and T. J. Overbye. Grid structural characteristics as validation criteria for synthetic networks. *IEEE Transactions on power systems*, 32(4):3258–3265, 2016.

J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33(5):989–1007, 1985a.

J. R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33(5):989–1007, 1985b.

J. R. Birge and F. Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

N. Blair, A. P. Dobos, J. Freeman, T. Neises, M. Wagner, T. Ferguson, P. Gilman, and S. Janzou. System advisor model, sam 2014.1. 14: General description. Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2014.

N. Boland, I. Dumitrescu, G. Froyland, and T. Kalinowski. Minimum cardinality non-anticipativity constraint sets for multistage stochastic programming. *Mathematical Programming*, 157(1):69–93, 2016.

P. Bonami, D. Salvagnin, and A. Tramontani. Implementing automatic Benders decomposition in a modern MIP solver. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 78–90. Springer, 2020a.

P. Bonami, D. Salvagnin, and A. Tramontani. Implementing automatic Benders decomposition in a modern MIP solver. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 78–90. Springer, 2020b.

B. Brunaud, C. Li, and M. Paz Ochoa. Source code of PlasmoAlgorithms. `https://github.com/bbrunaud/PlasmoAlgorithms.jl`. Accessed: 2018-11-21.

M. R. Bussieck and A. Drud. SBB: A new solver for mixed integer nonlinear programming. *Talk, OR*, 2001. URL `http://ftp.gamsworld.org/presentations/present_sbb.pdf`.

M. R. Bussieck, M. C. Ferris, and A. Meeraus. Grid-enabled optimization with GAMS. *INFORMS Journal on Computing*, 21(3):349–362, 2009.

R. H. Byrd, J. Nocedal, and R. A. Waltz. Knitro: An integrated package for nonlinear optimization. In *Large-scale Nonlinear Optimization*, pages 35–59. Springer, 2006.

D. C. Cafaro and I. E. Grossmann. Strategic planning, design, and development of the shale gas supply chain network. *AIChE Journal*, 60(6):2122–2142, 2014.

D. C. Cafaro, M. G. Drouven, and I. E. Grossmann. Optimization models for planning shale gas well refracture treatments. *AIChE Journal*, 62(12):4297–4307, 2016.

D. C. Cafaro, M. G. Drouven, and I. E. Grossmann. Continuous-time formulations for the optimal planning of multiple refracture treatments in a shale gas well. *AIChE Journal*, 64 (5):1511–1516, 2018.

B. A. Calfa, A. Agarwal, I. E. Grossmann, and J. M. Wassick. Data-driven multi-stage scenario tree generation via statistical property and distribution matching. *Computers & Chemical Engineering*, 68:7–23, 2014.

Y. Cao and V. M. Zavala. A scalable global optimization algorithm for stochastic nonlinear programs. *Under Review*, 2017.

Y. Cao and V. M. Zavala. A scalable global optimization algorithm for stochastic nonlinear programs. *Journal of Global Optimization*, 75(2):393–416, 2019.

C. C. Carøe and R. Schultz. Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1):37–45, 1999.

S. Ceria and J. Soares. Convex programming for disjunctive convex optimization. *Mathematical Programming*, 86(3):595–614, 1999.

B. Christian and S. Cremaschi. Heuristic solution approaches to the pharmaceutical r&d pipeline management problem. *Computers & Chemical Engineering*, 74:34–47, 2015.

Y. Chu and F. You. Integration of scheduling and dynamic optimization of batch processes under uncertainty: Two-stage stochastic programming approach and enhanced generalized

benders decomposition algorithm. *Industrial & Engineering Chemistry Research*, 52(47): 16851–16869, 2013.

W. Cole, P. Kurup, M. Hand, D. Feldman, B. Sigrin, E. Lantz, T. Stehly, C. Augustine, C. Turchi, P. O'Connor, et al. 2016 annual technology baseline (atb). Technical report, National Renewable Energy Lab.(NREL), Golden, CO (United States), 2016.

M. Colvin and C. T. Maravelias. A stochastic programming approach for clinical trial planning in new drug development. *Computers & Chemical Engineering*, 32(11):2626–2642, 2008.

A. J. Conejo, L. Baringo, S. J. Kazempour, and A. S. Siddiqui. Investment in electricity generation and transmission. *Cham Zug, Switzerland: Springer International Publishing*, 119, 2016a.

A. J. Conejo, L. Baringo, S. J. Kazempour, and A. S. Sissiqui. *Investment in Electricity Generation and Transmission - Decision Making under Uncertainty*. Springer International Publishing, 2016b. ISBN 978-3-319-29501-5. doi: 10.1007/978-3-319-29501-5.

I. CPLEX. 12.7. 0 user' s manual, 2016.

I. I. CPLEX. V12. 1: User's manual for CPLEX. *International Business Machines Corporation*, 46(53):157, 2009.

G. B. Dantzig. Linear programming under uncertainty. *Management science*, 1(3-4):197–206, 1955.

F. J. De Sisternes and M. D. Webster. Optimal selection of sample weeks for approximating the net load in generation planning problems. *esd. mit. edu*, 2013.

J. Ding and A. Somani. A long-term investment planning model for mixed energy infrastructure integrated with renewable energy. In *2010 IEEE Green Technologies Conference*, pages 1–10, April 2010.

L. Ding, S. Ahmed, and A. Shapiro. A python package for multi-stage stochastic programming. *Optimization online*, pages 1–41, 2019.

O. Dowson and L. Kapelevich. SDDP.jl: a Julia package for stochastic dual dynamic programming. *INFORMS Journal on Computing*, 2020. doi: https://doi.org/10.1287/ijoc.2020.0987. Articles in Advance.

C. Draxl, A. Clifton, B.-M. Hodge, and J. McCaa. The wind integration national dataset (wind) toolkit. *Applied Energy*, 151:355–366, 2015.

M. G. Drouven and I. E. Grossmann. Multi-period planning, design, and strategic models for long-term, quality-sensitive shale gas development. *AIChE Journal*, 62(7):2296–2323, 2016.

M. G. Drouven, I. E. Grossmann, and D. C. Cafaro. Stochastic programming models for optimal shale well development and refracturing planning under uncertainty. *AIChE Journal*, 63(11):4799–4813, 2017.

I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

ERCOT. Hourly Load Data Archives, 2106. URL `http://www.ercot.com/gridinfo/load/load_hist/`.

L. F. Escudero and J. F. Monge. On capacity expansion planning under strategic and operational uncertainties based on stochastic dominance risk averse management. *Computational Management Science*, 15(3-4):479–500, 2018.

L. F. Escudero, A. Garín, M. Merino, and G. Pérez. The value of the stochastic solution in multistage problems. *Top*, 15(1):48–64, 2007.

L. F. Escudero, M. A. Garín, and A. Unzueta. Scenario cluster lagrangean decomposition for risk averse in multistage stochastic optimization. *Computers & Operations Research*, 85:154–171, 2017.

L. F. Escudero, J. F. Monge, and A. M. Rodríguez-Chía. On pricing-based equilibrium for network expansion planning. a multi-period bilevel approach under uncertainty. *European Journal of Operational Research*, 2020.

A. Flores-Quiroz, R. Palma-Behnke, G. Zakeri, and R. Moreno. A column generation approach for solving generation expansion planning problems with high renewable energy penetration. *Electric Power Systems Research*, 136:232 – 241, 2016. ISSN 0378-7796.

E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

F. Forouzanfar and A. Reynolds. Joint optimization of number of wells, well locations and controls using a gradient-based algorithm. *Chemical Engineering Research and Design*, 92 (7):1315–1328, 2014.

K. C. Furman, N. Sawaya, and I. Grossmann. A computationally useful algebraic representation of nonlinear disjunctive convex sets using the perspective function. *Optimization Online*, 2016. URL `http://www.optimization-online.org/DB_FILE/2016/07/5544.pdf`.

L. Gacitua, P. Gallegos, R. Henriquez-Auba, A. Lorca, M. Negrete-Pincetic, D. Olivares, A. Valenzuela, and G. Wenzel. A comprehensive review on expansion planning: Models and tools for energy policy analysis. *Renewable and Sustainable Energy Reviews*, 98: 346–360, 2018.

D. Gade, S. Küçükyavuz, and S. Sen. Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, 144(1-2):39–64, 2014.

D. Gade, G. Hackebeil, S. M. Ryan, J.-P. Watson, R. J.-B. Wets, and D. L. Woodruff. Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. *Mathematical Programming*, 157(1):47–67, 2016.

J. Gao and F. You. Deciphering and handling uncertainty in shale gas supply chain design and optimization: Novel modeling framework and computationally efficient solution algorithm. *AIChE Journal*, 61(11):3739–3755, 2015a.

J. Gao and F. You. Shale gas supply chain design and operations toward better economic and life cycle environmental performance: Minlp model and global optimization algorithm. *ACS Sustainable Chemistry & Engineering*, 3(7):1282–1291, 2015b.

Á. García-Cerezo, L. Baringo, and R. García-Bertrand. Representative days for expansion decisions in power systems. *Energies*, 13(2):335, 2020.

H. I. Gassmann and E. Schweitzer. A comprehensive input format for stochastic linear programs. *Annals of Operations Research*, 104(1-4):89–125, 2001.

B. H. Gebreslassie, Y. Yao, and F. You. Design under uncertainty of hydrocarbon biorefinery supply chains: multiobjective stochastic programming models, decomposition algorithm, and a comparison between cvar and downside risk. *AIChE Journal*, 58(7):2155–2179, 2012.

A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

V. Goel and I. E. Grossmann. A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. *Computers & Chemical Engineering*, 28(8):1409–1429, 2004.

V. Goel and I. E. Grossmann. A class of stochastic programs with decision dependent uncertainty. *Mathematical Programming*, 108(2-3):355–394, 2006.

I. E. Grossmann. *Advanced Optimization for Process Systems Engineering*. Cambridge Series in Chemical Engineering. Cambridge University Press, 2021.

I. E. Grossmann and R. W. Sargent. Optimum design of multipurpose chemical plants. *Industrial & Engineering Chemistry Process Design and Development*, 18(2):343–348, 1979.

I. E. Grossmann and F. Trespalacios. Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. *AIChE Journal*, 59(9): 3276–3295, 2013.

O. J. Guerra, A. J. Calderón, L. G. Papageorgiou, J. J. Siirola, and G. V. Reklaitis. An optimization framework for the integration of water management and shale gas supply chain design. *Computers & Chemical Engineering*, 92:230–255, 2016.

O. J. Guerra, A. J. Calderón, L. G. Papageorgiou, and G. V. Reklaitis. Integrated shale gas supply chain design and water management under uncertainty. *AIChE Journal*, 65(3): 924–936, 2019.

M. Guignard. Lagrangean relaxation. *Top*, 11(2):151–200, 2003.

G. Guillén-Gosálbez and I. E. Grossmann. Optimal design and planning of sustainable chemical supply chains under uncertainty. *AIChE Journal*, 55(1):99–121, 2009.

A. Gupta and C. D. Maranas. Managing demand uncertainty in supply chain planning. *Computers & Chemical Engineering*, 27(8-9):1219–1227, 2003.

J.-H. Han and I.-B. Lee. Multiperiod stochastic optimization model for carbon capture and storage infrastructure under uncertainty in co2 emissions, product prices, and operating costs. *Industrial & Engineering Chemistry Research*, 51(35):11445–11457, 2012.

W. E. Hart, J.-P. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219, 2011.

W. E. Hart, C. D. Laird, J.-P. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, and J. D. Siirola. *Pyomo-optimization modeling in python*, volume 67. Springer, 2017.

L. Hellemo, P. I. Barton, and A. Tomasgard. Decision-dependent probabilities in stochastic programs with recourse. *Computational Management Science*, 15(3-4):369–395, 2018.

R. Hemmati, R.-A. Hooshmand, and A. Khodabakhshian. State-of-the-art of transmission expansion planning: Comprehensive review. *Renewable and Sustainable Energy Reviews*, 23:312–319, 2013.

R. Horst and H. Tuy. *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013.

K. Høyland and S. W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, 2001.

K. Huang and S. Ahmed. The value of multistage stochastic programming in capacity planning under uncertainty. *Operations Research*, 57(4):893–904, 2009.

IBM. IBM ILOG CPLEX Optimization Studio CPLEX User's Manual. Technical report, IBM Corp., 2015.

J. Jalving, S. Abhyankar, K. Kim, M. Hereld, and V. M. Zavala. A graph-based computational framework for simulation and optimisation of coupled infrastructure networks. *IET Generation, Transmission & Distribution*, 11(12):3163–3176, 2017.

R. Kannan. *Algorithms, analysis and software for the global optimization of two-stage stochastic programs*. PhD thesis, Massachusetts Institute of Technology, 2018.

R. Karuppiah and I. E. Grossmann. Global optimization of multiscenario mixed integer nonlinear programming models arising in the synthesis of integrated water networks under uncertainty. *Computers & Chemical Engineering*, 32(1-2):145–160, 2008.

M. Kaut. Scenario generation for stochastic programming introduction and selected methods. *SINTEF Technology and Society*, pages 1–61, 2011.

M. Kaut and S. W. Wallace. Evaluation of scenario-generation methods for stochastic programming. 2003.

M. R. Kılınç and N. V. Sahinidis. Exploiting integrality in the global optimization of mixed-integer nonlinear programming problems with BARON. *Optimization Methods and Software*, 33(3):540–562, 2018.

M. R. Kılınç, J. Linderoth, and J. Luedtke. Lift-and-project cuts for convex mixed integer nonlinear programs. *Mathematical Programming Computation*, 9(4):499–526, 2017.

J. Kim, M. J. Realff, and J. H. Lee. Optimal design and global sensitivity analysis of biomass supply chain networks for biofuels under uncertainty. *Computers & Chemical Engineering*, 35(9):1738–1751, 2011.

K. Kim and V. M. Zavala. Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. *Mathematical Programming Computation*, 10(2):225–266, 2018.

A. J. King and R. T. Rockafellar. Asymptotic theory for solutions in statistical estimation and stochastic programming. *Mathematics of Operations Research*, 18(1):148–162, 1993.

J. Kittrell and C. Watson. Don't overdesign process equipment. *Chemical Engineering Progress*, 62(4):79, 1966.

A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

B. R. Knudsen, I. E. Grossmann, B. Foss, and A. R. Conn. Lagrangian relaxation based decomposition for well scheduling in shale-gas systems. *Computers & Chemical Engineering*, 63:234–249, 2014.

G. R. Kocis and I. E. Grossmann. Relaxation strategy for the structural optimization of process flow sheets. *Industrial & Engineering Chemistry Research*, 26(9):1869–1880, 1987.

G. R. Kocis and I. E. Grossmann. Global optimization of nonconvex mixed-integer nonlinear programming (minlp) problems in process synthesis. *Industrial & Engineering Chemistry Research*, 27(8):1407–1421, 1988.

N. E. Koltsaklis and A. S. Dagoumas. State-of-the-art generation expansion planning: A review. *Applied Energy*, 230:563–589, 2018.

N. E. Koltsaklis and M. C. Georgiadis. A multi-period, multi-regional generation expansion planning model incorporating unit commitment constraints. *Applied Energy*, 158:310 – 331, 2015. ISSN 0306-2619.

L. Kotzur, P. Markewitz, M. Robinius, and D. Stolten. Impact of different time series aggregation methods on optimal energy system design. *Renewable Energy*, 117:474–487, 2018.

V. Krishnan, J. Ho, B. F. Hobbs, A. L. Liu, J. D. McCalley, M. Shahidehpour, and Q. P. Zheng. Co-optimization of electricity transmission and generation resources for planning and policy analysis: review of concepts and modeling approaches. *Energy Systems*, 7(2): 297–332, 2016. ISSN 1868-3975.

S. Küçükyavuz and S. Sen. An introduction to two-stage stochastic mixed-integer programming. In *Leading Developments from INFORMS Communities*, pages 1–27. INFORMS, 2017.

G. Laporte and F. V. Louveaux. The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142, 1993.

N. H. Lappas and C. E. Gounaris. Multi-stage adjustable robust optimization for process scheduling under uncertainty. *AIChE Journal*, 62(5):1646–1667, 2016.

C. L. Lara, D. S. Mallapragada, D. J. Papageorgiou, A. Venkatesh, and I. E. Grossmann. Deterministic electric power infrastructure planning: Mixed-integer programming model

and nested decomposition algorithm. *European Journal of Operational Research*, 271(3): 1037–1054, 2018.

C. L. Lara, J. D. Siirola, and I. E. Grossmann. Electric power infrastructure planning under uncertainty: stochastic dual dynamic integer programming (SDDiP) and parallelization scheme. *Optimization and Engineering*, pages 1–39, 2019.

H. Le Cadre, A. Papavasiliou, and Y. Smeers. Wind farm portfolio optimization under network capacity constraints. *European Journal of Operational Research*, 247(2):560–574, 2015.

S. Legg, A. Benavides-Serrano, J. D. Siirola, J.-P. Watson, S. Davis, A. Bratteteig, and C. D. Laird. A stochastic programming approach for gas detector placement using CFD-based dispersion simulations. *Computers & Chemical Engineering*, 47:194–201, 2012.

A. A. Levis and L. G. Papageorgiou. A hierarchical solution approach for multi-site capacity planning under uncertainty in the pharmaceutical industry. *Computers & Chemical Engineering*, 28(5):707–725, 2004.

C. Li and I. E. Grossmann. An improved L-shaped method for two-stage convex 0-1 mixed integer nonlinear stochastic programs. *Computers & Chemical Engineering*, 112:165 – 179, 2018a. ISSN 0098-1354.

C. Li and I. E. Grossmann. An improved L-shaped method for two-stage convex 0-1 mixed integer nonlinear stochastic programs. *Computers & Chemical Engineering*, 112:165–179, 2018b.

C. Li and I. E. Grossmann. A generalized Benders decomposition-based branch and cut algorithm for two-stage stochastic programs with nonconvex constraints and mixed-binary first and second stage variables. *Journal of Global Optimization*, 75(2):247–272, oct 2019a. ISSN 0925-5001.

C. Li and I. E. Grossmann. A finite $\epsilon$-convergence algorithm for two-stage stochastic convex nonlinear programs with mixed-binary first and second-stage variables. *Journal of Global Optimization*, 2019b.

C. Li, D. E. Bernal, K. C. Furman, M. A. Duran, and I. E. Grossmann. Sample average approximation for stochastic nonconvex mixed integer nonlinear programming via outer-approximation. *Optimization and Engineering*, pages 1–29, 2020a.

C. Li, A. J. Conejo, P. Liu, B. Omell, J. D. Siirola, and I. E. Grossmann. Mixed-integer linear programming models and algorithms for generation and transmission expansion planning of power systems. *Optimization Online*, 2020b.

P. Li, H. Arellano-Garcia, and G. Wozny. Chance constrained programming approach to process optimization under uncertainty. *Computers & Chemical Engineering*, 32(1-2): 25–45, 2008.

X. Li, E. Armagan, A. Tomasgard, and P. I. Barton. Stochastic pooling problem for natural gas production network design and operation under uncertainty. *AIChE Journal*, 57(8): 2120–2135, 2011a.

X. Li, E. Armagan, A. Tomasgard, and P. I. Barton. Stochastic pooling problem for natural gas production network design and operation under uncertainty. *AIChE Journal*, 57(8): 2120–2135, 2011b.

