LARGE SCALE MULTISTAGE STOCHASTIC INTEGER PROGRAMMING WITH APPLICATIONS IN ELECTRIC POWER SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Jikai Zou

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Industrial and Systems Engineering

Georgia Institute of Technology

August 2017

LARGE SCALE MULTISTAGE STOCHASTIC INTEGER PROGRAMMING WITH APPLICATIONS IN ELECTRIC POWER SYSTEMS

Approved by:

Dr. Shabbir Ahmed (advisor)
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Andy Sun (co-advisor)
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Alexander Shapiro
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Natashia Boland
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Andy Philpott
Department of Engineering Science
The University of Auckland

Date Approved: May 1st, 2017

ACKNOWLEDGEMENTS

I am indebted to a number of remarkable people who supported my thesis in different forms. First and foremost, I would like to thank my advisors, Dr. Shabbir Ahmed and Dr. Andy Sun, for their guidance and intellectual insights which are indispensable for this work. I also want to thank them for their care and support of my personal growth and career pursuit. I would like to thank my committee members, Dr. Alexander Shapiro, Dr. Natashia Boland, and Dr. Andy Philpott, for the helpful discussion and feedback.

I feel extremely grateful to have the opportunity to study in such a vibrant and exciting academic environment in the past four years. In particular, I would like to thank Dr. Craig Tovey, Dr. Robert Foley, Dr. Arkadi Nemirovski, Dr. Santanu Dey, Dr. David Goldberg, and Dr. Alejandro Toriello, for their excellent courses. Furthermore, I want to thank all my classmates, Rui Gao, Weijun Xie, Yifan Liu, Helin Zhu, Yang Cao, Çağlar Çağlayan, and Fabien Caspani, for the valuable discussions and friendship; I also want to thank Diego Moran, Daniel Silva, Ezgi Karabulut, Luke Marshall, Beste Basciftci, Yufeng Cao, and Asteroide Santana, for organizing DOS seminars together. Finally I would like to thank all the staff of ISyE, especially Amanda Ford and Mark Reese.

I am extraordinarily fortunate to be accompanied by great friends over the past four years. Those good memories will be my lifelong treasure. My warm thanks go to: Daniel Zink, Heike Zink, Kevin Ryan, Lauren Ryan, Álvaro Lorca, Mathias Klapp, James Bailey, Jeffrey Pavelka, Brian Kues, Amelia Musselman, Andres Iroume, Burak Kocuk, Alfredo Torrico Palacios, Tony Yaacoub, Joshua Hale, Can Zhang, Xiaowei Yue, Yilun Chen, Yuan Li, Yuchen Zheng, Junzhuo Chen, Yuanshuo Zhao, Di Wu, and Junxuan Li.

Finally, I would like to thank my parents, Jie Zou and Xiaofeng Zhong. Thank you for all the support and love over the past twenty-nine years of my life. I want to specially dedicate this thesis to my wife, Zhihua Dong. Thank you for being there all the time despite the distance, and thanks for your patience, inspiration, and love, along this incredible journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Multistage stochastic integer programming (MSIP) is a framework for sequential decision making under uncertainty, where the uncertainty is modeled by a general stochastic process, and the decision space involves integer variables and complicated constraints. Many power system applications, such as generation capacity planning and scheduling under uncertainty stemming from renewable generation, demand variability and price volatility, can be naturally formulated as MSIP problems. In this thesis, we develop general purpose solution methods for large-scale MSIP problems and demonstrate their effectiveness on various power systems applications.

In the first part of this thesis, we consider an MSIP approach for electrical power generation capacity expansion problems under demand and fuel price uncertainty. We propose a partially adaptive stochastic mixed integer optimization model in which the capacity expansion plan is fully adaptive to the uncertainty evolution up to a certain period, and is static thereafter. Any solution to the partially adaptive model is feasible to the multistage model and we provide analytical bounds on the quality of such a solution. We propose an algorithm that solves a sequence of partially adaptive models, to recursively construct an approximate solution to the multistage problem. We apply the proposed approach to a realistic generation expansion case study.

In the second part of this thesis, we develop decomposition algorithms for general MSIP problems with binary state variables. By exploiting the binary nature of the state variables, we extend the nested Benders decomposition algorithm to this problem class. Key to our developments are new families of cuts that guarantee finite convergence of the proposed algorithm. We also propose a stochastic variant of the nested Benders decomposition algorithm, called Stochastic Dual Dynamic integer Programming (SDDiP), and give a rigorous proof of its finite convergence with probability one to an optimal policy. We provide extensive computational results using the SDDiP approach for generation capacity

planning, portfolio optimization, and airline revenue management problems.

The final part of this thesis focuses on adapting the SDDiP approach to solve the multistage stochastic unit commitment (MSUC) problem. Unit commitment is a key operational problem in power systems used to determine the optimal generation schedule over the next day or week. Incorporating uncertainty in this already difficult optimization problem imparts severe challenges. We reformulate the MSUC problem such that each stage problem only depends on information from the previous stage and the uncertainty realization. This new formulation is amenable to our SDDiP approach. We propose a variety of computational enhancements to adapt the method to MSUC. Through extensive computational results, we demonstrate the effectiveness of our approach in solving realistic scale MSUC problems.