X. Li, A. Tomasgard, and P. I. Barton. Nonconvex generalized benders decomposition for stochastic separable mixed-integer nonlinear programs. *Journal of Optimization Theory and Applications*, 151(3):425, 2011c.

X. Li, Y. Chen, and P. I. Barton. Nonconvex generalized benders decomposition with piecewise convex relaxations for global optimization of integrated process design and operation problems. *Industrial & Engineering Chemistry Research*, 51(21):7287–7299, 2012a.

X. Li, A. Tomasgard, and P. I. Barton. Decomposition strategy for the stochastic pooling problem. *Journal of Global Optimization*, 54(4):765–790, 2012b.

C. H. Lim, J. T. Linderoth, J. R. Luedtke, and S. J. Wright. Parallelizing subgradient methods for the lagrangian dual in stochastic mixed-integer programming. *Informs Journal on Optimization*, 3(1):1–22, 2021.

R. Lima. Ibm ilog cplex-what is inside of the box? In *Proc. 2010 EWO Seminar*, pages 1–72, 2010.

J. Liu, G. Li, and S. Sen. Coupled learning enabled stochastic programming with endogenous uncertainty. *Optimization Online*, 2019.

M. L. Liu and N. V. Sahinidis. Optimization in process planning under uncertainty. *Industrial & Engineering Chemistry Research*, 35(11):4154–4165, 1996.

P. Liu, E. N. Pistikopoulos, and Z. Li. Decomposition based stochastic programming approach for polygeneration energy systems design under uncertainty. *Industrial & Engineering Chemistry Research*, 49(7):3295–3305, 2010.

Y. Liu, R. Sioshansi, and A. J. Conejo. Hierarchical clustering to find representative operating periods for capacity-expansion modeling. *IEEE Transactions on Power Systems*, 33 (3):3029–3039, 2017a.

Y. Liu, R. Sioshansi, and A. J. Conejo. Multistage stochastic investment planning with multiscale representation of uncertainties and decisions. *IEEE Transactions on Power Systems*, 33(1):781–791, 2017b.

T. Lohmann and S. Rebennack. Tailored Benders decomposition for a long-term power expansion model with short-term demand response. *Management Science*, 63(6):2027–2048, 2017.

M. Lubin and I. Dunning. Computing in operations research using julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

M. Lubin, E. Yamangil, R. Bent, and J. P. Vielma. *Extended Formulations in Mixed-Integer Convex Programming*, pages 102–113. Springer International Publishing, Cham, 2016. ISBN 978-3-319-33461-5.

X. Ma, T. Plaksina, and E. Gildin. Optimization of placement of hydraulic fracture stages in horizontal wells drilled in shale gas reservoirs. In *Unconventional Resources Technology Conference*, pages 1479–1489. Society of Exploration Geophysicists, American Association of Petroleum, 2013.

W.-K. Mak, D. P. Morton, and R. K. Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1-2):47–56, 1999.

D. S. Mallapragada, D. J. Papageorgiou, A. Venkatesh, C. L. Lara, and I. E. Grossmann. Impact of model resolution on scenario outcomes for electricity sector system expansion. *Energy*, 163:1231–1244, 2018.

Markus G. Drouven. Optimization Engineering at EQT. `https://pubsonline.informs.org/do/10.1287/orms.2018.06.07/full/`, 2018.

J. R. McFarland, A. A. Fawcett, A. C. Morris, J. M. Reilly, and P. J. Wilcoxen. Overview of the emf 32 study on us carbon tax scenarios. *Climate Change Economics*, 9(01):1840002, 2018.

D. Mejía-Giraldo and J. D. McCalley. Maximizing future flexibility in electric generation portfolios. *IEEE Transactions on Power Systems*, 29(1):279–288, 2013.

E. Mijangos. An algorithm for two-stage stochastic mixed-integer nonlinear convex problems. *Annals of Operations Research*, 235(1):581–598, 2015.

R. Misener and C. A. Floudas. Antigone: algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014.

R. Misener, J. P. Thompson, and C. A. Floudas. Apogee: Global optimization of standard, generalized, and extended pooling problems via linear and logarithmic partitioning schemes. *Computers & Chemical Engineering*, 35(5):876–892, 2011.

P. Nahmmacher, E. Schmid, L. Hirth, and B. Knopf. Carpe diem: A novel approach to select representative days for long-term power system modeling. *Energy*, 112:430–442, 2016.

L. C. Norton and I. E. Grossmann. Strategic planning model for complete process flexibility. *Industrial & Engineering Chemistry Research*, 33(1):69–76, 1994.

L. Ntaimo. Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations Research*, 58(1):229–243, 2010.

L. Ntaimo and S. Sen. The million-variable "march" for stochastic combinatorial optimization. *Journal of Global Optimization*, 32(3):385–400, 2005.

L. Ntaimo and M. W. Tanner. Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs. *Journal of Global Optimization*, 41(3):365–384, 2008.

E. Ogbe. *New rigorous decomposition methods for mixed-integer linear and nonlinear programming*. PhD thesis, Queen's University (Canada), 2016.

E. Ogbe and X. Li. A joint decomposition method for global optimization of multiscenario nonconvex mixed-integer nonlinear programs. *Journal of Global Optimization*, pages 1–35, 2018.

F. Oliveira, V. Gupta, S. Hamacher, and I. E. Grossmann. A lagrangean decomposition approach for oil supply chain investment planning under uncertainty with risk considerations. *Computers & Chemical Engineering*, 50:184–195, 2013.

A. Ondeck, M. Drouven, N. Blandino, and I. E. Grossmann. Multi-operational planning of shale gas pad development. *Computers & Chemical Engineering*, 2019. ISSN 0098-1354. doi: https://doi.org/10.1016/j.compchemeng.2019.03.035. URL `http://www.sciencedirect.com/science/article/pii/S0098135418307920`.

R. P. O'Neill, E. A. Krall, K. W. Hedman, and S. S. Oren. A model and approach to the challenge posed by optimal power systems planning. *Mathematical Programming*, 140(2):239–266, 2013. ISSN 1436-4646. doi: 10.1007/s10107-013-0695-3. URL `http://dx.doi.org/10.1007/s10107-013-0695-3`.

V. Oree, S. Z. S. Hassen, and P. J. Fleming. Generation expansion planning optimisation with renewable energy integration: A review. *Renewable and Sustainable Energy Reviews*, 69:790–803, 2017.

B. Palmintier and M. Webster. Impact of unit commitment constraints on generation expansion planning with renewables. In *2011 IEEE power and energy society general meeting*, pages 1–7. IEEE, 2011.

H.-S. Park and C.-H. Jun. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications*, 36(2):3336–3341, 2009.

M. Park, S. Park, F. D. Mele, and I. E. Grossmann. Modeling of purchase and sales contracts in supply chain optimization. *Industrial & Engineering Chemistry Research*, 45(14):5013–5026, 2006.

G. Paules IV and C. Floudas. Stochastic programming in process synthesis: a two-stage model with minlp recourse for multiperiod heat-integrated distillation sequences. *Computers & Chemical Engineering*, 16(3):189–210, 1992.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

M. V. Pereira and L. M. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375, 1991.

S. Pfenninger. Dealing with multiple decades of hourly wind and pv time series in energy models: A comparison of methods to reduce time resolution and the planning implications of inter-annual variability. *Applied Energy*, 197:1–13, 2017.

G. C. Pflug and A. Pichler. Time-consistent decisions and temporal decomposition of coherent risk functionals. *Mathematics of Operations Research*, 41(2):682–699, 2016.

A. Pina, C. A. Silva, and P. Ferrão. High-resolution modeling framework for planning electricity systems with high penetration of renewables. *Applied Energy*, 112:215 – 223, 2013. ISSN 0306-2619.

S. Pineda and A. Conejo. Scenario reduction for risk-averse electricity trading. *IET Generation, Transmission & Distribution*, 4(6):694–705, 2010.

E. Pistikopoulos and M. Ierapetritou. Novel approach for optimal process design under uncertainty. *Computers & Chemical Engineering*, 19(10):1089–1110, 1995.

K. Poncelet, E. Delarue, J. Duerinck, D. Six, and W. D'haeseleer. The importance of integrating the variability of renewables in long-term energy planning models. *KU Leuven*, pages 1–18, October 2014.

K. Poncelet, H. Höschle, E. Delarue, A. Virag, and W. D' haeseleer. Selecting representative days for capturing the implications of integrating intermittent renewables in generation expansion planning problems. *IEEE Transactions on Power Systems*, 32(3):1936–1948, 2016.

W. B. Powell. A unified framework for optimization under uncertainty. In *Optimization Challenges in Complex, Networked and Risky Systems*, pages 45–83. INFORMS, 2016.

W. B. Powell. A unified framework for stochastic optimization. *European Journal of Operational Research*, 275(3):795–821, 2019.

D. Pozo, E. E. Sauma, and J. Contreras. A three-level static MILP model for generation and transmission expansion planning. *IEEE Transactions on Power systems*, 28(1):202–210, 2012.

D. Pozo, J. Contreras, and E. E. Sauma. Unit commitment with ideal and generic energy storage units. *IEEE Transactions on Power Systems*, 29(6):2974–2984, Nov 2014. ISSN 0885-8950. doi: 10.1109/TPWRS.2014.2313513.

Y. Qi and S. Sen. The ancestral Benders cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming. *Mathematical Programming*, 161(1-2):193–235, 2017.

R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817, 2016. ISSN 0377-2217.

S. Rebennack, J. Kallrath, and P. M. Pardalos. Optimal storage design for a multi-product plant: A non-convex minlp formulation. *Computers & Chemical Engineering*, 35(2):255–271, 2011.

R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147, 1991.

J. P. Ruiz and I. E. Grossmann. A hierarchy of relaxations for nonlinear convex generalized disjunctive programming. *European Journal of Operational Research*, 218(1):38–47, 2012.

A. Ruszczyński. Decomposition methods in stochastic programming. *Mathematical Programming*, 79(1-3):333–353, 1997.

A. Ruszczyński. Risk-averse dynamic programming for markov decision processes. *Mathematical programming*, 125(2):235–261, 2010.

S. M. Ryan, R. J.-B. Wets, D. L. Woodruff, C. Silva-Monroy, and J.-P. Watson. Toward scalable, parallel progressive hedging for stochastic unit commitment. In *Power and Energy Society General Meeting (PES), 2013 IEEE*, pages 1–5. IEEE, 2013.

H. Sadeghi, M. Rashidinejad, and A. Abdollahi. A comprehensive sequential review study through the generation expansion planning. *Renewable and Sustainable Energy Reviews*, 67:1369–1394, 2017.

G. Sand and S. Engell. Modeling and solving real-time scheduling problems by stochastic integer programming. *Computers & Chemical Engineering*, 28(6-7):1087–1103, 2004.

R. Sargent. Process systems engineering: A retrospective view with questions for the future. *Computers & Chemical Engineering*, 29(6):1237–1241, 2005.

N. Sawaya. *Reformulations, relaxations and cutting planes for generalized disjunctive programming*. PhD thesis, Carnegie Mellon University, 2006.

O. Schmidt, A. Hawkes, A. Gambhir, and I. Staffell. The future cost of electrical energy storage based on experience rates. *Nature Energy*, 2(17110):1–1, 2017. doi: 10.1038/nenergy.2017.110.

R. Schultz. On structure and stability in stochastic programs with random technology matrix and complete integer recourse. *Mathematical Programming*, 70(1-3):73–89, 1995.

I. J. Scott, P. M. Carvalho, A. Botterud, and C. A. Silva. Clustering representative days for power systems generation expansion planning: Capturing the effects of variable renewables and energy storage. *Applied Energy*, 253:113603, 2019.

P. Seljom and A. Tomasgard. Sample average approximation and stability tests applied to energy system design. *Energy Systems*, pages 1–25, 2019.

S. Sen and Y. Deng. Learning enabled optimization: Towards a fusion of statistical learning and stochastic programming. *INFORMS Journal on Optimization (submitted)*, 2018.

S. Sen and H. D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223, 2006.

M. Sengupta, Y. Xie, A. Lopez, A. Habte, G. Maclaurin, and J. Shelby. The national solar radiation data base (nsrdb). *Renewable and Sustainable Energy Reviews*, 89:51–60, 2018.

A. Shapiro. Asymptotic analysis of stochastic programs. *Annals of Operations Research*, 30 (1):169–186, 1991.

A. Shapiro. Asymptotic behavior of optimal solutions in stochastic programming. *Mathematics of Operations Research*, 18(4):829–845, 1993.

A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. MOS-SIAM Series on Optimization, 2014.

H. D. Sherali and W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

H. D. Sherali and B. M. Fraticelli. A modification of Benders decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22(1-4):319–342, 2002.

H. D. Sherali and X. Zhu. On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. *Mathematical Programming*, 108(2):597–616, 2006.

W. Short, P. Sullivan, T. Mai, M. Mowers, C. Uriarte, N. Blair, D. Heimiller, and A. Martinez. Regional energy deployment system (reeds). Technical Report December, National Renewable Technology Laboratory (NREL), 2011. URL `https://www.nrel.gov/analysis/reeds/pdfs/reeds_documentation.pdf`.

A. Shortt and M. O'Malley. Impact of variable generation in generation resource planning models. In *IEEE PES General Meeting*, pages 1–6, July 2010. doi: 10.1109/PES.2010.5589461.

J. D. Siirola. Pyomo: Introduction & IDAES development. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2017.

R. A. Stubbs and S. Mehrotra. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86(3):515–532, 1999.

M. Sun, F. Teng, X. Zhang, G. Strbac, and D. Pudjianto. Data-driven representative day selection for investment decisions: A cost-oriented approach. *IEEE Transactions on Power Systems*, 34(4):2925–2936, 2019.

B. Tarhan and I. E. Grossmann. A multistage stochastic programming approach with strategies for uncertainty reduction in the synthesis of process networks with uncertain yields. *Computers & Chemical Engineering*, 32(4-5):766–788, 2008.

M. Tawarmalani and N. V. Sahinidis. *The Pooling Problem*. Springer US, Boston, MA, 2002. ISBN 978-1-4757-3532-1.

M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.

H. Teichgraeber and A. R. Brandt. Clustering methods to find representative periods for the optimization of energy systems: An initial framework and comparison. *Applied Energy*, 239:1283–1293, 2019.

H. Teichgraeber, C. P. Lindenmeyer, N. Baumgärtner, L. Kotzur, D. Stolten, M. Robinius, A. Bardow, and A. R. Brandt. Extreme events in time series aggregation: A case study for optimal residential energy supply systems. *arXiv preprint arXiv:2002.03059*, 2020.

J. J. Torres, C. Li, R. M. Apap, and I. E. Grossmann. A review on the performance of linear and mixed integer two-stage stochastic programming algorithms and software. *Optimization Online*, 2019.

W. W. Tso, C. D. Demirhan, C. F. Heuberger, J. B. Powell, and E. N. Pistikopoulos. A hierarchical clustering decomposition algorithm for optimizing renewable power systems with storage. *Applied Energy*, 270:115190, 2020.

N. G. Ude, H. Yskandar, and R. C. Graham. A comprehensive state-of-the-art survey on the transmission network expansion planning optimization algorithms. *IEEE Access*, 7: 123158–123181, 2019.

U.S. Energy Information Administration. Annual Energy Outlook 2019. `https://www.eia.gov/outlooks/aeo/pdf/aeo2019.pdf`, 2019a.

U.S. Energy Information Administration. Henry Hub Natural Gas Spot Price. `https://www.eia.gov/dnav/ng/hist/rngwhhdD.htm`, 2019b.

R. M. Van Slyke and R. Wets. L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663, 1969.

J. Viswanathan and I. E. Grossmann. A combined penalty function and outer-approximation method for MINLP optimization. *Computers & Chemical Engineering*, 14(7):769–782, 1990.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1): 25–57, 2006.

S. W. Wallace and W. T. Ziemba. *Applications of stochastic programming*. SIAM, 2005.

J.-P. Watson and D. L. Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8(4): 355–370, 2011.

J.-P. Watson, D. L. Woodruff, and W. E. Hart. PySP: modeling and solving stochastic programs in Python. *Mathematical Programming Computation*, 4(2):109–149, 2012.

J. Wei and M. J. Realff. Sample average approximation methods for stochastic MINLPs. *Computers & Chemical Engineering*, 28(3):333–346, 2004.

T. Westerlund and K. Lundqvist. *Alpha-ECP, version 5.01: An interactive MINLP-solver based on the extended cutting plane method*. Åbo Akademi, 2001.

K. C. Wilson and L. J. Durlofsky. Optimization of shale gas field development using direct search techniques and reduced-physics models. *Journal of Petroleum Science and Engineering*, 108:304–315, 2013.

Z. Wu, Y. Liu, W. Gu, Y. Wang, and C. Chen. Contingency-constrained robust transmission expansion planning under uncertainty. *International Journal of Electrical Power & Energy Systems*, 101:331–338, 2018.

L. Yang, I. E. Grossmann, and J. Manno. Optimization models for shale gas water management. *AIChE Journal*, 60(10):3490–3501, 2014.

Y. Yang and P. I. Barton. Integrated crude selection and refinery optimization under uncertainty. *AIChE journal*, 62(4):1038–1053, 2016.

Y. Ye, J. Li, Z. Li, Q. Tang, X. Xiao, and C. A. Floudas. Robust optimization and stochastic programming approaches for medium-term production scheduling of a large-scale steelmaking continuous casting process under demand uncertainty. *Computers & Chemical Engineering*, 66:165–185, 2014.

A. Yeganefar, M. R. Amin-Naseri, and M. K. Sheikh-El-Eslami. Improvement of representative days selection in power system planning by incorporating the extreme days of the net load to take account of the variability and intermittency of renewable resources. *Applied Energy*, 272:115224, 2020.

W. Yu and K. Sepehrnoori. Optimization of multiple hydraulically fractured horizontal wells in unconventional gas reservoirs. *Journal of Petroleum Engineering*, 2013, 2013.

V. M. Zavala. Stochastic optimal control model for natural gas networks. *Computers & Chemical Engineering*, 64:103–113, 2014.

L. J. Zeballos, C. A. Méndez, A. P. Barbosa-Povoa, and A. Q. Novais. Multi-period design and planning of closed-loop supply chains with uncertain supply and demand. *Computers & Chemical Engineering*, 66:151–164, 2014.

Z. Zeng and S. Cremaschi. Artificial lift infrastructure planning for shale gas producing horizontal wells. *Proceedings of the FOCAPO/CPC, Tuscan, AZ, USA*, pages 8–12, 2017.

Z. Zeng and S. Cremaschi. Multistage stochastic models for shale gas artificial lift infrastructure planning. In *Computer Aided Chemical Engineering*, volume 44, pages 1285–1290. Elsevier, 2018.

Z. Zeng, B. Christian, and S. Cremaschi. A generalized knapsack-problem based decomposition heuristic for solving multistage stochastic programs with endogenous and/or exogenous uncertainties. *Industrial & Engineering Chemistry Research*, 57(28):9185–9199, 2018.

H. Zhang, G. T. Heydt, V. Vittal, and J. Quintero. An improved network model for transmission expansion planning considering reactive power and network losses. *IEEE Transactions on Power Systems*, 28(3):3471–3479, 2013.

M. Zhang and S. Küçükyavuz. Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs. *SIAM Journal on Optimization*, 24(4):1933–1951, 2014.

Q. Zhang, J. L. Cremer, I. E. Grossmann, A. Sundaramoorthy, and J. M. Pinto. Risk-based integrated production scheduling and electricity procurement for continuous power-intensive processes. *Computers & Chemical Engineering*, 86:90–105, 2016.

Y. Zhu and T. Kuno. A disjunctive cutting-plane-based branch-and-cut algorithm for 0- 1 mixed-integer convex nonlinear programs. *Industrial & Engineering Chemistry Research*, 45(1):187–196, 2006.

J. Zou, S. Ahmed, and X. A. Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, 2018a. ISSN 1436-4646.

J. Zou, S. Ahmed, and X. A. Sun. Multistage stochastic unit commitment using stochastic dual dynamic integer programming. *IEEE Transactions on Power Systems*, 34(3):1814–1823, 2018b.

J. Zou, S. Ahmed, and X. A. Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, 175(1-2):461–502, 2019.

# Appendix A

# Chapter 3 supplementary material

## A.1 Proofs of the theorems

**Proposition 4.** $\eta_\omega \geq z^{*k}_{SL,\omega} - \mu^k_\omega x$ is valid for the Benders master problem.

*Proof.* We rewrite the full space problem $(P)$ as $(P')$ by introducing variables $\eta_\omega$.

$$(P'): \quad \min \quad z = \sum_{\omega \in \Omega} \eta_\omega \tag{A.1}$$

$$s.t. \quad A_0 x \geq b_0, \quad g_0(x) \leq 0 \tag{A.2}$$

$$A_{1,\omega} x + g_{1,\omega}(y_\omega) \leq b_{1,\omega} \quad \forall \omega \in \Omega \tag{A.3}$$

$$g_{2,\omega}(y_\omega) \leq b_{2,\omega} \quad \forall \omega \in \Omega \tag{A.4}$$

$$\eta_\omega \geq \tau_\omega (c^T x + d^T_\omega y_\omega) \quad \forall \omega \in \Omega \tag{A.5}$$

$$x \in X \tag{A.6}$$

$$y_\omega \in Y \quad \forall \omega \in \Omega \tag{A.7}$$

It is easy to see (P) and (P') are equivalent. The idea of Benders decomposition is to project the feasible region that include $(x, \eta_\omega, y_\omega)$ onto the $(x, \eta_\omega)$ space. Therefore, valid constraints for Benders master problem can be derived from (A.2)-(A.7), which involve only variables $x, \eta_\omega$. $\tau_\omega(c^T x + d_\omega^T y_\omega) + \mu^k x$ is the objective function of Lagrangean subproblem$(SL_\omega^k)$. This objective function is minimized over constraints (A.2)-(A.4),(A.6),(A.7) in $(SL_\omega^k)$. Thus, $z_{SL,\omega}^{*k} \leq \tau_\omega(c^T x + d_\omega^T y_\omega) + \mu^k x$, is a valid inequality if we have (A.2)-(A.4),(A.6),(A.7). By combining constraint (A.5), we can show that $\eta_\omega \geq z_{SL,\omega}^{*k} - \mu_\omega^k x$ is a valid inequality for the Benders master problem.

□

**Proposition 5.** The Benders master problem with the Lagrangean cuts (3.2c) yields a lower bound that is at least as tight as using Lagrangean decomposition.

*Proof.* In Lagrangean decomposition, the best lower bound is given by solving problem $(D)$

$$\begin{aligned} (D): \quad & \max_{\mu_\omega, \forall \omega \in \Omega} \quad \min_{x_\omega, y_\omega, \forall \omega \in \Omega} \quad \sum_\omega \tau_\omega(c^T x_\omega + d_\omega^T y_\omega) + \mu_\omega x_\omega \\ & s.t. \quad (3.3b) - (3.3f) \quad \forall \omega \in \Omega \end{aligned} \tag{A.8}$$

Let $\mu_\omega^*$ be the optimal dual multiplier to problem $(D)$. Let $x_\omega^*, y_\omega^*$ be the optimal primal solution when $\mu_\omega$ is fixed at $\mu_\omega^*$. Let $z_\omega^{D*} = \tau_\omega(c^T x_\omega^* + d_\omega^T y_\omega^*) + \mu_\omega^* x_\omega^*$. In the proposed algorithm, the Lagrangean cuts $\eta_\omega \geq z_\omega^{D*} - \mu_\omega^* x$, $\forall \omega \in \Omega$ are added to the Benders master problem. $\sum_{\omega \in \Omega} \eta_\omega \geq \sum_{\omega \in \Omega} (z_\omega^{D*} - \mu_\omega^* x) = \sum_{\omega \in \Omega} z_\omega^{D*}$ is implied by the Lagrangean cuts. Therefore, the lower bound obtained from the Benders master problem is at least as tight as the lower bound obtained from Lagrangean decomposition. □

## A.2 Subgradient method

We briefly describe the subgradient method (Oliveira et al., 2013) to update the multipliers. The nonanticipativity constraints are written as

$$x_{\omega 1} = x_{\omega 2}, x_{\omega 1} = x_{\omega 3}, \cdots, x_{\omega 1} = x_{\omega |\Omega|} \tag{A.9}$$

We define the multipliers associated with the NACs as $\pi_\omega, \omega = \omega_1, \omega_2 \cdots, \omega_{|\Omega|-1}$ The multipliers are updated from the formula:

$$\pi_\omega^{k+1} = \pi_\omega^k + t^k(\hat{x}_{\omega 1}^k - \hat{x}_{\omega+1}^k) \quad \omega = \omega_1, \omega_2, \cdots, \omega_{|\Omega|-1} \tag{A.10}$$

where $\hat{x}_\omega^k$ is the optimal solution to the Lagrangean subproblem $SL_\omega^k$. The value of $t^k$ is calculated by the formula:

$$t_k = \frac{\alpha_k(UB - LB)}{\sum_{\omega=\omega_1}^{\omega_{|\Omega|-1}} ||\hat{x}_{\omega 1}^k - \hat{x}_{\omega+1}^k||_2^2} \tag{A.11}$$

$\alpha_k$ is a scalar chosen between 0 and 2. The value of $\alpha_k$ is divided by a constant greater than 1 when $\sum_\omega z_{SL,\omega}^{*k}$ fails to improve.

# Appendix B

# Chapter 4 supplementary material

## B.1 Appendix 1: Benders cuts for the illustrative example

The Benders cuts that are generated in each node of the branch-and-bound tree for the illustrative example in section 4.3 are shown in Tables B.1-B.3.

## B.2 Appendix 2: Constrained layout problem under price uncertainty

The constrained layout problem is adapted from the PhD thesis of Sawaya Sawaya (2006). In this problem, we need to decide the layout of some units represented by rectangles with known lengths and widths. The units have to be manufactured before we install them at certain locations. In the manufacturing process, we need to provide the relative position of the units in order to produce the pipes that connect those units. The first-stage decisions include the designed relative position of the units. After the units are manufactured, they can be installed at several locations. In order to install some unit(s) at a given location, we need to purchase a circled area around the center of that location. The rectangle(s) installed

Table B.1: Benders cuts that are generated at the root node

| Iteration | Benders cuts for $\omega_1$ | Benders cuts for $\omega_2$ |
|---|---|---|
| 1 | $\eta_{\omega_1} \geq -218.536x_1 - 144x_2 - 16.1135x_3 - 16.1135x_4 + 73.5$ | $\eta_{\omega_2} \geq -244.5x_1 - 160x_2 - 18.0179x_3 - 18.0179x_4 + 98.5$ |
| 2 | $\eta_{\omega_1} \geq 0.497x_1 - 3670x_2 + 3790.08x_3 - 3782.6x_4 - 3715.16$ | $\eta_{\omega_2} \geq 0.496464x_1 - 3150x_2 + 3245.59x_3 - 3237.48x_4 - 3145.69$ |
| 3 | $\eta_{\omega_1} \geq -7010x_1 + 0.49604x_2 - 14326.1x_3 + 14672.1x_4 - 14597.2$ | $\eta_{\omega_2} \geq -13100x_1 + 0.495977x_2 - 14554.5x_3 + 13995.4x_4 - 13895.5$ |
| 4 | $\eta_{\omega_1} \geq 0.496x_1 - 33738.8x_2 + 3526.15x_3 + 3525.62x_4 - 6975.37$ | $\eta_{\omega_2} \geq 0.499x_1 - 42390.2x_2 + 9538.87x_3 + 9537.98x_4 - 18975.4$ |
| 5 | $\eta_{\omega_1} \geq 0.499992x_1 - 541.837x_2 + 62.1838x_3 + 62.2324x_4 - 50.4293$ | $\eta_{\omega_2} \geq 0.499999x_1 - 541.843x_2 + 105.178x_3 + 111.603x_4 - 117.794$ |
| 6 | $\eta_{\omega_1} \geq -49x_1 - 55.5x_2 + 9.52529x_3 + 9.51962x_4 + 57.4551$ | $\eta_{\omega_2} \geq -49x_1 - 55.5x_2 + 8.33968x_3 + 12.4331x_4 + 80.7273$ |
| 7 | $\eta_{\omega_1} \geq 0.0000172x_1 + 7.79833x_2 + 10.0592x_3 + 10.0902x_4 - 38.3289$ | $\eta_{\omega_2} \geq -0.0291935x_1 + 2.37213x_2 + 4.42782x_3 + 4.47516x_4 - 16.1181$ |
| 8 | $\eta_{\omega_1} \geq 0.5x_1 - 122.173x_2 + 27.6631x_3 + 22.5481x_4 + 9.92204$ | $\eta_{\omega_2} \geq 0.5x_1 - 122.173x_2 + 22.0681x_3 + 13.3055x_4 + 49.7594$ |
| 9 | $\eta_{\omega_1} \geq 0.5x_1 - 6x_2 + 3.00259x_3 + 2.99245x_4 - 3.74505$ | $\eta_{\omega_2} \geq 0.5x_1 - 68.4353x_2 + 15.7236x_3 + 15.6932x_4 + 31.8559$ |
| 10 | $\eta_{\omega_1} \geq -85.8x_1 + 0.5x_2 + 11.6499x_3 + 11.8356x_4 + 22.7981$ | $\eta_{\omega_2} \geq -85.7842x_1 + 0.5x_2 + 11.6177x_3 + 14.9939x_4 + 44.672$ |
| 11 | $\eta_{\omega_1} \geq 0.0000169x_1 + 0.5x_2 + 3.94807x_3 + 3.95063x_4 - 11.4815$ | $\eta_{\omega_2} \geq -55.6337x_1 + 0.5x_2 + 10.0374x_3 + 8.64236x_4 + 33.0624$ |
| 12 | $\eta_{\omega_1} \geq 0.5x_1 - 0.999963x_2 + 3.61146x_3 + 3.58604x_4 - 9.72527$ | $\eta_{\omega_2} \geq 0.5x_1 - 1.02808x_2 + 3.42646x_3 + 3.4266x_4 - 9.32503$ |
| 13 | $\eta_{\omega_1} \geq 0.0000171x_1 + 0.5x_2 + 3.86146x_3 + 4.98608x_4 - 12.4304$ | $\eta_{\omega_2} \geq -0.0292157x_1 + 0.5x_2 + 4.01615x_3 + 4.15695x_4 - 11.644$ |
| 14 | $\eta_{\omega_1} \geq 0.268432x_1 - 0.305251x_2 + 3.73107x_3 + 3.72861x_4 - 10.4742$ | $\eta_{\omega_2} \geq -49.3943x_1 + 0.5x_2 + 11.471x_3 + 18.4473x_4 + 15.9746$ |
| 15 | $\eta_{\omega_1} \geq 0.38848x_1 - 0.665399x_2 + 3.69908x_3 + 3.69815x_4 - 10.1589$ | $\eta_{\omega_2} \geq -49.001x_1 + 0.5x_2 + 12.1317x_3 + 13.1548x_4 + 20.2145$ |

Table B.2: Benders cuts that are generated at node 1

| Iteration | Benders cuts for $\omega_1$ | Benders cuts for $\omega_2$ |
|---|---|---|
| 1 | $\eta_{\omega_1} \geq -23.8893x_1 + 0.5x_2 + 1.96244x_3 + 3.67593x_4 + 14.7302$ | $\eta_{\omega_2} \geq -30.2523x_1 + 0.499996x_2 + 4.9294x_3 + 4.93241x_4 + 16.8905$ |
| 2 | $\eta_{\omega_1} \geq -0.0221633x_1 + 0.5x_2 + 3.1919x_3 + 3.19356x_4 - 9.88416$ | $\eta_{\omega_2} \geq -0.030201x_1 + 0.499998x_2 + 3.28529x_3 + 3.28533x_4 - 10.0405$ |
| 3 | $\eta_{\omega_1} \geq -1.2564x_1 + 0.499999x_2 + 3.02764x_3 + 3.03329x_4 - 8.32534$ | $\eta_{\omega_2} \geq -1.61672x_1 + 0.491218x_2 + 3.46615x_3 + 3.52462x_4 - 8.86527$ |

Table B.3: Benders cuts that are generated at node 2

| Iteration | Benders cuts for $\omega_1$ | Benders cuts for $\omega_2$ |
|---|---|---|
| 1 | $\eta_{\omega_1} \geq 6.41636x_1 + 0.499997x_2 + 5.16831x_3 + 5.17416x_4 - 20.2796$ | $\eta_{\omega_2} \geq 5.9259x_1 + 0.499993x_2 + 5.16145x_3 + 5.1623x_4 - 19.7497$ |
| 2 | $\eta_{\omega_1} \geq 0.631399x_1 + 0.499976x_2 + 5.06953x_3 + 5.12023x_4 - 14.3419$ | $\eta_{\omega_2} \geq -0.299457x_1 - 9.43408x_2 + 4.11646x_3 + 4.11753x_4 - 1.50045$ |

at that location has to be constrained within the circle that is purchased. The price of each area is uncertain and the designed units are installed according to the prices.

The sets, parameters, and variables that are used in the model is defined as follows:

**Sets**

$i \in N$ = rectangles (units)

$q \in Q$ = areas

$\omega \in \Omega$ = scenarios

**Parameters**

$L_i$ = length of rectangle $i$

$H_i$ = width of rectangle $i$

$xbar_q$ = the $x$ coordinate of area $q$

$ybar_q$ = the $y$ coordinate of area $q$

$\tau_\omega$ = probability of scenario $\omega$

$d_{q\omega}$ = unit price of area $q$ in scenario $\omega$

**first-stage decisions**

$x_i$ = the designed $x$ coordinate of the center of rectangle $i$

$y_i$ = the designed $y$ coordinate of the center of rectangle $i$

$delx_{ij}$ = the designed distance of the center of rectangle $i$ and the center of rectangle $j$ along the $x$ axis

$dely_{ij}$ = the designed distance of the center of rectangle $i$ and the center of rectangle $j$ along the $y$ axis

$Z^1_{ij}$, $Z^2_{ij}$, $Z^3_{ij}$, $Z^4_{ij} \in \{True, False\}$ decides the relative position of rectangle $i$ and rectangle $j$

**second-stage decisions**

$xs_{i\omega}$ = the installed $x$ coordinate of the center of rectangle $i$ in scenario $\omega$

$ys_{i\omega}$ = the installed $y$ coordinate of the center of rectangle $i$ in scenario $\omega$

$W_{qi\omega} = \{True, False\}$ whether to position rectangle $i$ in area $q$ or not

$S_{q\omega}$ = the size of area $q$ that is purchased in scenario $\omega$

Constraint (B.2) relates the designed relative distance of rectangle $i$ and rectangle $j$ in both $x$ and $y$ direction to the designed $x$ and $y$ coordinate of rectangle $i$ and rectangle $j$. There are variable costs associated with the relative designed distances, which corresponds to the pipes that connect those units. Constraint (B.3) prevents rectangle $i$ and rectangle $j$ from overlapping with each other. After the design decisions are made, the actual position to install each unit $i \in N$ are decided. Constraint (B.4) enforces that the actual installed positions of rectangle $i$ and rectangle $j$ in each scenario $\omega \in \Omega$ have to coincide with the designed relative position. Moreover, in each scenario, each rectangle $i$ has to be constrained in exactly one circle $q \in Q$ that is purchased, which is enforced by constraint (B.5). The objective also includes the expected cost of the purchase of circles. Note that the problem is expressed as a GDP and can be reformulated with big-M or hull reformulation (see Table 7 in Grossmann and Trespalacios (2013)) as a convex MINLP, which is a two-stage stochastic program with mixed integer variables in both first and second stage.

$$\min \quad \sum_{i} \sum_{j,j>i} c_{ij}(delx_{ij} + dely_{ij}) + \sum_{\omega \in \Omega} \tau_\omega \sum_{q \in Q} (d_{q\omega} S_{q\omega}) \tag{B.1}$$

*s.t.*

$$\begin{aligned} &delx_{ij} \geq x_i - x_j \quad \forall i,j \in N, i<j \\ &delx_{ij} \geq x_j - x_i \quad \forall i,j \in N, i<j \\ &dely_{ij} \geq y_i - y_j \quad \forall i,j \in N, i<j \\ &dely_{ij} \geq y_j - y_i \quad \forall i,j \in N, i<j \end{aligned} \tag{B.2}$$

$$\begin{bmatrix} Z_{ij}^1 \\ x_i + L_i/2 \leq x_j - L_j/2 \end{bmatrix} \vee \begin{bmatrix} Z_{ij}^2 \\ x_j + L_j/2 \leq x_i - L_i/2 \end{bmatrix} \vee \begin{bmatrix} Z_{ij}^3 \\ y_i + H_i/2 \leq y_j - H_j/2 \end{bmatrix} \vee \begin{bmatrix} Z_{ij}^4 \\ y_j + H_j/2 \leq y_i - H_i \end{bmatrix}$$

$$\forall i,j \in N, i<j \tag{B.3}$$

$$
\begin{aligned}
&xs_{i\omega} - xs_{j\omega} = x_i - x_j \quad \forall i, j \in N, i < j, \omega \in \Omega \\
&ys_{i\omega} - ys_{j\omega} = y_i - y_j \quad \forall i, j \in N, i < j, \omega \in \Omega
\end{aligned}
\tag{B.4}
$$

$$
\vee_{q \in Q} \begin{bmatrix}
W_{qi\omega} \\
(xs_{i\omega} - L_i/2 - xbar_q)^2 + (ys_{i\omega} + H_i/2 - ybar_q)^2 \leq S_{q\omega} \\
(xs_{i\omega} - L_i/2 - xbar_q)^2 + (ys_{i\omega} - H_i/2 - ybar_q)^2 \leq S_{q\omega} \\
(xs_{i\omega} + L_i/2 - xbar_q)^2 + (ys_{i\omega} + H_i/2 - ybar_q)^2 \leq S_{q\omega} \\
(xs_{i\omega} + L_i/2 - xbar_q)^2 + (ys_{i\omega} - H_i/2 - ybar_q)^2 \leq S_{q\omega}
\end{bmatrix} \quad \forall i \in N, \omega \in \Omega \tag{B.5}
$$

$$
delx_{ij}, dely_{ij}, S_{q\omega} \in \mathbb{R}^1_+, Z^1_{ij}, Z^2_{ij}, Z^3_{ij}, Z^4_{ij}, W_{qi\omega} \in \{True, False\} \quad \forall i, j \in N, i < j, q \in Q, \omega \in \Omega \tag{B.6}
$$