CHAPTER 1

INTRODUCTION

1.1 Multistage Stochastic Integer Programming

Multistage stochastic integer programming (MSIP) is a framework for sequential decision making under uncertainty where the decision space is typically high dimensional and involves complicated constraints, and the uncertainty is modeled by general stochastic processes. Integer decisions are required by the nature of specific applications, and they usually exist in multiple decision periods. To describe a generic formulation for an MSIP, let us start with a canonical deterministic optimization problem with T stages:

$$\min_{(x_1,y_1),\ldots,(x_T,y_T)} \left\{ \sum_{t=1}^{T} f_t(x_t, y_t) : \ (x_{t-1}, x_t, y_t) \in X_t, \ \forall\, t = 1, \ldots, T \right\}.$$

In the above formulation we explicitly distinguish two sets of decision variables in each stage, namely, the *state* variable, denoted by x_t, which links successive stages, and the *local* or *stage* variable, denoted by y_t, which is only contained in the subproblem at stage t. This form is without loss of generality since any multistage optimization problem can be formulated in this form by introducing additional constraints and variables. Note that, for notational convenience, the above formulation includes variable x_0 which is assumed to be fixed. The function f_t and the set X_t denote the objective and constraints associated with stage t, respectively. We focus on the mixed-integer linear setting where the objective function f_t is linear, and the constraint system X_t is of the form

$$B_t x_{t-1} + A_t x_t + C_t y_t \geq b_t$$

along with integrality restrictions on a subset of the variables. The data required in stage t is $\xi_t := (f_t, X_t)$ where, with some notational abuse, we have used f_t and X_t to denote the data for the objective f_t and constraints in X_t. Let us denote the feasible region of the stage t problem by $F_t(x_{t-1}, \xi_t)$ which depends on the decision in stage $t-1$ and the information ξ_t available in stage t. Suppose now the data $(\boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T)$ 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 ξ_t to denote a specific realization. Similarly, we use $\boldsymbol{\xi}_{[t,t']}$ 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 ξ_t and then take an action (x_t, y_t) depending on the previous stage decision x_{t-1} (also known as *state*) and the observed data ξ_t to optimize the *expected* future cost. A nested formulation for this MSIP problem is:

$$\min_{(x_1,y_1)\in F_1} \Big\{ f_1(x_1,y_1) + \mathbb{E}_{\boldsymbol{\xi}_{[2,T]}|\xi_{[1,1]}} \Big[\min_{(x_2,y_2)\in F_2(x_1,\xi_2)} \Big\{ f_2(x_2,y_2,\xi_2) + \cdots + \mathbb{E}_{\boldsymbol{\xi}_{[T,T]}|\boldsymbol{\xi}_{[1,T-1]}} \Big[\min_{(x_T,y_T)\in F_T(x_{T-1},\xi_T)} \Big\{ f_T(x_T,y_T,\xi_T) \Big\} \Big] \Big\} \Big], \quad (1.1)$$

where $\mathbb{E}_{\boldsymbol{\xi}_{[t,T]}|\boldsymbol{\xi}_{[1,t-1]}}$ denotes the conditional expectation operation in stage t with respect to $\boldsymbol{\xi}_{[t,T]}$ given $\boldsymbol{\xi}_{[1,t-1]}$ in stage $t-1$. In the cases when all decision variables are continuous, such a problem is usually referred to as *multistage stochastic linear programming*, or MSLP.

1.1.1 Scenario Tree Formulation

Computational approaches for MSIP are based on approximating the stochastic process $(\boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_T)$ by a process having finitely many realizations in the form of a scenario tree (see e.g., Ruszczynski and Shapiro 2003). Such an approximation may be constructed by Monte Carlo methods as in the sample average approximation (SAA) approach or various other constructive methods (Kuhn 2006; Shapiro 2003; Pennanen 2009; Høyland and Wallace 2001; Pflug 2001; Heitsch et al. 2006). Under the scenario tree setting, problem (1.1) is

a stochastic optimization problem over a finite number of scenarios, thus can be regarded as a large-scale deterministic mixed integer program. In the following, we introduce some standard notation for the scenario tree model.

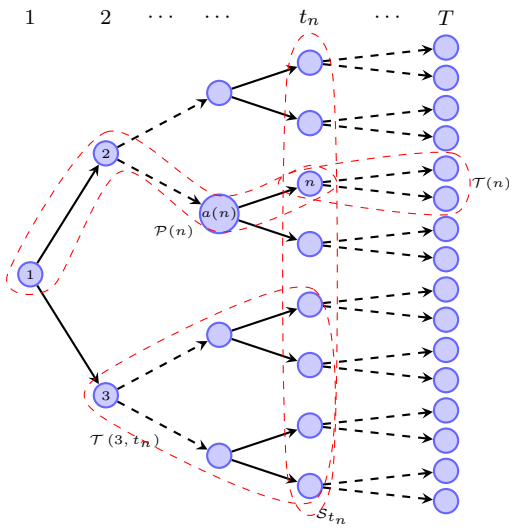


Figure 1.1: An example of a scenario tree

Let $\mathcal{T}$ be the scenario tree associated with the underlying stochastic process. There are T levels corresponding to the T decision-making stages and the set of nodes in stage t is denoted by $\mathcal{S}_t$. The root node in stage 1 is labeled 1, i.e., $\mathcal{S}_1 = \{1\}$. Each node n in stage $t > 1$ has a unique parent node $a(n)$ in stage $t-1$. We denote the stage containing node n by $t(n)$. The subtree with root n and ending at period τ $(t_n \leq \tau \leq T)$ is denoted as $\mathcal{T}(n,\tau)$. Denote $\mathcal{T}(n,T)$ as $\mathcal{T}(n)$ for simplicity. The set of children nodes of a node n is denoted by $\mathcal{C}(n)$. The set of nodes on the unique path from node 1 to node n, including node n, is denoted by $\mathcal{P}(n)$. A truncated path $\mathcal{P}(n,t)$ denotes the path from the t-th ancestor node of n to n. A node $n \in \mathcal{S}_t$ represents a state of the world in stage t and corresponds to the information sequence $\{\xi_m = (f_m, X_m)\}_{m\in\mathcal{P}(n)}$. The total probability associated with node n is denoted as p_n, which is the probability of realization of the $t(n)$-period data sequence $\{\xi_m\}_{m\in\mathcal{P}(n)}$. The sum of probabilities of all nodes in each level is equal to 1, i.e., $\sum_{n\in\mathcal{S}_t} p_n = 1$ for all t. For $m \in \mathcal{T} \setminus \{1\}$ and $n = a(m)$, $q_{nm} := p_m/p_n$ is the conditional