# Appendix C

# Chapter 5 supplementary material

## C.1 Appendix 1: Mathematical Formulation of Stochastic Pooling Problem with Contract Selection

**Sets**

$i$=feeds

$j$=products

$l$=pools

$p$=qualities

$c$=contracts

$\omega$=scenarios

$T_X$=$(i, l)$ pairs for which input to pool connection allowed

$T_Y$= $(l, j)$ pairs for which pool to output connection allowed

$T_Z$=$(i, j)$ pairs for which input to output connection allowed

**Parameters**

$A_i^U$=maximum available flow

$A_i^L$=minimum available flow

$S_l^U$=maximum pool size

$S_l^L$=minimum pool size

$D_j^U$=maximum product demand

$C_{ik}$=feed concentration

$P_{jk}^U$=maximum allowable product concentration

$P_{jk}^L$=minimum allowable product concentration

$\alpha_l$=fixed cost parameter for pools

$\beta_l$=variable cost parameter for pools

$\delta_i$=fixed cost for feed storage

$\xi_i$=variable cost for feed storage

$\tau_\omega$=probability of scenario $\omega$

$\psi_{i\omega}^c$=unit price for feed $i$ under contract $c$ in scenario $\omega$

$d_{j\omega}$=product unit price in scenario $\omega$

$\sigma_i^c$=minimum purchasable amount of feed $i$ under contract $c$

**First stage variables**

binary variables:

$\lambda_i$=whether feed $i$ exists

$\theta_l$=whether pool $l$ exists

continuous variables:

$S_l$=capacity of pool $l$

$A_i$=capacity of pool $i$

**Second stage variables**

binary variables:

$u_{i\omega}^c$=whether contract $c$ is selected in purchasing feed $i$ in scenario $\omega$

continuous variables:

$y_{lj\omega}$=flow from pool $l$ to product $j$ in scenario $\omega$

$z_{ij\omega}$=flow of feed $i$ to product $j$ in scenario $\omega$

$q_{il\omega}$=proportion of flow from input $i$ to pool $l$ in scenario $\omega$

$CT_{i\omega}$=cost of purchasing feed $i$ in scenario $\omega$

$CT^c_{i\omega}$=cost of purchasing feed $i$ under contract $c$ in scenario $\omega$

$B^c_{i\omega}$=the amount of feed $i$ purchased under contract $c$ in scenario $\omega$

We study pooling problem with purchase contracts under price uncertainty. The first stage decisions include whether to select the feeds and pools, the capacity of the feeds and pools, whether the feeds exist.

$$A^L_i \lambda_i \leq A_i \leq A^U_i \lambda_i \quad \forall i \tag{C.1}$$

$$S^L_l \theta_l \leq S_l \leq S^U_l \theta_l \quad \forall l \tag{C.2}$$

After the pools are selected, the uncertainties in the prices of the feeds and products are realized. We need to decide the material flows just as in a standard pooling problem. Equations (C.3)-(C.11) are the equations in the pq-formulation of the pooling problem.

$$\sum_{l:(i,l)\in T_X} \sum_{j:(l,j)\in T_Y} q_{il\omega} y_{lj\omega} + \sum_{j:(i,j)\in T_Z} z_{ij\omega} \leq A_i \quad \forall i, \omega \tag{C.3}$$

$$\sum_{j:(l,j)\in T_Y} y_{lj\omega} \leq S_l \quad \forall l, \omega \tag{C.4}$$

$$\sum_{l:(l,j)\in T_Y} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} z_{ij\omega} \leq D^U_{j\omega} \quad \forall j, \omega \tag{C.5}$$

$$\sum_{i:(i,l)\in T_X} q_{il\omega} = \theta_l \quad \forall i, \omega \tag{C.6}$$

$$\begin{aligned} P^L_{jk}\Big( \sum_{l:(l,j)\in T_Y} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} z_{ij\omega}\Big) \leq \sum_{l:(l,j)\in T_Y} \sum_{i:(i,l)\in T_X} C_{ik} q_{il\omega} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} C_{ik} z_{ij\omega} \\ \leq P^U_{jk}\Big( \sum_{l:(l,j)\in T_Y} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} z_{ij\omega}\Big) \quad \forall j, k, \omega \end{aligned} \tag{C.7}$$

$$0 \leq q_{il\omega} \leq \lambda_i \quad \forall (i,l) \in T_X, \omega \tag{C.8}$$

$$0 \leq y_{lj\omega} \leq \min\{S_l, D_j^U, \sum_{i:(i,l)\in T_X} A_i^U\} \quad \forall (l,j) \in T_Y \tag{C.9}$$

$$0 \leq z_{ij\omega} \leq \min\{A_i^U, D_j^U\} \quad \forall (i,j) \in T_Z, \omega \tag{C.10}$$

$$\sum_{i:(i,l)\in T_X} q_{il\omega} y_{lj\omega} = y_{lj\omega} \quad \forall l, j, \omega \tag{C.11}$$

In this case study, we also consider different types of purchase contracts when the prices are realized. The formulation of contracts is based Park et al. Park et al. (2006). (C.12)-(C.15) decides that the total amount of feed $i$ purchased can come from exactly one contract.

$$\sum_{l:(i,l)\in T_X} \sum_{j:(l,j)\in T_Y} q_{il\omega} y_{lj\omega} + \sum_{j:(i,j)\in T_Z} z_{ij\omega} = \sum_c B_{i\omega}^c \quad \forall i, \omega \tag{C.12}$$

$$A_i^L u_{i\omega}^c \leq B_{i\omega}^c \leq A_i^U u_{i\omega}^c \quad \forall i, \omega, c \tag{C.13}$$

$$\sum_c u_{i\omega}^c \leq \lambda_i \quad \forall i, \omega \tag{C.14}$$

$$CT_{i\omega} = \sum_c CT_{i\omega}^c \quad \forall i, \omega \tag{C.15}$$

(C.16) describes fixed price contract.

$$CT_{i\omega}^f = \psi_{i\omega}^f B_{i\omega}^f \quad \forall i, \omega \tag{C.16}$$

(C.17)-(C.22) describe discount after a certain amount contract.

$$CT_{i\omega}^d = \psi_{i\omega}^{d1} B_{i\omega}^{d1} + \psi_{i\omega}^{d2} B_{i\omega}^{d2} \quad \forall i, \omega \tag{C.17}$$

$$B^{d}_{i\omega} = B^{d1}_{i\omega} + B^{d2}_{i\omega} \quad \forall i, \omega \tag{C.18}$$

$$B^{d1}_{i\omega} = B^{d11}_{i\omega} + B^{d12}_{i\omega} \quad \forall i, \omega \tag{C.19}$$

$$0 \leq B^{d11}_{i\omega} \leq \sigma^{d}_{i\omega} u^{d1}_{i\omega} \quad \forall i, \omega \tag{C.20}$$

$$B^{d12}_{i\omega} = \sigma^{d}_{i\omega} u^{d2}_{i\omega} \quad \forall i, \omega \tag{C.21}$$

$$0 \leq B^{d2}_{i\omega} \leq A^{U}_{i} u^{d2}_{i\omega} \quad \forall i, \omega \tag{C.22}$$

(C.23)-(C.28) describe bulk discount contract.

$$CT^{b}_{i\omega} = \psi^{b1}_{i\omega} B^{b1}_{i\omega} + \psi^{b2}_{i\omega} B^{b2}_{i\omega} \quad \forall i, \omega \tag{C.23}$$

$$B^{b}_{i\omega} = B^{b1}_{i\omega} + B^{b2}_{i\omega} \quad \forall i, \omega \tag{C.24}$$

$$0 \leq B^{b1}_{i\omega} \leq \sigma^{b}_{i\omega} u^{b1}_{i\omega} \quad \forall i, \omega \tag{C.25}$$

$$\sigma^{b}_{i\omega} u^{b2}_{i\omega} \leq B^{b2}_{i\omega} \leq A^{U}_{i} u^{b2}_{i\omega} \quad \forall i, \omega \tag{C.26}$$

$$u^{b1}_{i\omega} + u^{b2}_{i\omega} = u^{b}_{i\omega} \quad \forall i, \omega \tag{C.27}$$

Because of the decisions on purchase contracts, there are binary variables in the second stage decisions.

$$u^{c}_{i\omega}, u^{d1}_{i\omega}, u^{d2}_{i\omega}, u^{b1}_{i\omega}, u^{b2}_{i\omega} \in \{0, 1\} \tag{C.28}$$

The objective includes the fixed and variable cost of installing the pools and fixed cost

of the feeds, purchase of feeds, and sales of final products.

$$\min \sum_{l} \left(\alpha_l \theta_l + \beta_l S_l\right) + \sum_{i} \left(\delta_i \lambda_i + \xi_i A_i\right) + \sum_{\omega} \tau_\omega \Big( \sum_{i} CT_{i\omega} - \sum_{j} d_j \big( \sum_{l:(l,j)\in T_Y} y_{lj\omega} + \sum_{i:(i,j)\in T_Z} z_{ij\omega} \big) \Big) \tag{C.29}$$

## C.2 Appendix 2: Convex Hull Reformulation of Piecewise McCormick Envelopes in the Crude Selection problem

In the crude selection problem described in subsection 5.6.2, the bilinear terms come from the product of the split fractions and some material flows. The bilinear terms are only in the stage 2 problems. In order to obtain tight convex relaxations in the Benders subproblems, we use the hull relaxation of the piesewise McCormick envelopes.

We represent the split fractions by variable $q_i$, $i \in I$ where $I$ is the set of indices for the $q_i$ variables. The material flows that occur in the bilinear terms are represented by $x_j$, $j \in J$ where $J$ is the set of indices for the $x_j$ variables. Variables $x_j$, $j \in J_i$ are the material flows that occur in the bilinear terms with variable $q_i$. The bilinear terms are represented by $z_{ij}$, $i \in I, j \in J_i$. All the other variables in stage 2 are represented by $y_2$. The stage 1 variables are represented by $y_1$. Without loss of generality, we represent the constraints in a given subproblem by

$$A_1 y_1 + A_2 y_2 + Bz \leq d \tag{C.30a}$$

$$z_{ij} = q_i x_j \quad \forall i \in I, j \in J_i \tag{C.30b}$$

where both the constraints and variable bounds in each subproblem are included. Since the McCormick envelopes of $z_{ij}$ can yield weak relaxations, we uniformly discretize each variable $q_i$ into $m$ intervals where $m$ is an integer that is greater than one. The piecewise McCormick

envelopes can be represented using the following disjunctions.

$$\vee_{n\in\{1,2,\cdots,m\}} \begin{bmatrix} A_1y_1 + A_2y_2 + Bz \leq d \\ z_{ij} \geq q_i \cdot x_j^L + \frac{n-1}{m}(x_j - x_j^L) \quad \forall j \in J_i \\ z_{ij} \geq q_i \cdot x_j^U + \frac{n}{m}(x_j - x_j^U) \quad \forall j \in J_i \\ z_{ij} \leq q_i \cdot x_j^L + \frac{n}{m}(x_j - x_j^L) \quad \forall j \in J_i \\ z_{ij} \leq q_i \cdot x_j^U + \frac{n-1}{m}(x_j - x_j^U) \quad \forall j \in J_i \\ x_j^L \leq x_j \leq x_j^U \quad \forall j \in J_i \\ \frac{n-1}{m} \leq q_i \leq \frac{n}{m} \end{bmatrix} \quad \forall i \in I \tag{C.31}$$

where $x_j^U$ and $x_j^L$ are the upper and lower bounds of variable $x_j$, respectively. In each disjunct $n$, variable $q_i$ is bounded between $\frac{n-1}{m}$ and $\frac{n}{m}$. The McCormick envelopes in disjunct $n$ are changed accordingly with the bounds of $q_i$. The McCormick envelopes of the variables $x_j$ that occur in the same bilinear term of $q_i$ are all included in each disjunct. Note that constraint (C.30a) is included in all the disjuncts. If (C.30a) were not included in the disjuncts, the hull relaxation Grossmann and Trespalacios (2013) of (C.31) would be the same as the normal McCormick envelope relaxation. We use the hull relaxation of (C.31) to have a tight relaxation for the Benders master problem in the crude selection problem. $m = 5$ is used in the computational experiments.

# Appendix D

# Chapter 7 supplementary material

## D.1 Appendix A: Mathematical Formulation for the Deterministic Model

**Nomenclature**

**Sets**

$t \in T$= Time points in the scheduling horizon

$w \in W$ =Wells

$u \in U$ =well operation periods

$o \in O$ =operations $\{TS, HZ, FRAC, TIL\}$

$a \in A$ =well age

**Parameters**

$NRI_w$= $NRI$ of well $w$

$\phi_t$= Discounted rate time period for time period $t$

$ADR$ =Annual discount rate, which is assumed to be 10%

$l_w$=Lateral length of well $w$

$P_{t,w}^{NPV}$ =$NPV$ of a well $w$ where $t$ is the starting week of $TIL$

$\pi_t$= Natural gas price forecast for period t

$\hat{T}$= Total number of time periods used to generate revenue

$\gamma_{a,w}$=Gas production forecast per foot for well $w$ of age $a$

$t^{dev}_{w,o}$ =Development time for operation $o$ at well $w$

$t^{start}_{w,o}$=Earliest start time for operation $o$ at well $w$

$Store^{max}$ = Maximum capacity of storage

$P^{max}_t$ = Maximum capacity of gas that can be produced for time period $t$

$Mob_o$=Mobilization cost for operation $o$

$WOC_{w,o}$=Operation cost for operation $o$ at well $w$

$T_{planning}$=Total number of time periods used for planning horizon

$pDiscProRev^{out}_{w,t}$=Discounted revenue per ft for well outside planning horizon if well is turned in line at time $t$

$P^{max}_t$=Maximum capacity of gas that can be produced for time period $t$

**Binary Variables**

$s^{in}_t$ = If gas is being put into storage at time period $t$

$y^{start}_{t,w,o}$ = If operation $o$ starts in time period $t$ at well $w$

$z^{active}_{t,w,o}$=If operation $o$ is active during time period $t$ at well $w$

**Continuous Variables**

$NPV$=Net present value

$Disc.Rev$=Discounted revenue

$Disc.MC$=Discounted mobilization cost

$Disc.OC$=Discounted operation cost

$Disc.Rev^{in}_t$=Discounted revenue inside the scheduling horizon for time period $t$

$Disc.Rev^{out}_w$=Discounted revenue outside the scheduling horizon for well $w$

$MCO_{t,p,o}$=Mobilization cost for time period $t$ at pad $p$ for operation $o$

$P_{t,w}$=Amount of gas produced at well $w$ during time period $t$

$Store^{out}_t$ = Amount of gas out of storage during time period $t$

$Store^{in}_t$ = Amount of gas put into storage during time period $t$

$StoreLevel_t$ = Storage level during time period $t$

**Parameter Calculations** The parameter $pDiscProdRevOutside_{w,t}$ is calculated prior to optimizing. The parameter represents the revenue per foot outside the planning horizon if a well $w$ is turned in line at time period $t$. To calculate the parameter, we need to sum over the time outside the planning horizon $(a + t > T)$ while the summation is within the time horizon that revenues are considered $(a + t < \hat{T})$. Each term in the summation represents the discount rate $\phi_{a+t}$ times the gas price $\pi_{a+t}$ times the production $\gamma_{a,w}$ when this well is at age $a$.

$$pDiscProdRevOutside_{w,t} = \sum_{a:\left\{T<a+t<\hat{T}\right\}} (\phi_{a+t} \cdot \pi_{a+t} \cdot \gamma_{a,w}) \quad \forall t \in T, w \in W \tag{D.1}$$

The parameter $t^{start}_{w,o}$, the earliest time to start operation $o$ at well $w$, is calculated as,

$$t^{start}_{w,o} = 1 + \sum_{o' \leq o-1} t^{dev}_{w,o'} \quad \forall o \in O, w \in W \tag{D.2}$$

Operation $o$ cannot start on well $w$ until all its previous operations have been completed on well $w$. We assume that the time index starts from 1. Therefore, we need to plus one on the right hand side of (D.2).

### D.1.1 Constraints

We outline the constraints used in the model. We show different types of reformations for some of the constraints.

**General Constraints**

**Single Operation Per Well**: At most one operation can start at each time period $t$ for each well $w$,

$$\sum_{o \in O} y^{start}_{t,w,o} \leq 1 \quad \forall w \in W, t \in T \tag{D.3}$$

**Operation Done at Most Once**: Each operation can only be performed at most once for

each well throughout the planning horizon.

$$\sum_{t \in T} y^{start}_{t,w,o} \leq 1 \quad \forall w \in W, o \in O \tag{D.4}$$

**Earliest Start Time**: Operation $o$ cannot start before its earliest start time defined in (D.2).

$$y^{start}_{t,w,o} = 0 \quad \forall t \in T, w \in W, o \in O, t < t^{start}_{w,o} \tag{D.5}$$

**Sequencing Operations (ordering)**: Operation $o$ cannot start until operation *o-1* is complete. Since each operation can be performed at most once in the entire planning horizon, the time that operation $o$ starts, represented by the left hand side of (D.6), must be greater than or equal to the time that operation $o-1$ starts plus the development time of operation $o-1$, which is represented by the right hand side of (D.6).

$$\sum_{t \in T} \left(t \cdot y^{start}_{t,w,o}\right) \geq \sum_{t \in T} \left(\left[t + t^{dev}_{w,o-1}\right] \cdot y^{start}_{t,w,o-1}\right) \quad \forall w \in W, o \in O, o > 1 \tag{D.6}$$

**Sequencing Operations (ordering) r1**:An alternative way to model this constraint could be using equation (D.7). Here variable $y^{start}_{t,w,o}$ is forced to zero if operation $o-1$ has not been completed at time $t$.

$$y^{start}_{t,w,o} \leq \sum_{t' \leq t - t^{dev}_{w,o-1}} y^{start}_{t',w,o-1} \quad \forall t \in T, w \in W, o \in O, o > 1, t \geq t^{start}_{w,o} \tag{D.7}$$

However, this reformulation performs worse than equation (D.6). Therefore, equation (D.6) is used in the case study.

**Sequencing Operations (tightening constraint):** Operation $o$ cannot happen before operation $o-1$, nor can operation $o-1$ happen after operation $o$,

$$\sum_{t' \leq t} y^{start}_{t',w,o} + \sum_{t' \geq t} y^{start}_{t',w,o-1} \leq 1 \quad \forall w \in W, t \in T, o \in O, o > 1 \tag{D.8}$$

**Operations Completion**: Equation (D.9) forces that the operations on well $w$ must be completed in the planning horizon if the first operation starts in the planning horizon, i.e,

no well can be left unfinished within the planning horizon.

$$\sum_{t \in T} y^{start}_{t,w,o} = \sum_{t \in T} y^{start}_{t,w,o-1} \quad \forall w \in W, o \in O, o > 1, t \geq t^{start}_{w,o} \tag{D.9}$$

**Single Operation Per Time Period Per Pad**: At most one operation can be active at each time period $t$ in the wellpad,

$$\sum_{w \in W} \sum_{o \in O} z^{active}_{t,w,o} \leq 1 \quad \forall t \in T \tag{D.10}$$

where $z^{active}_{t,w,o}$ is a binary variable that decides whether operation $o$ is performed on well $w$ at time $t$,

$$z^{active}_{t,w,o} = \sum_{t' : \{t' \leq t,\ t - t^{dev}_{w,o} < t'\}} y^{start}_{t',w,o} \quad \forall t \in T, w \in W, o \in O \tag{D.11}$$

<u>**Base Production**</u>

The following equation gives production for well $w$ at time $t$. If well $w$ is turned in line at time $t - a$, the age of the well at time $t$ is $a$. The production can be calculated based on the production curve.

$$P_{t,w} = \sum_{a : \{a < t\}} y^{start}_{t-a,w,TIL} \cdot \gamma_{a,w} \cdot l_w \quad \forall t \in T, w \in W \tag{D.12}$$

<u>**Mobilization Constraints**</u>

**Cost of Mobilization:** If operation $o$ is not active at time $t - 1$ but is active at time $o$, we need to move the corresponding equipment and crew for operation $o$ to the wellpad at time $t$ and a mobilization cost is incurred.

$$MCO_{t,p,o} \geq Mob_o \cdot \left( \sum_{w \in W} \left( z^{active}_{t,w,o} \right) - \sum_{w \in W} \left( z^{active}_{t-1,w,o} \right) \right) \quad \forall t \in T, p \in P, o \in O \tag{D.13}$$

Since the mobilization constraints make the problem hard to solve, we propose several reformulations to account for the mobilization costs.

**Cost of Mobilization r1:** We define new binary variables $z_{t,o}^{change}$ which decide whether the wellpad change to perform operation $o$ from a different operation $o' \neq o$ at time $t$ or not. Similar to (D.13), variable $z_{t,o}^{change}$ can be calculated by,

$$z_{t,o}^{change} \geq \sum_{w \in W} \left(z_{t,w,o}^{active}\right) - \sum_{w \in W} \left(z_{t-1,w,o}^{active}\right) \tag{D.14}$$

The mobilization cost at time $t$ can be calculated by summing over the potential operations change at time $t$.

$$MCO_{t,p,o} = \sum_{o \in O} Mob_o z_{t,o}^{change} \tag{D.15}$$

Since the optimization model always try to minimize the costs, variables $z_{t,o}^{change}$ always equal to zero if the corresponding operation is not changed at time $t$.

**Cost of Mobilization r2:** Mobilization cost occurs if we start operation $o$ at time $t$ at any well $w$ and there is no operation $o$ ends at time $t$.

$$MCO_{t,p,o} \geq Mob_o \cdot \left( \sum_{w \in W} \left(y_{t,w,o}^{start}\right) - \sum_{w \in W} \left(y_{t-t_{w,o}^{dev},w,o}^{start}\right) \right) \quad \forall t \in T, p \in P, o \in O \tag{D.16}$$

We did computational experiments with CPLEX on all the three formulations for mobilization. Formulation **cost of mobilization r1** performs best when we give high branching priority to $z_{t,o}^{change}$.

**Capacity Constraint**: For every time period $t$, the production plus the gas released out of storage minus the gas curtailed in the storage must be less than or equal to the maximum gas that can be produced at time $t$. Note that the virtual storage is considered for the whole wellpad. We do not distinguish the storage for each well.

$$\left( \sum_{w \in W} P_{t,w} \right) + Store_t^{out} - Store_t^{in} \leq P_t^{max} \quad \forall t \in T \tag{D.17}$$

**No Storage In/Out**: If storage is being loaded, no gas can be released from storage. If the storage is not being loaded, we constrain the amount of gas released to be less than the

maximum (based on number of wells $W$).

$$\left(1 - s_t^{in}\right) \cdot W \cdot Store^{max} \geq Store_t^{out} \quad \forall t \in T \tag{D.18}$$

**Storage In Used**: Maximum amount of gas put into storage, based on number of wells and if storage is allowed

$$s_t^{in} \cdot W \cdot Store^{max} \geq Store_t^{in} \quad \forall t \in T \tag{D.19}$$

**Storage Level**: For all periods greater than 1, the storage level at time $t$ is equal to the storage level at time *t-1* plus any gas added to storage at time *t,* minus any gas removed from storage at time $t$

$$StoreLevel_t = StoreLevel_{t-1} + Store_t^{in} - Store_t^{out} \quad \forall t \in T, t > 1 \tag{D.20}$$

**Initial Storage Level**: At initial time period ($t$ = *1*), the storage level is equal to any gas added to storage minus any gas removed from storage

$$StoreLevel_1 = Store_1^{in} - Store_1^{out} \tag{D.21}$$

<u>**Revenues**</u>

The **discounted revenue inside the scheduling horizon at time** $t$ **(**$Disc.Rev_t^{in}$ **)** with capacity constraints is,

$$Disc.Rev_t^{in} = \phi_t \cdot \pi_t \cdot \left( \sum_w \left(P_{t,w} \cdot NRI_w\right) + NRI\left(Store_t^{out} - Store_t^{in}\right) \right) \quad \forall t \in T \tag{D.22}$$

where we sum over the sales of gas production at each well $w$ plus the revenue from the gas storage.

The **discounted revenue outside the scheduling horizon (** $Disc.Rev_w^{out}$ **)** is:

$$Disc.Rev_w^{out} = NRI_w \cdot l_w \sum_t pDiscProdRevOutside_{w,t} y_{t,w,TIL}^{start} \quad \forall w \in W \tag{D.23}$$

where parameter $pDiscProdRevOutside_{w,t}$ has been precalculated in equation (D.1). It calculates the revenue well $w$ can generate outside the planning horizon if it is turned in line at time $t$.

The **discounted revenue for gas left in fictional storage (*Disc. Rev*$^{left}$)** is calculated with equation (D.24). There can be some gas left in the storage at the end of the planning horizon. We assume that the sales of the left gas is evenly distributed outside the planning horizon, i.e., from time $T$ to $\hat{T}$.

$$Disc.Rev^{left} = \sum_{u:\{T<u<\hat{T}\}} \left( \phi_u \cdot \pi_u \cdot \bar{NRI} \cdot \frac{StoreLevel_T}{\left(\hat{T} - T\right)} \right) \tag{D.24}$$

The **discounted revenue (*Disc. Rev*)** is equal to the summation of the three types of revenues described above,

$$Disc.Rev = \sum_t Disc.Rev_t^{in} + \sum_w Disc.Rev_w^{out} + Disc.Rev^{left} \tag{D.25}$$

<u>**Costs**</u>

The **discounted operating cost (*Disc. OC*)** is calculated by the following:

$$Disc.OC = \sum_t \phi_t \cdot \sum_o \sum_w \left( y_{t,w,o}^{start} \cdot WOC_{w,o} \right) \tag{D.26}$$

The **discounted mobilization cost (*Disc. MC*)**, calculated based on the **mobilization costs ($\boldsymbol{MCO_{t,p,o}}$)**

$$Disc.MC = \sum_{t} \phi_t \cdot \sum_{p} \sum_{o} MCO_{t,p,o} \tag{D.27}$$

**Objective**

Maximizing NPV

$$NPV = Disc.Rev - Disc.OC - Disc.MC \tag{D.28}$$

# Appendix E

# Chapter 8 supplementary material

## E.1 MILP formulation

**Nomenclature**

**Indices and Sets**

| | |
|---|---|
| $r \in \mathcal{R}$ | set of regions within the area considered |
| $i \in \mathcal{I}$ | set of generator clusters |
| $i \in \mathcal{I}_r$ | set of generator clusters in region $r$ |
| $i \in \mathcal{I}_r^{\text{old}}$ | set of existing generator clusters in region $r$ at the beginning of the time horizon, $\mathcal{I}_r^{\text{old}} \subseteq \mathcal{I}_r$ |
| $i \in \mathcal{I}_r^{\text{new}}$ | set of potential generator clusters in region $r$, $\mathcal{I}_r^{\text{new}} \subseteq \mathcal{I}_r$ |
| $i \in \mathcal{I}_r^{\text{TH}}$ | set of thermal generator clusters in region $r$, $\mathcal{I}_r^{\text{TH}} \subseteq \mathcal{I}_r$ |
| $i \in \mathcal{I}_r^{\text{RN}}$ | set of renewable generator clusters in region $r$, $\mathcal{I}_r^{\text{RN}} \subseteq \mathcal{I}_r$ |
| $i \in \mathcal{I}_r^{\text{Told}}$ | set of existing thermal generator clusters in region $r$, $\mathcal{I}_r^{\text{Told}} \subseteq \mathcal{I}_r^{\text{TH}}$ |
| $i \in \mathcal{I}_r^{\text{Tnew}}$ | set of potential thermal generator clusters in region $r$, $\mathcal{I}_r^{\text{Tnew}} \subseteq \mathcal{I}_r^{\text{TH}}$ |
| $i \in \mathcal{I}_r^{\text{Rold}}$ | set of existing renewable generator clusters in region $r$, $\mathcal{I}_r^{\text{Rold}} \subseteq \mathcal{I}_r^{\text{RN}}$ |
| $i \in \mathcal{I}_r^{\text{Rnew}}$ | set of potential renewable generator clusters in region $r$, $\mathcal{I}_r^{\text{Rnew}} \subseteq \mathcal{I}_r^{\text{RN}}$ |
| $l \in \mathcal{L}^{\text{old}}$ | set of existing transmission lines |
| $l \in \mathcal{L}^{\text{new}}$ | set of prospective transmission lines |
| $j \in \mathcal{J}$ | set of storage unit clusters |
| $t \in \mathcal{T}$ | set of time periods (years) within the planning horizon |
| $d \in \mathcal{D}$ | set of representative days in each year $t$ |
| $s \in \mathcal{S}$ | set of sub-periods of time per representative day $d$ in year $t$ |

| | |
|---|---|
| $k \in \mathcal{K}$ | set of iterations in the Nested Decomposition algorithm |

## Deterministic Parameters

| | |
|---|---|
| $L_{r,t,d,s}$ | load demand in region $r$ in sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $L_t^{\max}$ | peak load in year $t$ (MW) |
| $W_d$ | weight of the representative day $d$ |
| $Hs$ | duration of sub-period $s$ (hours) |
| $Qg_{i,r}^{\text{np}}$ | nameplate (nominal) capacity of a generator in cluster $i$ in region $r$ $(MW)$ |
| $Ng_{i,r}^{\text{old}}$ | number of existing generators in each cluster, $i \in \mathcal{I}_r^{\text{old}}$, per region $r$ at the beginning of the time horizon |
| $Nt_l^{\text{old}}$ | number of units of existing transmission lines for line $l$ |
| $Ng_i^{\max}$ | maximum number of generators in the potential clusters $i \in \mathcal{I}_r^{\text{new}}$ |
| $Q_{i,t}^{\text{inst,UB}}$ | upper bound on yearly capacity installations based on generation technology ($^{\text{MW}}/_{\text{year}}$) |
| $R_t^{\min}$ | system's minimum reserve margin for year $t$ (fraction of the peak load) |
| $ED_t$ | energy demand during year $t$ (MWh) |
| $LT_i$ | expected lifetime of generation cluster $i$ (years) |
| $T_t^{\text{remain}}$ | remaining time until the end of the time horizon at year $t$ (years) |
| $Ng_{i,r,t}^{\text{r}}$ | number of generators in cluster $i$ of region $r$ that achieved their expected lifetime |
| $Q_i^{\text{v}}$ | capacity value of generation cluster $i$ (fraction of the nameplate capacity) |
| $Cf_{i,r,t,d,s}$ | capacity factor of generation cluster $i \in \mathcal{I}_r^{\text{RN}}$ in region $r$ at sub-period $s$, of representative day $d$ of year $t$ (fraction of the nameplate capacity) |
| $Pg_i^{\min}$ | minimum operating output of a generator in cluster $i \in \mathcal{I}_r^{\text{TH}}$ (fraction of the nameplate capacity) |
| $Ru_i^{\max}$ | maximum ramp-up rate for cluster $i \in \mathcal{I}_r^{\text{TH}}$ (fraction of nameplate capacity) |
| $Rd_i^{\max}$ | maximum ramp-down rate for cluster $i \in \mathcal{I}_r^{\text{TH}}$ (fraction of nameplate capacity) |
| $F_i^{\text{start}}$ | fuel usage at startup (MMbtu/MW) |
| $Frac_i^{\text{spin}}$ | maximum fraction of nameplate capacity of each generator that can contribute to spinning reserves (fraction of nameplate capacity) |
| $Frac_i^{\text{Qstart}}$ | maximum fraction of nameplate capacity of each generator that can contribute to quick-start reserves (fraction of nameplate capacity) |
| $Op^{\min}$ | minimum total operating reserve (fraction of the load demand) |
| $Spin^{\min}$ | minimum spinning operating reserve (fraction of the load demand) |
| $Qstart^{\min}$ | minimum quick-start operating reserve (fraction of the load demand) |
| $\alpha^{\text{RN}}$ | fraction of the renewable generation output covered by quick-start reserve (fraction of total renewable power output) |
| $B_l$ | susceptance of transmission line $l$ |

| | |
|---|---|
| $F_l^{\max}$ | capacity of transmission line $l$ (MW) |
| $TIC_{l,t}$ | discounted investment cost of transmission line $l$ in year $t$ |
| $Ns_{j,r}$ | number of existing storage units in each cluster $j$ per region $r$ at the beginning of the time horizon |
| $Charge_j^{\min}$ | minimum operating charge for storage unit in cluster $j$ (MW) |
| $Charge_j^{\max}$ | maximum operating charge for storage unit in cluster $j$ (MW) |
| $Discharge_j^{\min}$ | minimum operating discharge for storage unit in cluster $j$ (MW) |
| $Discharge_j^{\max}$ | maximum operating discharge for storage unit in cluster $j$ (MW) |
| $Storage_j^{\min}$ | minimum storage capacity for storage unit in cluster $j$ (MWh) |
| $Storage_j^{\max}$ | maximum storage capacity (i.e. nameplate capacity) for storage unit in cluster $j$ (MWh) |
| $\eta_j^{\text{charge}}$ | charging efficiency of storage unit in cluster $j$ (fraction) |
| $\eta_j^{\text{discharge}}$ | discharging efficiency of storage unit in cluster $j$ (fraction) |
| $LT_j^{\text{s}}$ | lifetime of storage unit in cluster $j$ (years) |
| $Ir$ | nominal interest rate |
| $If_t$ | discount factor for year $t$ |
| $OCC_{i,t}$ | overnight capital cost of generator cluster $i$ in year $t$ (\$/MW) |
| $ACC_{i,t}$ | annualized capital cost of generator cluster $i$ in year $t$ (\$/MW) |
| $DIC_{i,t}$ | discounted investment cost of generator cluster $i$ in year $t$ (\$/MW) * |
| $SIC_{j,t}$ | investment cost of storage cluster $j$ in year $t$ (\$/MW) |
| $CC_i^{\text{m}}$ | capital cost multiplier of generator cluster $i$ (unitless) |
| $LE_i$ | life extension cost for generator cluster $i$ (fraction of the investment cost of corresponding new generator) |
| $FOC_{i,t}$ | fixed operating cost of generator cluster $i$ (\$/MW) |
| $P_{i,t}^{\text{fuel}}$ | price of fuel for generator cluster $i$ in year $t$ (\$/MMBtu) |
| $HR_i$ | heat rate of generator cluster $i$ (MMBtu/MWh) |
| $Tx_t^{CO_2}$ | carbon tax in year $t$ (\$/kg $CO_2$) |
| $EF_i^{CO_2}$ | full lifecycle $CO_2$ emission factor for generator cluster $i$ ($kgCO_2$/MMBtu) |
| $VOC_{i,t}$ | variable O&M cost of generator cluster $i$ (\$/MWh) |
| $RN_t^{\min}$ | minimum renewable energy production requirement during year $t$ (fraction of annual energy demand) |
| $PEN_t^{\text{rn}}$ | penalty for not meeting renewable energy quota target during year $t$ (\$/MWh) |
| $PEN_t^c$ | penalty for curtailment during year $t$ (\$/MWh) |
| $C_i^{\text{start}}$ | fixed startup cost for generator cluster $i$ (\$/MW) |
| $M_l$ | big-M parameter for line $l$ in transmission expansion equations |
| $s(l)$ | sending end of transmission line $l$ |
| $r(l)$ | receiving end of transmission line $l$ |

*$DIC_{i,t}$ is used in the calculation for the life extension investment cost, which is in terms of a fraction $LE_i$ of the capital cost. Therefore the investment cost for the existing cluster is approximated as being the same as for the potential clusters that have the same or similar generation technology.