probability of transitioning from node n to node m. Therefore, the sum of probabilities of the nodes with the same parent is equal to the probability of their parent node, i.e., $\sum_{m\in\mathcal{C}(n)} p_m = p_n$ for all $n \in \mathcal{T}$. Each node in the final stage $\mathcal{S}_T$ corresponds to a realization of the data for the full planning horizon, i.e., all T periods, and is called a scenario. We say two scenarios s and s' are *indistinguishable* at period t if the paths corresponding to s and s' pass through the same set of nodes up to period t. Let $\Omega = \{\mathcal{P}(n) : n \in \mathcal{S}_T\}$ be the set of all possible scenarios and $N = |\Omega|$ the total number of scenarios. Figure 1.1 depicts an illustration of a general scenario tree. A *recombining* scenario tree is one in which for any two nodes n and n' in $\mathcal{S}_t$, the set of children nodes $\mathcal{C}(n)$ and $\mathcal{C}(n')$ are defined by identical data and probabilities.

Since the decisions in a stage are taken after observing the data realization we associate the decisions to the nodes of the tree. The resulting formulation, called the *extensive form*, is

$$\min_{x_n,y_n} \left\{ \sum_{n\in\mathcal{T}} p_n f_n(x_n, y_n) : \ (x_{a(n)}, x_n, y_n) \in X_n \ \forall\, n \in \mathcal{T} \right\}. \tag{1.2}$$

Note that, for notational convenience, we include the variable $x_{a(1)}$ which is assumed to be fixed. While (1.2) is a deterministic optimization problem, it has very large scale as the size of the scenario tree grows exponentially with dimension of the uncertain parameters and the number of stages. An alternative to the extensive form (1.2) is to formulate the MSIP problem via the following dynamic programming (DP) equations

$$\min_{x_1,y_1} \left\{ f_1(x_1, y_1) + \sum_{m\in\mathcal{C}(1)} q_{1m} Q_m(x_1) : \ (x_{a(1)}, x_1, y_1) \in X_1 \right\} \tag{1.3}$$

where for each node $n \in \mathcal{T} \setminus \{1\}$

$$Q_n(x_{a(n)}) = \min_{x_n,y_n} \left\{ f_n(x_n, y_n) + \sum_{m\in\mathcal{C}(n)} q_{nm} Q_m(x_n) : \ (x_{a(n)}, x_n, y_n) \in X_n \right\}. \tag{1.4}$$

The optimization problem in (1.4) is referred to as a node problem. We will also denote the

optimal value function (of $x_{a(n)}$) and the *expected cost-to-go function* at node n by $Q_n(\cdot)$ and $\mathcal{Q}_n(\cdot) := \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(\cdot)$, respectively.

1.1.2 Existing Approaches and Challenges

MSIP involves the triple difficulties of uncertainty, dynamics and nonconvexity, and therefore is an extremely challenging class of problems. These problems require carefully embedding a stochastic process within a mixed integer program while preserving the decision dynamics. A common formulation involves approximating the stochastic process by a scenario tree and exploding the underlying optimization problem into a very large scale mixed integer program (1.2). While such an approach results in problems suitable for deterministic optimization methods, it is typically limited to fairly small problems. An alternative approach is a nested value function based formulation akin to the usual dynamic programming approach. The nonconvexity associated with the value functions of integer programs and their recursions result in very complex structured dynamic programs. Details on the related literature can be found in the beginning of each chapter.

1.2 Applications to Energy Sectors

Many power system applications can be naturally formulated as MSIP problems because of their multistage decision dynamics, integer requirement on decisions, as well as the need of incorporating both endogenous and exogenous uncertainty. A success story of using general multistage stochastic programming approach to the energy sector is hydrothermal generation scheduling in Brazil (Pereira and Pinto 1991), involving the month-to-month planning of power generation of a system of hydro and thermal plants in order to meet energy demand in the face of stochastic water inflows into the hydro-reservoirs (see also Cerisola et al. 2012; Philpott and Matos 2012; Shapiro et al. 2013).

Numerous other applications in energy have been proposed since then. A long term capacity planning of generation and transmission systems (see e.g., Akbari et al. 2011;

Baringo and Conejo 2013) seek optimal investment decisions to expand system generation and transmission capacities. Such problems usually involve uncertainties from demand, energy price, technology evolution, and government regulations. Another example is the mid-term generation scheduling, also known as unit commitment (see e.g. Takriti et al. 2000; Sen et al. 2006; Cerisola et al. 2009a). In these problems, system operator must determine which generation units to run in each time step (hourly or shorter) over the next day or week, and at what output level the running units should generate. To ensure the reliability and security requirement of the schedule, certain forms of reserved capacity and contingency are often imposed. Other applications include planning and operation of renewable energy systems (Jacobs et al. 1995; Fleten and Kristoffersen 2008; Bruno et al. 2016), management of electricity storage systems (Meibom et al. 2011; Mokrian and Stephen 2006), etc. A more comprehensive introduction of general stochastic programming applications in energy-related optimization problems can be found in Wallace and Fleten (2003). It is worth mentioning that the power of MSIP is not limited to energy sector, it has also found various applications in finance, manufacturing, as well as natural resource management.