## Continuous variables

| | |
|---|---|
| $\Phi$ | net present cost throughout the time horizon, including amortized investment cost, operational and environmental cost (\$) |
| $\Phi_t^{\text{opex}}$ | amortized operating costs in year $t$ (\$) |
| $\Phi_t^{\text{capex}}$ | amortized investment costs in year $t$ (\$) |
| $\Phi_t^{\text{PEN}}$ | amortized penalty costs in year $t$ (\$) |
| $p_{i,r,t,d,s}$ | power output of generation cluster $i$ in region $r$ during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $def_t^{\text{rn}}$ | deficit from renewable energy quota target during year $t$ (MWh) |
| $cu_{r,t,ss,s}$ | curtailment slack generation in region $r$ during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $p_{l,t,d,s}^{\text{flow}}$ | power transfer through transmission line $l$ during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $p_{l,t,d,s}^{\text{flow+}}$ | nonnegative variable, power flow from the sending end of transmission line $l$, $s(l)$, to the receiving end of line $l$, $r(l)$ during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $p_{l,t,d,s}^{\text{flow-}}$ | nonnegative variable, power flow from the receiving end of transmission line $l$, $r(l)$, to the sending end of line $l$, $s(l)$ during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $\theta_{r,t,d,s}$ | voltage angle at region $r$ during sub-period $s$ of representative day $d$ of year $t$ |
| $\theta_{s(l),t,d,s}$ | voltage angle at sending end of transmission line $l$ during sub-period $s$ of representative day $d$ of year $t$ |
| $\theta_{r(l),t,d,s}$ | voltage angle at receiving end of transmission line $l$ during sub-period $s$ of representative day $d$ of year $t$ |
| $\Delta\theta_{l,t,d,s}^1$ | disaggregated variable in the hull formulation, angle difference between the angles at sending end and receiving end of transmission line $l$ during sub-period $s$ of representative day $d$ of year $t$ (MW) if the transmission line $l$ exists in year $t$ |
| $\Delta\theta_{l,t,d,s}^2$ | disaggregated variable in the hull formulation, angle difference between the angles at sending end and receiving end of transmission line $l$ during sub-period $s$ of representative day $d$ of year $t$ (MW) if the transmission line $l$ does not exist in year $t$ |
| $\Delta\theta_{l,t,d,s}^+$ | nonnegative variable, angle difference between the angles at sending end and receiving end of transmission line $l$ during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $\Delta\theta_{l,t,d,s}^-$ | nonnegative variable, angle difference between the angles at receiving end and sending end of transmission line $l$ during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $q_{i,r,t,d,s}^{\text{spin}}$ | spinning reserve capacity of generation cluster $i$ in region $r$ during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $q_{i,r,t,d,s}^{\text{Qstart}}$ | quick-start capacity reserve of generation cluster $i$ in region $r$ during sub-period $s$ of representative day $d$ of year $t$ (MW) |

| | |
|---|---|
| $ngo^{\mathrm{rn}}_{i,r,t}$ | number of generators that are operational in cluster $i \in \mathcal{I}^{\mathrm{RN}}_r$ of region $r$ in year $t$ (continuous relaxation) |
| $ngb^{\mathrm{rn}}_{i,r,t}$ | number of generators that are built in cluster $i \in \mathcal{I}^{\mathrm{RN}}_r$ of region $r$ in year $t$ (continuous relaxation) |
| $ngr^{\mathrm{rn}}_{i,r,t}$ | number of generators that retire in cluster $i \in \mathcal{I}^{\mathrm{RN}}_r$ of region $r$ in year $t$ (continuous relaxation) |
| $nge^{\mathrm{rn}}_{i,r,t}$ | number of generators that had their life extended in cluster $i \in \mathcal{I}^{\mathrm{RN}}_r$ of region $r$ in year $t$ (continuous relaxation) |
| $p^{\mathrm{charge}}_{j,r,t,d,s}$ | power being charged to storage cluster $j$ is region $r$, during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $p^{\mathrm{discharge}}_{j,r,t,d,s}$ | power being discharged to storage cluster $j$ is region $r$, during sub-period $s$ of representative day $d$ of year $t$ (MW) |
| $p^{\mathrm{level}}_{j,r,t,d,s}$ | state of charge of storage cluster $j$ is region $r$, during sub-period $s$ of representative day $d$ of year $t$ (MWh) |
| $p^{\mathrm{level},0}_{j,r,t,d}$ | state of charge of storage cluster $j$ is region $r$ at hour zero of representative day $d$ of year $t$ (MWh) |
| $nso_{j,r,t}$ | number of storage units that are operational in cluster $j$ of region $r$ in year $t$ (continuous relaxation) |
| $nsb_{j,r,t}$ | number of storage units that are built in cluster $j$ of region $r$ in year $t$ (continuous relaxation) |
| $nsr_{j,r,t}$ | number of storage units that retire in cluster $j$ of region $r$ in year $t$ (continuous relaxation) |
| $\Phi_t$ | objective function value for subproblem $t$ assuming exact representation of the cost-to-go function ($) |
| $\Phi_{t,k}$ | objective function value for subproblem $t$ in iteration $k$ ($) |
| $\phi_{t,k}$ | cost-to-go function ($) |
| $\alpha_t$ | expected future year cost, when calculating the cost for year $t$ ($) |
| $\Phi^{\mathrm{LP}}_{t,k}$ | net present cost of the linear relaxation of the subproblem for year $t$ in iteration $k$ ($) |
| $\Phi^{\mathrm{LR}}_{t,k}$ | net present cost of the Lagrangean relaxation of the subproblem for year $t$ in iteration $k$ ($) |
| $\Phi^{\mathrm{LD}}_{t,k}$ | net present cost of the Lagrangean dual of the subproblem for year $t$ in iteration $k$ ($) |
| $\Phi^{\mathrm{OP}}_{t,k}$ | net present cost of the original MILP subproblem for year $t$ in iteration $k$ ($) |
| $ngo^{\mathrm{rn,prev}}_{i,r,t}$ | number of generators that are operational in cluster $i \in \mathcal{I}^{\mathrm{RN}}_r$ of region $r$ in year $t-1$ (continuous relaxation) |
| $ngb^{\mathrm{rn,LT}}_{i,r,t}$ | number of generators that are built in cluster $i \in \mathcal{I}^{\mathrm{RN}}_r$ of region $r$ in year $t-LT_i$ (continuous relaxation) |
| $ngo^{\mathrm{th,prev}}_{i,r,t}$ | number of generators that are operational in cluster $i \in \mathcal{I}^{\mathrm{TH}}_r$ of region $r$ in year $t-1$ (continuous relaxation) |
| $ngb^{\mathrm{th,prev}}_{i,r,t}$ | number of generators that are built in cluster $i \in \mathcal{I}^{\mathrm{TH}}_r$ of region $r$ in year $t-LT_i$ (continuous relaxation) |

| | |
|---|---|
| $x_t$ | investment variables in year $t$ in the concise notation |
| $y_t$ | operating variables in year $t$ in the concise notation |

## Discrete variables

| | |
|---|---|
| $ngo^{\text{th}}_{i,r,t}$ | number of generators that are operational in cluster $i \in \mathcal{I}^{\text{TH}}_r$ of region $r$ in year $t$ (integer variable) |
| $ngb^{\text{th}}_{i,r,t}$ | number of generators that are built in cluster $i \in \mathcal{I}^{\text{TH}}_r$ of region $r$ in year $t$ (integer variable) |
| $ngr^{\text{th}}_{i,r,t}$ | number of generators that retire in cluster $i \in \mathcal{I}^{\text{TH}}_r$ of region $r$ in year $t$ (integer variable) |
| $nge^{\text{th}}_{i,r,t}$ | number of generators that had their life extended in cluster $i \in \mathcal{I}^{\text{TH}}_r$ of region $r$ in year $t$ (integer variable) |
| $ntb_{l,t}$ | binary variable, to denote whether transmission line $l$ is built in year $t$ or not |
| $nte_{l,t}$ | binary variable, represents whether transmission line $l$ has been installed in year $t$ |
| $u_{i,r,t,d,s}$ | number of thermal generators ON in cluster $i \in \mathcal{I}_r$ of region $r$ during sub-period $s$ of representative day $d$ of year $t$ (integer variable) |
| $su_{i,r,t,d,s}$ | number of generators starting up in cluster $i$ during sub-period $s$ of representative day $d$ in year $t$ (integer variable) |
| $sd_{i,r,t,d,s}$ | number of generators shutting down in cluster $i$ during sub-period $s$ of representative day $d$ in year $t$ (integer variable) |

### E.1.1 Operational constraints

The energy balance (E.1) ensures that, in each sub-period $s$ of representative day $d$ in year $t$, the sum of instantaneous power $p_{i,r,t,d,s}$ generated by generator clusters $i$ in region $r$, plus the power flows through the transmission lines $l$ whose receiving end is region $r$, $\sum_{l|r(l)=r} p^{\text{flow}}_{l,t,d,s}$, minus the power flows through the transmission lines $l$ whose sending end is region $r$, $\sum_{l|s(l)=r} p^{\text{flow}}_{l,t,d,s}$, plus the power discharged from all the storage clusters $j$ in region $r$, $p^{\text{discharge}}_{j,r,t,d,s}$, equals the load demand $L_{r,t,d,s}$ at that region $r$, plus the power being charged to the storage clusters $j$ in region $r$, $p^{\text{charge}}_{j,r,t,d,s}$, plus a slack for curtailment of renewable generation $cu_{r,t,d,s}$. Note that we have specified the directions of the transmission lines. Therefore, the

signs of the power flows are unrestricted.

$$\sum_{i}(p_{i,r,t,d,s}) + \sum_{l|r(l)=r} p^{\text{flow}}_{l,t,d,s} - \sum_{l|s(l)=r} p^{\text{flow}}_{l,t,d,s} + \sum_{j} p^{\text{discharge}}_{j,r,t,d,s} = L_{r,t,d,s} + \sum_{j} p^{\text{charge}}_{j,r,t,d,s} + cu_{r,t,d,s} \quad \forall\ r,t,d,s \tag{E.1}$$

The capacity factor constraint (E.2) limits the power outlet $p_{i,r,t,d,s}$ of renewable generators to be equal to a fraction $Cf_{i,r,t,d,s}$ of the nameplate capacity $Qg^{\text{np}}_{i,r}$ in each sub-period $s$ of representative day $d$ in year $t$, where $ngo^{\text{rn}}_{i,r,t}$ represents the number of renewable generators that are operational in year $t$. Due to the flexibility in sizes for renewable generators, $ngo^{\text{rn}}_{i,r,t}$ is relaxed to be continuous.

$$p_{i,r,t,d,s} = Qg^{\text{np}}_{i,r} \cdot Cf_{i,r,t,d,s} \cdot ngo^{\text{rn}}_{i,r,t} \qquad \forall\ i \in \mathcal{I}^{\text{RN}}_{r}, r,t,d,s \tag{E.2}$$

The unit commitment constraint (E.3) computes the number of generators that are ON, $u_{i,r,t,d,s}$, or in startup, $su_{i,r,t,d,s}$, and shutdown, $sd_{i,r,t,d,s}$, modes in cluster $i$ in sub-period $s$ of representative day $d$ of year $t$, and treated as integer variables.

$$u_{i,r,t,d,s} = u_{i,r,t,d,s-1} + su_{i,r,t,d,s} - sd_{i,r,t,d,s} \qquad \forall\ i \in \mathcal{I}^{\text{TH}}_{r}, r,t,d,s \tag{E.3}$$

The ramping limit constraints (E.4)-(E.5) capture the limitation on how fast thermal units can adjust their output power, $p_{i,r,t,d,s}$, where $Ru^{\max}_{i}$ is the maximum ramp-up rate, $Rd^{\max}_{i}$ is the maximum ramp-down rate, and $Pg^{\min}_{i}$ is the minimum operating limit.

$$p_{i,r,t,d,s} - p_{i,r,t,d,s-1} \leq Ru^{\max}_{i} \cdot Hs \cdot Qg^{\text{np}}_{i,r} \cdot (u_{i,r,t,d,s} - su_{i,r,t,d,s}) + \max\left(Pg^{\min}_{i}, Ru^{\max}_{i} \cdot Hs\right) \cdot Qg^{\text{np}}_{i,r} \cdot su_{i,r,t,d,s} \quad \forall\ i \in \mathcal{I}^{\text{TH}}_{r}, r,t,d,s \tag{E.4}$$

$$p_{i,r,t,d,s-1} - p_{i,r,t,d,s} \leq Rd^{\max}_{i} \cdot Hs \cdot Qg^{\text{np}}_{i,r} \cdot (u_{i,r,t,d,s} - su_{i,r,t,d,s}) + \max\left(Pg^{\min}_{i}, Rd^{\max}_{i} \cdot Hs\right) \cdot Qg^{\text{np}}_{i,r} \cdot sd_{i,r,t,d,s} \quad \forall\ i \in \mathcal{I}^{\text{TH}}_{r}, r,t,d,s \tag{E.5}$$

Note that the first terms on the right hand side of (E.4) and (E.5) apply only for normal operating mode (i.e., generator is ON), while the second terms apply for the startup and shutdown modes. This means that generators in normal operating mode have their ramp rates limited by $Ru^{\max}_{i}$ and $Rd^{\max}_{i}$, while generators in startup and shutdown modes have

their ramp rates limited by the least restrictive between $Pg_i^{\min}$ and $Ru_i^{\max}$, $Rd_i^{\max}$ such that their operating limits (Equations E.6 and E.7) are still satisfied.

The operating limits constraints (E.6)-(E.7) specify that each thermal generator is either OFF and outputting zero power, or ON and running within the operating limits $Pg_i^{\min} \cdot Qg_{i,r}^{\text{np}}$ and $Qg_{i,r}^{\text{np}}$. The variable $u_{i,r,t,d,s}$ (integer variable) represents the number of generators that are ON in cluster $i \in \mathcal{I}_r^{\text{TH}}$ at the time period $t$, representative day $d$, and sub-period $s$.

$$u_{i,r,t,d,s} \cdot Pg_i^{\min} \cdot Qg_{i,r}^{\text{np}} \leq p_{i,r,t,d,s} \qquad \forall\ i \in \mathcal{I}_r^{\text{TH}}, r, t, d, s \tag{E.6}$$

$$p_{i,r,t,d,s} + q_{i,r,t,d,s}^{\text{spin}} \leq u_{i,r,t,d,s} \cdot Qg_{i,r}^{\text{np}} \qquad \forall\ i \in \mathcal{I}_r^{\text{TH}}, r, t, d, s \tag{E.7}$$

The upper limit constraint is modified in order to capture the need for generators to run below the maximum considering operating reserves, where $q_{i,r,t,d,s}^{\text{spin}}$ is a variable representing the spinning reserve capacity.

The total operating reserve constraint (E.8) dictates that the total spinning reserve, $q_{i,r,t,d,s}^{\text{spin}}$, plus quick-start reserve, $q_{i,r,t,d,s}^{\text{Qstart}}$, must exceed the minimum operating reserve, $Op^{\min}$, which is a percentage of the load $L_{r,t,d,s}$ in a reserve sharing region $r$ at each sub-period $s$.

$$\sum_{i \in \mathcal{I}_r^{\text{TH}}} \left( q_{i,r,t,d,s}^{\text{spin}} + q_{i,r,t,d,s}^{\text{Qstart}} \right) \geq Op^{\min} \cdot L_{r,t,d,s} \qquad \forall\ r, t, d, s \tag{E.8}$$

Spinning Reserve is the on-line reserve capacity that is synchronized to the grid system and ready to meet electric demand within 10 minutes of a dispatch instruction by the independent system operator (ISO). Quick-start (or non-spinning) reserve is the extra generation capacity that is not currently connected to the system but can be brought on-line after a short delay.

The total spinning reserve constraint (E.9) specifies that the total spinning reserve $q_{i,r,t,d,s}^{\text{spin}}$ must exceed the minimum spinning reserve, $Spin^{\min}$, which is a percentage of the load $L_{r,t,d,s}$ in a reserve sharing region $r$ at each sub-period $s$.

$$\sum_{i \in \mathcal{I}_r^{\text{TH}}} q_{i,r,t,d,s}^{\text{spin}} \geq Spin^{\min} \cdot L_{r,t,d,s} \qquad \forall\ r, t, d, s \tag{E.9}$$

Our model does not currently impose a minimum requirement for total quick-start reserve, as presented in Flores-Quiroz et al. (2016). However, this constraint could be easily incorporated in the formulation to address the extra secondary (quick-start) reserve requirements needed to account for the increasing short term uncertainty due to more renewable generators contributing to the grid.

The maximum spinning reserve constraint (E.10) states that the maximum fraction of capacity of each generator cluster that can contribute to spinning reserves is given by $Frac_i^{\text{spin}}$, which is a fraction of the nameplate capacity $Qg_{i,r}^{\text{np}}$.

$$q_{i,r,t,d,s}^{\text{spin}} \leq u_{i,r,t,d,s} \cdot Qg_{i,r}^{\text{np}} \cdot Frac_i^{\text{spin}} \qquad \forall\ i \in \mathcal{I}_r^{\text{TH}}, r, t, d, s \tag{E.10}$$

The maximum quick-start reserve constraint dictates that the maximum fraction of the capacity of each generator cluster that can contribute to quick-start reserves is given by $Frac_i^{\text{Qstart}}$ (fraction of the nameplate capacity $Qg_{i,r}^{\text{np}}$), and that quick-start reserves can only be provided by the generators that are OFF, i.e., not active.

$$q_{i,r,t,d,s}^{\text{Qstart}} \leq (ngo_{i,r,t}^{\text{th}} - u_{i,r,t,d,s}) \cdot Qg_{i,r}^{\text{np}} \cdot Frac_i^{\text{Qstart}} \qquad \forall\ i \in \mathcal{I}_r^{\text{TH}}, r, t, d, s \tag{E.11}$$

Here the integer variable $ngo_{i,r,t}^{\text{th}}$ represents the number of thermal generators that are operational (i.e., installed and ready to operate) at year $t$.

### E.1.2 Investment-related constraints

The planning reserve requirement (E.12) ensures that the operating capacity is greater than or equal to the annual peak load $L_t^{\max}$, plus a predefined fraction of reserve margin $R_t^{\min}$ of the annual peak load $L_t^{\max}$.

$$\sum_{i \in \mathcal{I}_r^{\text{RN}}} \sum_r \left(Qg_{i,r}^{\text{np}} \cdot Q_i^{\text{v}} \cdot ngo_{i,r,t}^{\text{rn}}\right) + \sum_{i \in \mathcal{I}_r^{\text{TH}}} \sum_r \left(Qg_{i,r}^{\text{np}} \cdot ngo_{i,r,t}^{\text{th}}\right) \geq (1 + R_t^{\min}) \cdot L_t^{\max} \quad \forall\ t \tag{E.12}$$

For all thermal generators, their full nameplate capacity $Qg_{i,r}^{\text{np}}$ counts towards the planning reserve requirement. However, for the renewable technologies (wind, PV and CSP), their contribution is less than the nameplate due to the inability to control dispatch and

the uncertainty of the output (Short et al., 2011). Therefore, the fraction of the capacity that can be reliably counted towards the planning reserve requirement is referred to as the capacity value $Q_i^{\mathrm{v}}$.

The minimum annual renewable generation requirement (E.13) ensures that, in case of policy mandates, the renewable generation quota target, $RN_t^{\min}$, which is a fraction of the energy demand $ED_t$, is satisfied. If not, i.e, if there is a deficit $def_t^{\mathrm{rn}}$ from the quota, this is subjected to a penalty that is included later in the objective function.

$$\sum_d \sum_s \left[ W_d \cdot Hs \cdot \left( \sum_r \left( \sum_{i \in \mathcal{I}_r^{\mathrm{RN}}} p_{i,r,t,d,s} - cu_{r,t,d,s} \right) \right) \right] + def_t^{\mathrm{rn}} \geq RN_t^{\min} \cdot ED_t \quad \forall\, t \tag{E.13}$$

Here $W_d$ represents the weight of the representative day $d$, $Hs$ is the length of the sub-period, $cu_{r,t,d,s}$ is the curtailment of renewable generation, and $ED_t$ represent the energy demand in year $t$:

$$ED_t = \sum_r \sum_d \sum_s (W_d \cdot Hs \cdot L_{r,t,d,s})$$

The maximum yearly installation constraints (E.14)-(E.15) limit the yearly installation per generation type in each region $r$ to an upper bound $Q_{i,t}^{\mathrm{inst,UB}}$ in MW/year. Here $ngb_{i,r,t}^{\mathrm{rn}}$ and $ngb_{i,r,t}^{\mathrm{th}}$ represent the number of renewable and thermal generators built in region $r$ in year $t$, respectively. Note that due to the flexibility in sizes for renewable generators, $ngb_{i,r,t}^{\mathrm{rn}}$ is relaxed to be continuous.

$$\sum_r ngb_{i,r,t}^{\mathrm{rn}} \leq Q_{i,t}^{\mathrm{inst,UB}} / Qg_{i,r}^{\mathrm{np}} \qquad \forall\, i \in \mathcal{I}_r^{\mathrm{Rnew}}, t \tag{E.14}$$

$$\sum_r ngb_{i,r,t}^{\mathrm{th}} \leq Q_{i,t}^{\mathrm{inst,UB}} / Qg_{i,r}^{\mathrm{np}} \qquad \forall\, i \in \mathcal{I}_r^{\mathrm{Tnew}}, t \tag{E.15}$$

### E.1.3 Generator balance constraints

Concerning renewable generator clusters, we define a set of constraints (E.16)-(E.17) to compute the number of generators in cluster $i$ that are ready to operate $ngo_{i,r,t}^{\mathrm{rn}}$, taking into account the generators that were already existing at the beginning of the planning

horizon $Ng_{i,r}^{\text{Rold}}$, the generators built $ngb_{i,r,t}^{\text{rn}}$, and the generators retired $ngr_{i,r,t}^{\text{rn}}$ at year $t$. It is important to highlight that we assume *no lead time* between the decision to build/install a generator and the moment it can begin producing electricity.

$$ngo_{i,r,t}^{\text{rn}} = Ng_{i,r}^{\text{Rold}} + ngb_{i,r,t}^{\text{rn}} - ngr_{i,r,t}^{\text{rn}} \qquad \forall\ i \in \mathcal{I}_r^{\text{RN}}, r, t = 1 \tag{E.16}$$

$$ngo_{i,r,t}^{\text{rn}} = ngo_{i,r,t-1}^{\text{rn}} + ngb_{i,r,t}^{\text{rn}} - ngr_{i,r,t}^{\text{rn}} \qquad \forall\ i \in \mathcal{I}_r^{\text{RN}}, r, t > 1 \tag{E.17}$$

As aforementioned, due to the flexibility in sizes for renewable generators, $ngo_{i,r,t}^{\text{rn}}$, $ngb_{i,r,t}^{\text{rn}}$, and $ngr_{i,r,t}^{\text{rn}}$ are relaxed to be continuous. Note that $ngb_{i,r,t}^{\text{rn}}$ for $i \in \mathcal{I}_r^{\text{Rold}}$ is fixed to zero in all time periods, i.e., the clusters of existing renewable generators cannot have any new additions during the time horizon considered.