1.3 Summary of Contribution

Motivated by its application potential, there has been a great deal of research on general multistage stochastic programming. Major progress has been made on theoretical issues such as structure, complexity, and approximability, as well as on effective decomposition algorithms. Much of the progress, however, has been restricted to the two-stage setting ($T = 2$ in (1.1)), or the *linear* setting, i.e. MSLP. In this thesis, we study the solution methods for MSIP. The goal is to design efficient and scalable algorithms to solve this type of large-scale optimization problems.

In Chapter 2, we consider an MSIP approach for electrical power generation capacity expansion problems under demand and fuel price uncertainty. We propose a partially

adaptive stochastic mixed integer optimization model in which the capacity expansion plan is fully adaptive to the uncertainty evolution up to a certain period, and is static thereafter. Any solution to the partially adaptive model is feasible to the multistage model and we provide analytical bounds on the quality of such a solution. We propose an algorithm that solves a sequence of partially adaptive models, to recursively construct an approximate solution to the multistage problem. We apply the proposed approach to a realistic generation expansion case study.

In Chapter 3, we develop decomposition algorithms for general MSIP problems with binary state variables. By exploiting the binary nature of the state variables, we extend the nested Benders decomposition algorithm to this problem class. Key to our developments are new families of cuts that guarantee finite convergence of the proposed algorithm. We also propose a stochastic variant of the nested Benders decomposition algorithm, called Stochastic Dual Dynamic integer Programming (SDDiP), and give a rigorous proof of its finite convergence with probability one to an optimal policy. We provide extensive computational results using the SDDiP approach for generation capacity planning, portfolio optimization and, airline revenue management problems.

Chapter 4 focuses on adapting the SDDiP approach to solve the multistage stochastic unit commitment (MSUC) problem. Unit commitment is a key operational problem in power systems used to determine the optimal generation schedule over the next day or week. Incorporating uncertainty in this already difficult optimization problem imparts severe challenges. We reformulate the MSUC problem such that each stage problem only depends on information from the previous stage and the uncertainty realization. This new formulation is amenable to our SDDiP approach. We propose a variety of computational enhancements to adapt the method to MSUC. Through extensive computational results, we demonstrate the effectiveness of our approach in solving realistic scale MSUC problems.

CHAPTER 2
PARTIALLY ADAPTIVE STOCHASTIC OPTIMIZATION FOR CAPACITY EXPANSION PROBLEMS

2.1 Introduction

Generation expansion planning (GEP) is the problem of determining an optimal construction and generation plan over a finite planning horizon of both existing and new generation power plants to meet future electricity demand, while satisfying operational, economic and regulatory constraints. The objective of GEP is to minimize the total investment cost and generation cost. Investment cost depends on the number of newly built generators over the planning horizon, and generation cost reflects the cost incurred at the operation level. GEP is considered a major part of power system planning problems. It is challenging due to its large scale, long-term horizon, and nonlinear and discrete nature. A major difficulty in GEP, as well as in more general capacity expansion problems, is to deal with uncertainty in future demand, and various other uncertainties such as technological breakthroughs, cost structures, etc.

From the optimization modeling perspective, we can address the uncertainty issue using a two-stage or multistage stochastic optimization model. In a two-stage model, the capacity expansion plan for the entire planning horizon is decided prior to the uncertainty realized and hence allows no adaptivity to uncertainty evolution over time. In contrast, a multistage stochastic optimization model allows full adaptivity to the uncertainty evolution, but is extremely difficult to solve.

In this chapter, we develop a unifying framework and an efficient algorithm for solving the stochastic capacity expansion problem. The key contributions are summarized here.

1. **A partial adaptive model for GEP.** We propose a partially adaptive (PA) stochastic

model for the general capacity expansion planning problem and apply it to the generation expansion problem. The partially adaptive stochastic model allows the capacity expansion decision to be fully adaptive to uncertainty up to a certain set of prescribed nodes in the scenario tree, and then restricts the expansion decisions to a two-stage structure thereafter. In this way, the PA model provides a natural generalization for the two-stage and multistage models. A related idea is used to approximate an infinite horizon stochastic program by a finite-horizon model through aggregating all future decisions after a finite time period (Grinold 1986). The two approaches have very different motivations. More importantly, the PA model proposed in this chapter allows more flexibility in constructing the 'compressed' scenario tree. Also, to the best of our knowledge, partially adaptive models have not been considered for the stochastic capacity expansion problem.

2. **Performance analysis on PA model.** We provide theoretical analysis of the performance of the PA model. In particular, we prove upper and lower bounds for the performance difference between a PA model and full multistage model. The bounds are related to the cost parameters as well as the structure of the compressed scenario tree.

3. **An approximation algorithm.** We further propose a new efficient approximation algorithm to solve the PA model. The algorithm recursively solves a sequence of smaller PA models and constructs a feasible solution to the multistage stochastic optimization model. We identify analytical conditions on problem parameters under which the proposed algorithm recovers an optimal multistage solution. We also conduct extensive computational experiments on a large scale 10-year GEP problem with real-world demand and fuel price data. The results demonstrate that the proposed algorithm can solve the GEP problem within a reasonable computation time limit and yields a significantly better solution than directly solving the multistage model.

2.2 Related Work

The first stochastic capacity expansion planning model with demand uncertainty dates back to Manne (1961), and has been followed by extensive work in the area (see e.g., Erlenkotter 1967; Giglio 1970; Freidenfelds 1980; Davis et al. 1987; Bean et al. 1992). These early works assume simplified underlying stochastic processes for the uncertainties to obtain analytical solutions and are typically restricted to capacity expansion with single resource.

Stochastic optimization approaches for capacity expansion problems utilize scenario trees to model uncertainty. Both two-stage and multistage stochastic optimization models have been proposed. In a typical two-stage stochastic GEP model, the first-stage decisions are the capacity expansion decisions over the *entire* planning horizon, which are made before any uncertainty is realized; then the operational level decisions for generation production are made as the second stage decision, which are fully adaptive to uncertainty realizations. For example, Jin et al. (2011) propose such a two-stage stochastic GEP model and further consider both risk-neutral and risk-averse objectives, and apply a random sampling based method to approximately solve the problem. Bloom (1983) and Bloom et al. (1984) investigate in a two-stage GEP model with probabilistic reliability constraints and solve it using generalized Benders' decomposition. The second stage subproblems are nonlinear because of the system reliability constraints and are solved using a procedure called probabilistic simulation. Bienstock and Shapiro (1988) propose another type of two-stage stochastic GEP model which decomposes the entire planning horizon into two stages. The authors consider demand and fuel price uncertainty, and solve the problem using a Benders decomposition type algorithm.

Multistage stochastic optimization models allow the capacity expansion decisions to be fully adaptive to uncertainty realizations. For instance, Berman et al. (1994) consider a scenario-based multistage stochastic optimization model for capacity expansion of a single technology. Chen et al. (2002) extend this model to multiple technologies. In both models,

the capacity expansion decisions are assumed to be continuous variables. Ahmed and Sahinidis (2003) consider a multistage stochastic formulation, where the capacity expansion decisions are binary variables. The authors propose an LP-relaxation-based approximation algorithm for this problem and prove its asymptotic optimality as the planning horizon goes to infinity. Ahmed et al. (2003) further exploit the special structure of the stochastic lot-sizing problem and develop a branch-and-bound algorithm to obtain global optimal solutions. Singh et al. (2009) propose a column-generation approach for solving such problems. Huang and Ahmed (2009) show that multistage capacity expansion models can have significant advantages in terms of total expected costs over two-stage models. Ryan et al. (2011) and Wallace and Fleten (2003) present comprehensive surveys on stochastic modeling in planning and operation of electric power systems. In a broader domain of multistage stochastic optimization, extensive research has been done in term of how to allocate "stages" (Grinold 1986; Nielsen and Zenios 1996; Dempster et al. 2000; Dupačová et al. 2009), but most of them assume continuous variables and focus on financial applications. It is fair to say that, despite the intense research efforts, multistage stochastic expansion models remain extremely challenging to solve for large-scale cases within a reasonable amount of computation time.

The remainder of the chapter is organized as follows. Section 2.3 presents the PA model development. In Section 2.4, we analyze the performance of the PA model. We present the approximation algorithm in Section 2.5 and computational results in Section 2.6. Finally, we provide concluding remarks in Section 2.7.

2.3 Model Development

In this section, we develop a partially adaptive stochastic mixed integer optimization model for the GEP problem. Section 2.3.1 contains a detailed PA model for the GEP problem. In Sections 2.3.2, we present a PA model for the general capacity expansion problem, and discuss its connections to two-stage and multistage models.

2.3.1 PA Model for GEP Problem

We consider two main sources of uncertainty in the GEP model, namely, the uncertainty of future demand and fuel prices, mainly natural gas prices, which are historically very volatile (Jin et al. 2011). A scenario tree approach is adopted to model the evolution of these uncertainties over a multi-year finite planning horizon.

Now we can introduce the PA model for the GEP problem. The basic structure of the model is adopted from the two-stage model of Jin et al. (2011). As discussed in the introduction, the PA-GEP model imposes a multistage structure for the capacity expansion decision from period 1 up to period μ, i.e. the expansion decisions in these periods need to satisfy *non-anticipativity* constraints, and then from period $\mu+1$ to T, the expansion decision follows a two-stage structure, i.e., they are made without the knowledge of uncertainty realization after period μ. Note that the operational level decisions of generation production have a multistage structure throughout the planning horizon. In the following, we present a detailed PA-GEP model with *critical time* μ.

Parameters

T	Number of time periods in planning horizon.
$\mathcal{I}$	Set of available technologies to expand capacity.
$\mathcal{K}_t$	Set of sub-periods in each planning period t.
c_{it}	Unit cost of a type i generator at period t. ($/MW)
m_i^{max}	Maximum capacity of a type i generator. (MW)
n_i^{max}	Maximum rating of output of a type i generator. (MW)
u_i^{max}	Maximum number of type i generators that can be built over the planning horizon.
u_i^0	Number of pre-existing type i generators.
b_{ink}	Unit generation cost for a type i generator in sub-period k at node n. ($/MWh)

d_{nk}	Hourly demand in sub-period k at node n. (MW)
h_{nk}	Number of hours in sub-period k at node n.
q	Unit penalty cost for unmet demand. ($/MW)
r	Annual interest rate.
μ	Critical time of the model.