We also define a set of constraints (E.18) to enforce the renewable generators that reached the end of their lifetime to either retire, $ngr_{i,r,t}^{\text{rn}}$, or have their life extended, $nge_{i,r,t}^{\text{rn}}$. $Ng_{i,r,t}^{\text{r}}$ is a parameter that represents the number of old generators (i.e., $i \in \mathcal{I}_r^{\text{old}}$) that reached the end of their lifetime, $LT_i$, at year $t$. We assume that the new renewable generators will not need to retire within the planning horizon.

$$Ng_{i,r,t}^{\text{r}} = ngr_{i,r,t}^{\text{rn}} + nge_{i,r,t}^{\text{rn}} \qquad \forall\ i \in \mathcal{I}_r^{\text{Rold}}, r, t \tag{E.18}$$

Concerning thermal generator clusters, we define a set of constraints (E.19)-(E.20) to compute the number of generators in cluster $i$ that are ready to operate $ngo_{i,r,t}^{\text{th}}$, taking into account the generators that were already existing at the beginning of the planning horizon $Ng_{i,r}^{\text{Told}}$, the generators built $ngb_{i,r,t}^{\text{th}}$, and the generators retired $ngr_{i,r,t}^{\text{th}}$ at year $t$.

$$ngo_{i,r,t}^{\text{th}} = Ng_{i,r}^{\text{Told}} + ngb_{i,r,t}^{\text{th}} - ngr_{i,r,t}^{\text{th}} \qquad \forall\ i \in \mathcal{I}_r^{\text{TH}}, r, t = 1 \tag{E.19}$$

$$ngo_{i,r,t}^{\text{th}} = ngo_{i,r,t-1}^{\text{th}} + ngb_{i,r,t}^{\text{th}} - ngr_{i,r,t}^{\text{th}} \qquad \forall\ i \in \mathcal{I}_r^{\text{TH}}, r, t > 1 \tag{E.20}$$

Note that $ngb_{i,r,t}^{\text{th}}$ for $i \in \mathcal{I}_r^{\text{Told}}$ is fixed to zero in all time periods, i.e., the clusters of existing thermal generators cannot have any new additions during the time horizon considered.

We also define a set of constraints (E.21) to enforce the thermal generators that reached the end of their lifetime to either retire, $ngr_{i,r,t}^{\text{th}}$, or have their life extended $nge_{i,r,t}^{\text{th}}$. We

assume that the new thermal generators will not need to retire within the planning horizon.

$$Ng^{\mathrm{r}}_{i,r,t} = ngr^{\mathrm{th}}_{i,r,t} + nge^{\mathrm{th}}_{i,r,t} \qquad \forall\ i \in \mathcal{I}^{\mathrm{Told}}_{r}, r, t \tag{E.21}$$

Finally, we have constraint (E.22) that ensures that only installed generators can be in operation:

$$u_{i,r,t,d,s} \leq ngo^{\mathrm{th}}_{i,r,t} \qquad \forall\ i \in \mathcal{I}^{\mathrm{Tnew}}_{r}, r, t, d, s \tag{E.22}$$

### E.1.4 Storage constraints

We also include a set constraints related to the energy storage devices, which are assumed to be ideal and generic (Pozo et al., 2014). Constraints (E.23)-(E.24) compute the number of storage units that are ready to operate $nso_{j,r,t}$, taking into account the storage units already existing at the beginning of the planning horizon $Ns_{j,r}$ and the ones built $nsb_{j,r,t}$ and retired $nsr_{j,r,t}$ at year $t$. Due to the flexibility in sizes for storage units, $nso_{j,r,t}$, $nsb_{j,r,t}$, and $nsr_{j,r,t}$ are relaxed to be continuous.

$$nso_{j,r,t} = Ns_{j,r} + nsb_{j,r,t} - nsr_{s,r,t} \qquad \forall\ j, r, t = 1 \tag{E.23}$$

$$nso_{j,r,t} = nso_{j,r,t-1} + nsb_{j,r,t} - nsr_{j,r,t} \qquad \forall\ j, r, t > 1 \tag{E.24}$$

Constraints (E.25) and (E.26) establish that the power charge, $p^{\mathrm{charge}}_{j,r,t,d,s}$, and discharge, $p^{\mathrm{discharge}}_{j,r,t,d,s}$, of the storage units in cluster $j$, $nso_{j,r,t}$, has to be within the operating limits: $Charge^{\mathrm{min}}_{j}$ and $Charge^{\mathrm{max}}_{j}$, and $Discharge^{\mathrm{min}}_{j}$ and $Discharge^{\mathrm{min}}_{j}$, respectively.

$$Charge^{\mathrm{min}}_{j} \cdot nso_{j,r,t} \leq p^{\mathrm{charge}}_{j,r,t,d,s} \leq Charge^{\mathrm{max}}_{j} \cdot nso_{j,r,t} \qquad \forall\ j, r, t, d, s \tag{E.25}$$

$$Discharge^{\mathrm{min}}_{j} \cdot nso_{j,r,t} \leq p^{\mathrm{discharge}}_{j,r,t,d,s} \leq Discharge^{\mathrm{max}}_{j} \cdot nso_{j,r,t} \qquad \forall\ j, r, t, d, s \tag{E.26}$$

Constraint (E.27) specifies that the energy storage level, $p^{\mathrm{level}}_{j,r,t,d,s}$, for the storage units in cluster $j$, $nso_{j,r,t}$ has to be within the storage capacity limits $Storage^{\mathrm{min}}_{j}$ and $Storage^{\mathrm{max}}_{j}$.

$$Storage^{\mathrm{min}}_{j} \cdot nso_{j,r,t} \leq p^{\mathrm{level}}_{j,r,t,d,s} \leq Storage^{\mathrm{max}}_{j} \cdot nso_{j,r,t} \qquad \forall\ j, r, t, d, s \tag{E.27}$$

Constraints (E.28) and (E.29) show the power balance in the storage units. The state of

charge $p^{\text{level}}_{j,r,t,d,s}$ at the end of sub-period $s$ depends on the previous state of charge $p^{\text{level}}_{j,r,t,d,s-1}$, and the power charged $p^{\text{charge}}_{j,r,t,d,s}$ and discharged $p^{\text{discharge}}_{j,r,t,d,s}$ at sub-period $s$. The symbols $\eta^{\text{charge}}_j$ and $\eta^{\text{discharge}}_j$ represent the charging and discharging efficiencies, respectively. For the first hour of the day $d$ of year $t$, the previous state of charge (i.e., $s = 0$) is the variable $p^{\text{level},0}_{j,r,t,d}$.

$$p^{\text{level}}_{j,r,t,d,s} = p^{\text{level}}_{j,r,t,d,s-1} + \eta^{\text{charge}}_j \cdot p^{\text{charge}}_{j,r,t,d,s} + p^{\text{discharge}}_{j,r,t,d,s}/\eta^{\text{discharge}}_j \qquad \forall\ j,r,t,d,s > 1 \tag{E.28}$$

$$p^{\text{level}}_{j,r,t,d,s} = p^{\text{level},0}_{j,r,t,d} + \eta^{\text{charge}}_j \cdot p^{\text{charge}}_{j,r,t,d,s} + p^{\text{discharge}}_{j,r,t,d,s}/\eta^{\text{discharge}}_j \qquad \forall\ j,r,t,d,s = 1 \tag{E.29}$$

Constraints (E.30) and (E.31) force the storage units to begin $p^{\text{level},0}_{j,r,t,d}$ and end $p^{\text{level}}_{j,r,t,d,s=S}$ each day $d$ of year $t$ with 50% of their maximum storage $Storage^{\max}_j$. This is a heuristic to attach carryover storage level form one representative day to the next.

$$p^{\text{level},0}_{j,r,t,d} = 0.5 \cdot Storage^{\max}_j \cdot nso_{j,r,t} \qquad \forall\ j,r,t,d \tag{E.30}$$

$$p^{\text{level}}_{j,r,t,d,s} = 0.5 \cdot Storage^{\max}_j \cdot nso_{j,r,t} \qquad \forall\ j,r,t,d,s = S \tag{E.31}$$

### E.1.5 Symmetry breaking constraints of transmission lines

If the potential transmission lines that connect the same two buses are of the same property, the following symmetry breaking constraints can be added to break symmetry. Our computational results suggest that adding these constraints have marginal effects on the computational time. They are not required to be included.

$nte_{l,t} \geq nte_{l-1,t} \quad \forall t, l, l-1$ connecting the same two buses

### E.1.6 Objective Function

The objective of this model is to minimize the net present cost, $\Phi$, over the planning horizon, which includes operating costs $\Phi^{\text{opex}}$, investment costs $\Phi^{\text{capex}}$, and potential penalties $\Phi^{\text{PEN}}$ for not meeting the the targets on renewables.

$$\min \quad \Phi = \sum_t \left(\Phi^{\text{opex}}_t + \Phi^{\text{capex}}_t + \Phi^{\text{PEN}}_t\right) \tag{E.32}$$

The operating expenditure, $\Phi_t^{\text{opex}}$, comprises the variable $VOC_{i,t}$ and fixed $FOC_{i,t}$ operating costs, as well as fuel cost $P_i^{\text{fuel}}$ per heat rate $HR_i$, carbon tax $Tx_t^{\text{CO}_2}$ for $\text{CO}_2$ emissions $EF_i^{CO_2}$, and start-up cost (variable cost $P_i^{\text{fuel}}$ that depends on the amount of fuel burned for startup $F_i^{\text{start}}$, and fixed cost $C_i^{\text{start}}$).

$$\begin{aligned}
\Phi_t^{\text{opex}} = \; & If_t \cdot \Bigg[ \sum_d \sum_s W_d \cdot hs \cdot \\
& \left( \sum_i \sum_r (VOC_{i,t} + P_i^{\text{fuel}} \cdot HR_i + Tx_t^{\text{CO}_2} \cdot EF_i^{CO_2} \cdot HR_i) \cdot p_{i,r,t,d,s} \right) \\
& + \left( \sum_{i \in \mathcal{I}_r^{\text{RN}}} \sum_r FOC_{i,t} \cdot Qg_{i,r}^{\text{np}} \cdot ngo_{i,r,t}^{\text{rn}} \right) \\
& + \left( \sum_{i \in \mathcal{I}_r^{\text{TH}}} \sum_r FOC_{i,t} \cdot Qg_{i,r}^{\text{np}} \cdot ngo_{i,r,t}^{\text{th}} \right) \\
& + \sum_{i \in \mathcal{I}_r^{\text{TH}}} \sum_r \sum_d \sum_s W_d \cdot Hs \cdot su_{i,r,t,d,s} \cdot Qg_{i,r}^{\text{np}} \\
& \cdot \left( F_i^{\text{start}} \cdot P_i^{\text{fuel}} + F_i^{\text{start}} \cdot EF^{CO_2} \cdot Tx_t^{\text{CO}_2} + C_i^{\text{start}} \right) \Bigg]
\end{aligned} \tag{E.33}$$

The capital expenditure, $\Phi_t^{\text{capex}}$, includes the amortized cost of acquiring new generators, $DIC_{i,t}$, new storage devices, $SIC_{j,t}$, and the amortized cost of extending the life of generators that reached their expected lifetime, the amortized cost of building new transmission lines $TIC_{l,t}$. The cost of extending the life of a generator is assumed to be a fraction $LE_i$ of the investment cost, $DIC_{i,t}$, in a new generator with the same or equivalent generation technology. In this framework, the investment cost takes into account the remaining value at the end of the time horizon by considering the annualized capital cost and multiplying it by the number of years remaining in the planning horizon at the time of installation to

calculate the $DIC_{i,t}$.

$$\begin{aligned}\Phi_t^{\text{capex}} = \; & If_t \cdot \Bigg[ \sum_{i \in \mathcal{I}_r^{\text{Rnew}}} \sum_r DIC_{i,t} \cdot CC_i^{\text{m}} \cdot Qg_{i,r}^{\text{np}} \cdot ngb_{i,r,t}^{\text{rn}} \\ & + \sum_{i \in \mathcal{I}_r^{\text{Tnew}}} \sum_r DIC_{i,t} \cdot CC_i^{\text{m}} \cdot Qg_{i,r}^{\text{np}} \cdot ngb_{i,r,t}^{\text{th}} \\ & + \sum_j \sum_r SIC_{j,t} \cdot Storage_j^{\max} \cdot nsb_{j,r,t} \\ & + \sum_{i \in \mathcal{I}_r^{\text{RN}}} \sum_r DIC_{i,t} \cdot LE_i \cdot Qg_{i,r}^{\text{np}} \cdot nge_{i,r,t}^{\text{rn}} \\ & + \sum_{i \in \mathcal{I}_r^{\text{TH}}} \sum_r DIC_{i,t} \cdot LE_i \cdot Qg_{i,r}^{\text{np}} \cdot nge_{i,r,t}^{\text{th}} \\ & + \sum_{l \in \mathcal{L}^{\text{new}}} \sum_t TIC_{l,t} \cdot ntb_{l,t} \Bigg]\end{aligned} \tag{E.34}$$

The capital multiplier $CC_i^{\text{m}}$ associated with new generator clusters is meant to account for differences in depreciation schedules applicable to each technology, with higher values being indicative of slower depreciating schedule and vice versa.

Lastly, the penalty cost, $\Phi_t^{\text{PEN}}$, includes the potential fines for not meeting the renewable energy quota, $PEN_t^{\text{rn}}$, and curtailing the renewable generation.

$$\Phi_t^{\text{PEN}} = \; If_t \cdot \left( PEN_t^{\text{rn}} \cdot def_t^{\text{rn}} + PEN^c \cdot \sum_r \sum_d \sum_s cu_{r,t,d,s} \right) \tag{E.35}$$

**Calculated parameters**

The parameters $If_t$, $DIC_{i,t}$, $ACC_{i,t}$, and $T_t^{\text{remain}}$ are defined as follows. Regarding the parameters used in equations (E.33)-(E.35), the discount factor in year $t$, $If_t$, is calculated from the interest rate, $Ir$:

$$If_t = \frac{1}{(1+Ir)^t}$$

and the discounted investment cost $DIC_{i,t}$ is given by:

$$DIC_{i,t} = ACC_{i,t} \cdot \left( \sum_{t' \leq \min(LT_i, T^{\text{remain}})} If_{t'} \right)$$

where the annualized capital cost $ACC_{i,t}$ is given by:

$$ACC_{i,t} = \frac{OCC_{i,t} \cdot Ir}{1 - \frac{1}{(1+Ir)^{LT_i}}}$$

and the remaining time in the horizon $T_t^{\text{remain}}$ is defined by $T_t^{\text{remain}} = T - t + 1$.

The discounted investment cost of the transmission lines $TIC_{l,t}$ can be calculated similarly.

# E.2

**Theorem 10.** The alternative big-M formulation (ABM) has the same feasible region as the big-M (BM) formulation if the feasible region of ABM is projected to the space of $\Big\{ \bigoplus_{l\in\mathcal{L}^{\text{new}},t\in\mathcal{T},d\in\mathcal{D},s\in\mathcal{S}} (p^{\text{flow}}_{l,t,d,s}, \theta_{s(l),t,d,s}, \theta_{r(l),t,d,s}, nte_{l,t}) \Big\}$, where the symbol '$\oplus$' means the concatenation of all the variables $(p^{\text{flow}}_{l,t,d,s}, \theta_{s(l),t,d,s}, \theta_{r(l),t,d,s}, nte_{l,t})$ over the set $\mathcal{L}^{\text{new}}, \mathcal{T}, \mathcal{D}, \mathcal{S}$.

*Proof.* We first denote the feasible region of ABM as,

$$\mathcal{F}_{\text{ABM}} := \Big\{ \bigoplus_{l\in\mathcal{L}^{\text{new}},t\in\mathcal{T},d\in\mathcal{D},s\in\mathcal{S}} \big(p^{\text{flow}}_{l,t,d,s}, p^{\text{flow}+}_{l,t,d,s}, p^{\text{flow}-}_{l,t,d,s}, \theta_{s(l),t,d,s}, \theta_{r(l),t,d,s}, \Delta\theta^{+}_{l,t,d,s}, \Delta\theta^{-}_{l,t,d,s}, nte_{l,t}\big) \Big| (8.11) - (8.18) \Big\}$$

and the feasible region of BM as

$$\mathcal{F}_{\text{BM}} := \Big\{ \bigoplus_{l\in\mathcal{L}^{\text{new}},t\in\mathcal{T},d\in\mathcal{D},s\in\mathcal{S}} \big(p^{\text{flow}}_{l,t,d,s}, \theta_{s(l),t,d,s}, \theta_{r(l),t,d,s}, nte_{l,t}\big) \Big| (8.5), (8.6) \Big\}$$

To simplify notation, we drop the concatenation symbol '$\bigoplus_{l\in\mathcal{L}^{\text{new}},t\in\mathcal{T},d\in\mathcal{D},s\in\mathcal{S}}$' hereafter in the proof. the projection of $\mathcal{F}_{\text{ABM}}$ onto the space of $(p^{\text{flow}}_{l,t,d,s}, \theta_{s(l),t,d,s}, \theta_{r(l),t,d,s}, nte_{l,t})$ is denoted as $\text{Proj}(\mathcal{F}_{\text{ABM}})$. We want to prove that $\text{Proj}(\mathcal{F}_{\text{ABM}}) = \mathcal{F}_{\text{BM}}$.