Variables

x_{in}	Number of type i generators built at node n.
v_{ink}	Hourly generation of a type i generator in sub-period k at node n. (MW)
w_{nk}	Unmet hourly demand in sub-period k at node n. (MW)

Formulation

$$\min \sum_{n\in\mathcal{T}} \left(\sum_{i\in\mathcal{I}} \frac{p_n c_{it_n} m_i^{max}}{(1+r)^{t_n-1}} x_{in} + \sum_{i\in\mathcal{I}} \sum_{k\in\mathcal{K}_{t_n}} \frac{p_n h_{nk} b_{ink}}{(1+r)^{t_n-1}} v_{ink} + \sum_{k\in\mathcal{K}_{t_n}} \frac{p_n h_{nk} q}{(1+r)^{t_n-1}} w_{nk} \right) \tag{2.1a}$$

$$\text{s.t.} \sum_{m\in\mathcal{P}(n)} x_{im} + u_i^0 \geq \frac{1}{n_i^{max}} v_{ink}, \quad \forall\, i\in\mathcal{I}, k\in\mathcal{K}_{t_n}, n\in\mathcal{T} \tag{2.1b}$$

$$\sum_{m\in\mathcal{P}(n)} x_{im} \leq u_i^{max}, \quad \forall\, i\in\mathcal{I}, n\in\mathcal{S}_T \tag{2.1c}$$

$$\sum_{i\in\mathcal{I}} v_{ink} + w_{nk} = d_{nk}, \quad \forall\, k\in\mathcal{K}_{t_n}, n\in\mathcal{T} \tag{2.1d}$$

$$x_{in_1} = x_{in_2}, \quad \forall\, i\in\mathcal{I}, n_1, n_2 \in \mathcal{T}(m)\cap\mathcal{S}_t, m\in\mathcal{S}_\mu, t>\mu \tag{2.1e}$$

$$x_{in}\in\mathbb{Z}_+, v_{ink}, w_{nk}\in\mathbb{R}_+, \quad \forall\, i\in\mathcal{I}, k\in\mathcal{K}_{t_n}, n\in\mathcal{T}. \tag{2.1f}$$

In the PA-GEP model (2.1), the objective function consists of investment, generation, and penalty costs, all of which are discounted to the beginning of the planning horizon. Constraints (2.1b) indicate that the output by each type of generators during any sub-period cannot exceed the aggregate output rating of all available (both pre-existing and newly built) generators of that type. Constraints (2.1c) require the total number of each type of generators built in all possible scenarios to be within the quantity limitation. Furthermore, in

constraints (2.1d), we enforce the equality between the demand and the sum of generation output and unmet demand. Note that renewable generation such as wind and solar power can be easily incorporated in the above expansion model, e.g., by modeling wind and solar availability as random negative demand on the scenario tree.

The key constraint of the PA-GEP model is (2.1e), which imposes a two-stage model after period μ at each possible outcome in that period. As a result, for any possible realization at period μ, there is only one capacity expansion decision in each period subsequently. In other words, future capacity expansion decisions are made at period μ without knowing further uncertainty realizations. Note that non-anticipativity constraints, which impose the requirement that period t decision for two scenarios that are indistinguishable at stage t must be identical, are implicitly embedded in the nodal formulation due to the scenario tree structure.

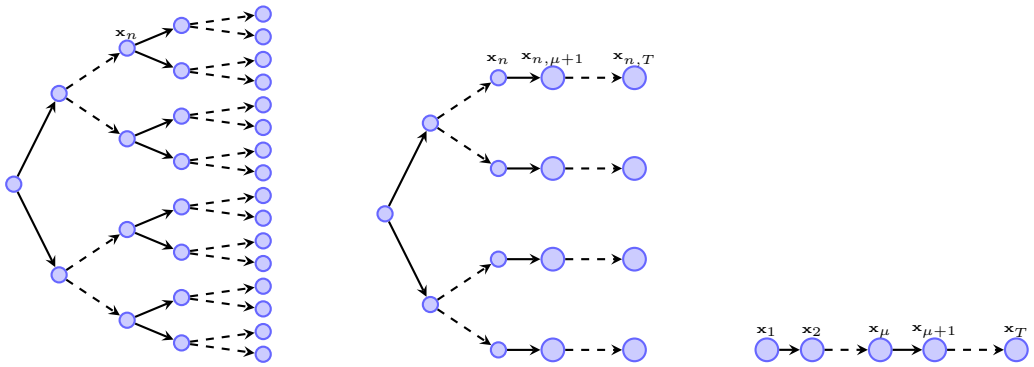
2.3.2 PA Model for General Capacity Expansion Planning

The model (2.1) can be written in a more abstract and general way to obtain a PA model for a general stochastic capacity expansion planning problem. All the analysis of the performance of the PA model in Section 2.4 is done for the following generic model.

$$
\begin{aligned}
[\Pi_{\text{PA}}(\mu)] \quad \min \quad & \sum_{n\in\mathcal{T}} p_n \left(a_n^\top x_n + \sum_{k\in\mathcal{K}_{t_n}} b_{nk}^\top y_{nk} \right) && \text{(2.2a)}\\
\text{s.t.} \quad & \sum_{m\in\mathcal{P}(n)} x_m \geq A_{nk} y_{nk}, \quad \forall\, k \in \mathcal{K}_{t_n}, n \in \mathcal{T} && \text{(2.2b)}\\
& \sum_{m\in\mathcal{P}(n)} x_m \leq u, \quad \forall\, n \in \mathcal{S}_T && \text{(2.2c)}\\
& B_{nk} y_{nk} = d_{nk}, \quad \forall\, k \in \mathcal{K}_{t_n}, n \in \mathcal{T} && \text{(2.2d)}\\
& x_{n_1} = x_{n_2}, \quad \forall\, n_1, n_2 \in \mathcal{T}(m) \cap \mathcal{S}_t, m \in \mathcal{S}_\mu, t > \mu && \text{(2.2e)}\\
& x_n \in \mathbb{Z}_+^I,\ y_{nk} \in \mathbb{R}_+^J, \quad \forall\, k \in \mathcal{K}_{t_n}, n \in \mathcal{T}. && \text{(2.2f)}
\end{aligned}
$$

Here, the vector x_n corresponds to the investment decisions (x_{in}) in all types of generators at node n, thus $I = |\mathcal{I}|$; y_{nk} corresponds to the operation level decisions in sub-period k at node n, i.e., the generation output (v_{ink}) and the amount of unmet demand (w_{nk}), thus $J = I + 1$. The parameters a_n and b_{nk} correspond to the objective coefficients of x_n and y_{nk}, respectively. The matrices A_{nk} and B_{nk} correspond to the coefficient matrix on the right-hand-side in constraints (2.1b) and coefficient matrix on the left-hand-side in constraints (2.1d), respectively. Constraints (2.2e) correspond to (2.1e). Finally, the parameter vectors u and d_{nk} correspond to the right-hand-side vectors in constraints (2.1c) and (2.1d), respectively. Note that d_{nk} has dimension of 1 in model (2.1), since electricity is the only type of output.

As a unifying framework, the proposed PA model $\Pi_{\text{PA}}(\mu)$ generalizes the existing two-stage and multistage models in the capacity planning literature (see e.g., Ahmed and Sahinidis 2003; Singh et al. 2009; Jin et al. 2011). Figure 2.1 illustrates the decision structures of the PA, multistage, and two-stage models. In particular, the left network in



(a) $\Pi_{\text{MS}} : \mu = T$ (b) $\Pi_{\text{PA}}(\mu) : \mu = t_n$ (c) $\Pi_{\text{TS}} : \mu = 1$

Figure 2.1: Capacity expansion decisions in Π_{MS}, $\Pi_{\text{PA}}(\mu)$ and Π_{TS}

Figure 2.1 represents the decision structure of the multistage model, where the decision maker has a specific expansion plan for each node in the scenario tree. The middle network corresponds to a PA model, where each node n has an expansion plan x_n up to time period μ. After that, the subtree $\mathcal{T}(n)$ is "compressed" into a chain, where the decision maker only has one expansion plan for all the nodes in each following time period. The two-stage model

Π_{TS} on the right has the most simple expansion plan, where each planning period has only one decision throughout the planning horizon.

For later use, we also provide explicit formulations for the two-stage and multistage capacity expansion models below.

$$
\begin{aligned}
[\Pi_{\text{TS}}] \quad \min \quad & \sum_{t=1}^{T} \bar{a}_t x_t + \sum_{n\in\mathcal{T}} p_n \Bigg(\sum_{k\in\mathcal{K}_{t_n}} b_{nk}^\top y_{nk} \Bigg) && \text{(2.3a)}\\
\text{s.t.} \quad & \sum_{s=1}^{t_n} x_s \geq A_{nk} y_{nk}, \quad \forall\, k \in \mathcal{K}_{t_n}, n \in \mathcal{T} && \text{(2.3b)}\\
& \sum_{s=1}^{T} x_s \leq u && \text{(2.3c)}\\
& B_{nk} y_{nk} = d_{nk}, \quad \forall\, k \in \mathcal{K}_{t_n}, n \in \mathcal{T} && \text{(2.3d)}\\
& x_t \in \mathbb{Z}_+^I,\ y_{nk} \in \mathbb{R}_+^J, \quad \forall\, t = 1, \ldots, T, k \in \mathcal{K}_{t_n}, n \in \mathcal{T}. && \text{(2.3e)}
\end{aligned}
$$

$$
[\Pi_{\text{MS}}] \quad \min \quad \left\{ \sum_{n\in\mathcal{T}} p_n \Bigg(a_n^\top x_n + \sum_{k\in\mathcal{K}_{t_n}} b_{nk}^\top y_{nk} \Bigg) \ :\ \text{(2.2b), (2.2c), (2.2d), (2.2f).} \right\} \quad \text{(2.4)}
$$

In Π_{TS}, $\bar{a}_t$ is the average cost across the period t, i.e., $\bar{a}_t = \sum_{n\in\mathcal{S}_t} p_n a_n$. Note that in a two-stage formulation, other choices of investment cost, such as the highest cost in each stage may also be used.

2.4 Performance of $\Pi_{\text{PA}}(\mu)$

In this section, we investigate the performance of the PA model $\Pi_{\text{PA}}(\mu)$, namely, the difference between optimal values of the PA model and the multistage model Π_{MS}. Let $v^{\text{PA}}(\mu)$ and v^{MS} be the optimal value of $\Pi_{\text{PA}}(\mu)$ and Π_{MS}, respectively. Note that an optimal solution to $\Pi_{\text{PA}}(\mu)$ is a feasible solution to the multistage model Π_{MS} with value $v^{\text{PA}}(\mu)$. Thus, we define $\text{Gap}(\mu) := v^{\text{PA}}(\mu) - v^{\text{MS}}$ as the performance gap between the two models. It is clear that $\text{Gap}(\mu) \geq 0$, for all $1 \leq \mu \leq T$, and $\text{Gap}(T) = 0$, since the optimal solution

to the PA model is a feasible solution to the multistage model. Also, Gap(1) is the gap between two-stage and multistage models, which is studied in Huang and Ahmed (2009). Therefore, the following analysis generalizes the bounds therein.

Notice that both $\Pi_{\text{PA}}(\mu)$ and Π_{MS} are integer programs, thus they both become difficult to solve when the planning horizon is long. Moreover, as long as $\mu < T$, Π_{MS} will have significantly more integer variables than $\Pi_{\text{PA}}(\mu)$ since the number of nodes in a scenario tree grows exponentially in the number of planning periods. Thus the multistage model becomes much more difficult to solve. If Gap(μ) is small for modest μ, then $\Pi_{\text{PA}}(\mu)$ can provide a good, easier-to-compute approximation to Π_{MS} with guaranteed performance. For this reason, we provide analytical lower and upper bounds for Gap(μ), using instance data and the optimal LP-relaxation solutions of these two models.