We first prove that $\mathcal{F}_{\text{BM}} \subseteq \text{Proj}(\mathcal{F}_{\text{ABM}})$. It suffices to prove that for any $\big(p^{\text{flow},0}_{l,t,d,s}, \theta^{0}_{s(l),t,d,s}, \theta^{0}_{r(l),t,d,s}, nte^{0}_{l,t}\big) \in \mathcal{F}_{\text{BM}}$. We can always find $p^{\text{flow}+,0}_{l,t,d,s}$, $p^{\text{flow}-,0}_{l,t,d,s}$, $\Delta\theta^{+,0}_{l,t,d,s}$, $\Delta\theta^{-,0}_{l,t,d,s}$, such that

$$\big(p^{\text{flow},0}_{l,t,d,s}, p^{\text{flow}+,0}_{l,t,d,s}, p^{\text{flow}-,0}_{l,t,d,s}, \theta^{0}_{s(l),t,d,s}, \theta^{0}_{r(l),t,d,s}, \Delta\theta^{+,0}_{l,t,d,s}, \Delta\theta^{-,0}_{l,t,d,s}, nte^{0}_{l,t}\big) \in \mathcal{F}_{\text{ABM}}$$

It is easy to check that by setting,

$p^{\text{flow}+,0}_{l,t,d,s} = \max\left(0, p^{\text{flow},0}_{l,t,d,s}\right)$

$p^{\text{flow}-,0}_{l,t,d,s} = \max\left(0, -p^{\text{flow},0}_{l,t,d,s}\right)$

$\Delta\theta^{+,0}_{l,t,d,s} = \max\left(0, \theta^{0}_{s(l),t,d,s} - \theta^{0}_{r(l),t,d,s}\right)$

$\Delta\theta^{-,0}_{l,t,d,s} = \max\left(0, \theta^{0}_{r(l),t,d,s} - \theta^{0}_{s(l),t,d,s}\right)$

for all $l \in \mathcal{L}^{\text{new}}, t \in \mathcal{T}, d \in \mathcal{D}, s \in \mathcal{S}$, constraints (8.11)-(8.18) can be satisfied.

Next, we prove that $\text{Proj}(\mathcal{F}_{\text{ABM}}) \subseteq \mathcal{F}_{\text{BM}}$. It suffices to show that equations (8.5) and (8.6) are consequences of equations (8.11)-(8.18) by carefully choosing Farkas multipliers. For example, inequality $p^{\text{flow}}_{l,t,d,s} - B_l(\theta_{s(l),t,d,s} - \theta_{r(l),t,d,s}) \leq M_l(1 - nte_{l,t})$ can be obtained by $(8.11) - (8.14) + (8.15) - B_l \times (8.16)$. Inequality $p^{\text{flow}}_{l,t,d,s} \leq F^{\max}_l nte_{l,t}$ can be obtained by (8.15)+(8.17)+$[-p^{\text{flow}-,0}_{l,t,d,s} \leq 0]$. The other two inequalities in equations (8.5) and (8.6) can be obtained similarly. We conclude $\text{Proj}(\mathcal{F}_{\text{ABM}}) \subseteq \mathcal{F}_{\text{BM}}$. □

## E.3 Computational results for the GTEP problem with 4 representative days(continued)

Some additional results for the GTEP problem with 4 representative days are provided in this section.

### E.3.1 Problem sizes after presolve

To show the effects of CPLEX presolve in reducing the size of the fullspace model, the sizes of the three formulations after presolve are shown in Table E.5.

Table E.5: Sizes of the three formulations after CPLEX presolve

| formulation | Integer Var | Binary Var | Continuous Var | Constraints |
|---|---|---|---|---|
| big-M | 277,811 | 19,552 | 409,997 | 1,305,617 |
| alternative big M | 277,811 | 19,552 | 811,236 | 1,708,633 |
| hull | 277,811 | 19,552 | 409,997 | 1,305,709 |

### E.3.2 Computational results for solving the problem with the integer variables in the operating problems relaxed

Since the integer variables in the operating subproblems are relaxed in the tailored Benders decomposition algorithm, it is fair to compare the Benders decomposition results with solving the fullspace model while relaxing the integer variables in the operating problems. The computational results are shown in Table E.6. It is clear that the fullspace problem cannot be solved even with most of the integer variables relaxed.

We also evaluate the LP relaxations of the three different formulations. In this particular instance, the optimal objective value of the three formulations are the same, being 260.848. It should be noted that this is still consistent with our theoretical analysis since problems with different feasible region can yield same optimal objective value. In some smaller instances that we had tested before, we did observe that the objective value of the LP relaxation of the hull reformulation is larger than those of the big-M and the alternative big-M reformulation.

Table E.6: Fullspace GTEP problem with relaxed operating integer variables.

| formulation | Integer Var | Binary Var | Continuous Var | Constraints | UB | LB | Wall time |
|---|---|---|---|---|---|---|---|
| big-M | 2,280 | 2,800 | 836,266 | 1,543,966 | - | 21.13 | 36,000 |
| alternative big M | 2,280 | 2,800 | 1,373,866 | 2,081,566 | - | 21.13 | 36,000 |
| hull | 2,280 | 2,800 | 1,105,066 | 2,081,566 | - | 21.13 | 36,000 |

### E.3.3 The convergence of the nested Benders decomposition algorithm

The upper and lower bound that can be obtained from the nested Benders decomposition algorithm with respect to time (in seconds) are shown in Figures E.1-E.3. The bounds are significantly improved in the first few iterations. However, closing the gap takes the majority of the time, i.e., the bounds are improved slowly after around 15,000 seconds.

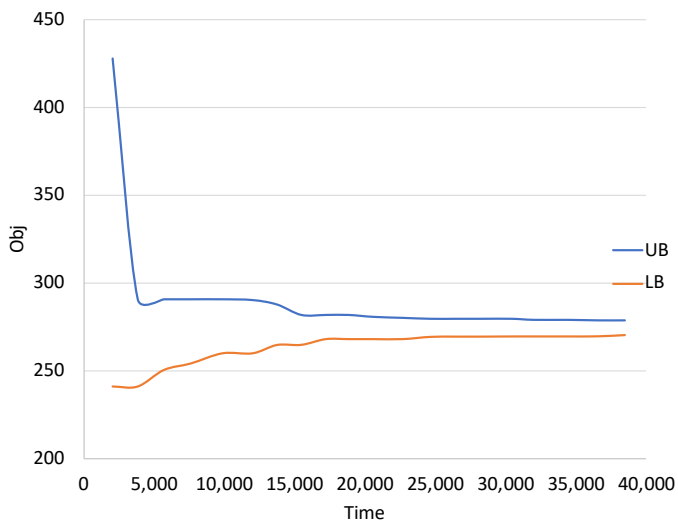


Figure E.1: The upper and the lower bounds of the nested Benders decomposition algorithm with the big-M formulation

### E.3.4 Problem size in the Benders decomposition algorithm

The sizes of the Benders master and the Benders subproblem are show in Tables E.7 and E.8, respectively.

Table E.7: Size of the Benders master problem

| Integer Var | Binary Var | Continuous Var | Constraints |
|---|---|---|---|
| 2,280 | 2,800 | 5,800 | 7,986 |

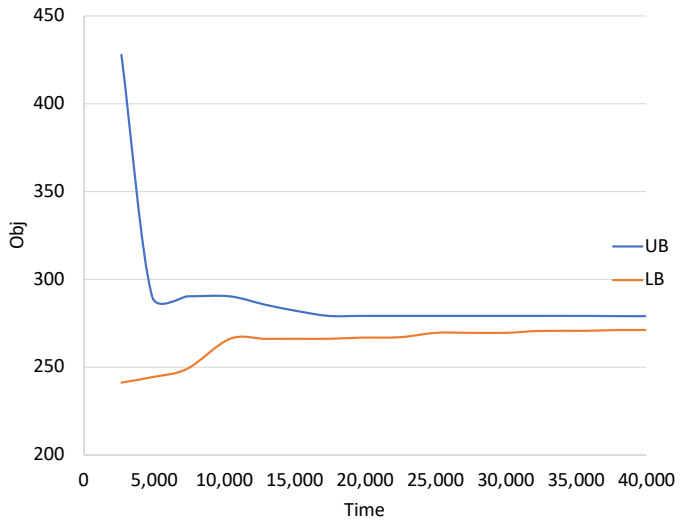


Figure E.2: The upper and the lower bounds of the nested Benders decomposition algorithm with the alternative big-M formulation

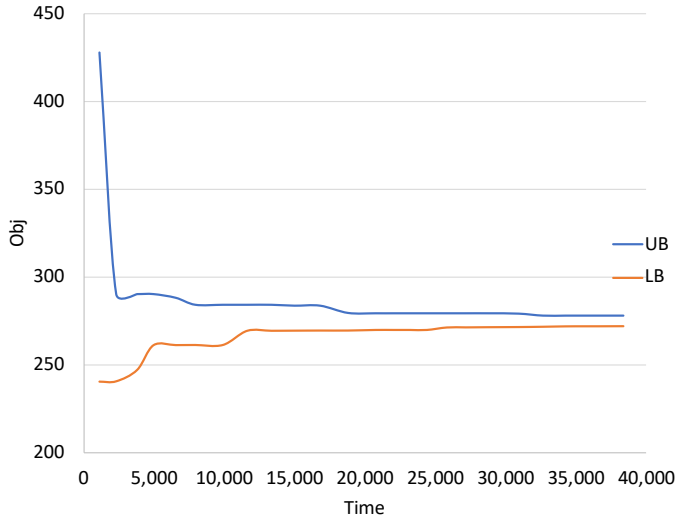


Figure E.3: The upper and the lower bounds of the nested Benders decomposition algorithm with the hull formulation

Table E.8: Size of the Benders subproblem with the different formulations

| formulation | Integer Var | Binary Var | Continuous Var | Constraints |
|---|---|---|---|---|
| big-M | 13,632 | 0 | 27,951 | 76,799 |
| alternative big M | 13,632 | 0 | 54,831 | 103,679 |
| hull | 13,632 | 0 | 41,391 | 103,679 |

## E.4 Sensitivity analysis of the number of representative days

The number of representative days in the operating problem can affect the capacity expansion results Lara et al. (2018); Teichgraeber and Brandt (2019); Scott et al. (2019). There is a computational tractability versus model fidelity trade-off in deciding the number of representative days. As long as we have enough computational resources, it is always worth including as many representative days as possible in the model to have a more accurate representation.

We perform a sensitivity analysis on the capacity expansion decisions and the optimal objective value found by the algorithm varying the number of representative days from 4 to 15. All these GTEP models with the alternative big-M formulation are solved to within 1% optimality gap within 10 hours. The capacities of nuclear and coal do not change over the 20 years in all the cases. Therefore, we only report the capacities of natural gas, solar and wind at the end of the time horizon with different representative days in Figure E.4. The total transmission costs and optimal objective values are also shown in Figure E.4. All the values shown in Figure E.4 are normalized by their corresponding values in the 15-representative-day results.

It can be seen from Figure E.4 that as we increase the number of representative days the optimal value of the objective function increases as well. Recall that we select the centroid of each cluster in the time series clustering algorithm. Therefore, as the number of clusters increases, it is more likely to select days with higher volatility as our representative days, which increase the operating cost.

There are no general trends in the generation and capacity expansion results, except that wind capacity has a decreasing trend with the increase in the number of representative days. If the number of representative days is larger than 10, most of the values shown in Figure E.4 are within 10% of the 15 representative day values.

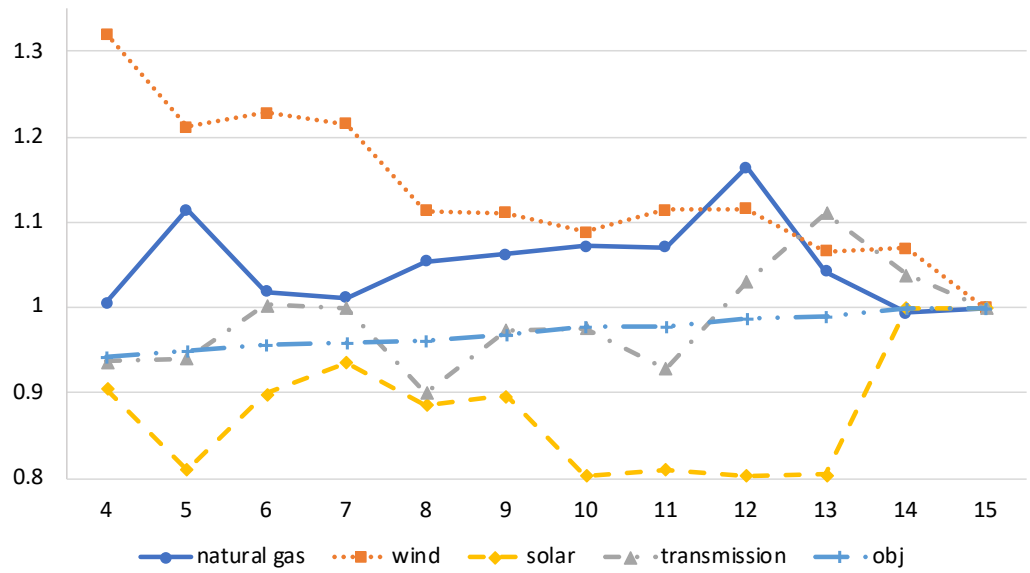


Figure E.4: Natural gas, wind, solar capacity, total transmission cost at the end of the planning horizon and the optimal value of objective function for different representative days. All the values are normalized by the corresponding 15-representative-day values respectively.

## E.5 Comments on Warm-start Both Algorithms

One of the anonymous reviewer of this paper has raised the question of warm-starting both algorithms. The suggestions raised by this reviewer are "For the GTEP, I see multiple ways to generate incumbent investment solutions: (1) Solve the model with fewer representative days, perhaps even one day; (2) solve the model in 2-, 3-, or 5-year increments and interpolate intermediate investment decisions; (3) relax the DC flow requirement; (4) disallow certain technologies from being built (e.g., coal, nuclear, and storage); or some combinations thereof."

We agree with this reviewer that Warm-start could potentially accelerate these two decomposition algorithms. We also agree that points (1)-(4) proposed by the reviewer can reduce the size of the fullspace model but sill generate some "trial" investment decisions. These trial solutions can be used to warm-start both decomposition algorithms by generating a number of Benders cuts before we start the Benders/nested Benders iteration. However,

the Benders implementation is from CPLEX that does not support the callbacks to implement the warm-start techniques yet. On the other hand, our experience suggests that an implementation of Benders by ourselves in Python is mostly likely to fail to perform better than the implementation in CPLEX that is well-engineered codes.

If we take a step back and consider a vanilla implementation of Benders decomposition algorithm, i.e, without using CPLEX Benders implementation or its lazycallback, we would expect warm-start to reduce the number of times that the master problem is solved. In our case, the master problem only involves the investment decisions, which are relatively easy to solve. The benefit of warm-start may not be as significant as in other applications where the master problem is the bottleneck.

As for the nested Benders decomposition algorithm, warm-start should be able to reduce the number of forward passes as suggested by the reviewer. Since the forward pass involves both the investment decisions and the operating decisions, the time saving should be more significant than warmstarting the Benders decomposition. However, the iteration/time limit is not the main reason why nested Benders decomposition does not return a tight optimality gap. The main reason is that all the integer variables after year one have to be relaxed in nested Benders. The tightest lower bound that can be obtained using the nested Benders is weaker than that of the Benders decomposition. The other reason why nested Benders is slow is because of the Pyomo/Python-based implementation is not as fast as using the CPLEX Benders implementation in model generation. Due to these two fundamental limitations, although warm-start should improve the performance of the nested Benders decomposition by decreasing the number of forward passes, we do not expect the improved nested Benders algorithms to perform better than the Benders decomposition by CPLEX.

# Appendix F

# Chapter 9 supplementary material

## F.1 Proofs of the Theorems

### F.1.1 Proof of Theorem 1

*Proof.* It suffices to prove that for any feasible solution to (FD) it is always possible to construct some feasible solution to (RD) that yields the same objective value. Suppose ($x_t$, $y_{t,d}$, $\forall t \in \mathcal{T}, d \in \mathcal{D}$) is a feasible solution of (FD). Suppose that by applying the $k$-means clustering algorithm, set $\mathcal{D}$ is partitioned into $|\mathcal{K}|$ clusters, i.e., $\mathcal{D} = \cup_{k\in\mathcal{K}}\mathcal{D}_k$ where set $\mathcal{D}_k$ represents the days in the $k$th cluster. Constraint (9.4b) can be rewritten as

$$A_{t,d}x_t + B_t y_{t,d} \leq b_{t,d} \quad \forall d \in \mathcal{D}_k \tag{F.1}$$

for all $k \in \mathcal{K}, t \in \mathcal{T}$. We aggregate all the constraints in (F.1) for a given cluster $k$ in year $t$ and obtain

$$(\sum_{d\in\mathcal{D}_k} A_{t,d})x_t + B_t(\sum_{d\in\mathcal{D}_k} y_{t,d}) \leq \sum_{d\in\mathcal{D}_k} b_{t,d} \tag{F.2}$$

Divide (F.3) by $|\mathcal{D}_k|$ on both sides we have

$$A_{t,k}x_t + B_t(\frac{\sum_{d\in\mathcal{D}_k} y_{t,d}}{|\mathcal{D}_k|}) \leq b_{t,k} \tag{F.3}$$

because by definition of the $k$-means clustering algorithm, $A_{t,k}$, $b_{t,k}$ are the mean values of $\{A_{t,d}, \forall d \in \mathcal{D}_k\}$ and $\{b_{t,d}, \forall d \in \mathcal{D}_k\}$, respectively.

It is easy to see that if we set $y_{t,k} = \frac{\sum_{d \in \mathcal{D}_k} y_{t,d}}{|\mathcal{D}_k|}$ for all $k \in \mathcal{K}, t \in \mathcal{T}$. $(x_t, y_{t,k}, k \in \mathcal{K}, t \in \mathcal{T})$ is a feasible solution of (RD) that yields the same objective value as $(x_t,\ y_{t,d},\ \forall t \in \mathcal{T}, d \in \mathcal{D})$ for problem (FD). This completes the proof. □

### F.1.2 Proof of Theorem 2

*Proof.* Since the clustering error is zero, the input parameters in each cluster must be the same. Therefore, in the fullspace problem (RD), the optimal operating decisions for the days in each cluster $k \in \mathcal{K}$ are the same, i.e., $y_{t,d} = y_{t,d'}$ for any $d, d'$ within the same cluster. The decisions within each cluster can be aggregated without any sacrifice of optimality. The aggregated problem is exactly the problem (RD) without relaxing the integrality constraints on $y_{t,k}$. □

### F.1.3 Proof of Theorem 3

*Proof.* Since there is only one cluster and one year in the planning horizon, the clustering error of the cost-based approach being zero implies that the optimal investment decisions of all the days are the same if the CEP problem is solved for each day in $\mathcal{D}$ individually. Denote this investment decision as $\mathbf{x}^{common}$. Clearly, $\mathbf{x}^{common}$ is optimal for (FD). Furthermore, (RD) is solved using the medoid of the single cluster that corresponds to one of the days in $\mathcal{D}$. By definition, $\mathbf{x}^{common}$ is also optimal for (RD). □

### F.1.4 Proof of Theorem 4

*Proof.* Using the same proof reasoning as in Theorem 1, the variables and constraints corresponding to $\mathcal{K}_1$ can be aggregated to produce the variables and constraints corresponding to $\mathcal{K}_2$ and the inequality follows. □