A brief outline of our approach is summarized as follows. Motivated by an important substructure of these models, we decompose the original problem into subproblems, each of which corresponds to a single type of expansion technology. We solve the LP relaxations of the original PA and multistage models with multiple types of technologies. Then we use their optimal solutions and input data to bound the gap for single-technology subproblems. Finally we aggregate these bounds for subproblems to obtain both upper and lower bounds on Gap(μ).

In the following, we derive an upper bound in detail, a lower bound can be derived in a similar fashion as explained in subsection 2.4.4. Main results are summarized in Theorems 3 and 4 in Section 2.4.4.

2.4.1 Decomposition Reformulation

We first describe a decomposition reformulation of the generic capacity expansion planning model. This reformulation separates capacity expansion decisions from operation decisions. For simplicity, we let x and y denote vectors $\{x_n\}_{n\in\mathcal{T}}$ and $\{\{y_{nk}\}_{k\in\mathcal{K}_{t_n}}\}_{n\in\mathcal{T}}$. The multistage

model Π_{MS} can be decomposed as follows.

$$v^{\text{MS}} = \min \left\{ \sum_{n\in\mathcal{T}} \sum_{k\in\mathcal{K}_{t_n}} p_n b_{nk}^\top y_{nk} + \sum_{i\in\mathcal{I}} V_i(y) \ : \ B_{nk} y_{nk} = d_{nk}, y_{nk} \in \mathbb{R}^J_+, \forall k \in \mathcal{K}_{t_n}, n \in \mathcal{T} \right\}, \tag{2.5}$$

where for each $i \in \mathcal{I}$,

$$V_i(y) = \min \left\{ \sum_{n\in\mathcal{T}} p_n a_{in} x_{in} \ : \ [A_{nk} y_{nk}]_i \le \sum_{m\in\mathcal{P}(n)} x_{im} \le u_i, x_{in} \in \mathbb{Z}_+, \forall k \in \mathcal{K}_{t_n}, n \in \mathcal{T} \right\}. \tag{2.6}$$

The above reformulation allows problem (2.6) to be solved for each type of technology individually, if operation decision y is given. Let $\{x^{\text{MLP}}, y^{\text{MLP}}\}$ be an optimal solution to the linear relaxation of the multistage model Π_{MS}, and let $\delta_{in} = \max_{k\in\mathcal{K}_{t_n}} \{[A_{nk} y^{\text{MLP}}_{nk}]_i\}$ for all $i \in \mathcal{I}$ and $n \in \mathcal{T}$. Since y^{MLP} is optimal in the LP relaxation of Π_{MS} but not necessarily in $\Pi_{\text{PA}}(\mu)$, and $\Pi_{\text{PA}}(\mu)$ is an integer program, we have

$$v^{\text{PA}}(\mu) \le \sum_{n\in\mathcal{T}} \sum_{k\in\mathcal{K}_{t_n}} p_n b_{nk}^\top y^{\text{MLP}}_{nk} + \sum_{i\in\mathcal{I}} o^{\text{P}}_i(\mu), \quad v^{\text{MS}} \ge \sum_{n\in\mathcal{T}} \sum_{k\in\mathcal{K}_{t_n}} p_n b_{nk}^\top y^{\text{MLP}}_{nk} + \sum_{i\in\mathcal{I}} v^{\text{M}}_i,$$

where

$$o^{\text{P}}_i(\mu) = \min \quad \sum_{n\in\mathcal{T}} p_n a_{in} x_{in} \tag{2.7a}$$

$$\text{s.t.} \quad \sum_{m\in\mathcal{P}(n)} x_{im} \ge \delta_{in}, \quad \forall\, n \in \mathcal{T} \tag{2.7b}$$

$$\sum_{m\in\mathcal{P}(n)} x_{im} \le u_i, \quad \forall\, n \in \mathcal{S}_T \tag{2.7c}$$

$$x_{in_1} = x_{in_2}, \quad \forall\, n_1, n_2 \in \mathcal{T}(m) \cap \mathcal{S}_t, m \in S_\mu, t > \mu \tag{2.7d}$$

$$x_{in} \in \mathbb{Z}_+, \quad \forall\, n \in \mathcal{T}, \tag{2.7e}$$

and

$$v_i^{\mathrm{M}} = \min \quad \left\{ \sum_{n \in \mathcal{T}} p_n a_{in} x_{in} \ : \ \text{s.t. (2.7b)} - \text{(2.7c)}, \ x_{in} \in \mathbb{R}_+, \quad \forall\, n \in \mathcal{T}. \right\} \tag{2.8}$$

Note that $o_i^{\mathrm{P}}(\mu)$ is the optimal investment cost of a partially adaptive, single-technology capacity expansion planning problem with fixed operation decisions y^{MLP}. Let $v_i^{\mathrm{P}}(\mu)$ denote the optimal value of its LP relaxation. v_i^{M} is the optimal investment cost of the LP relaxation of a multistage, single-technology problem with fixed operation decisions y^{MLP}. Moreover, u is an integral vector, thus $\delta_{in} \leq \lceil \delta_{in} \rceil \leq u_i$ for all $i \in \mathcal{I}$ and $n \in \mathcal{T}$. Since x_{in} takes nonnegative integer values in (2.7), the problem remains the same if the right-hand-side of constraints (2.7b) is rounded up to $\lceil \delta_{in} \rceil$.

Because of the decomposition reformulation, we can focus on single-technology problems, and bound the gap between the PA model and the multistage model by

$$\mathrm{Gap}(\mu) = v^{\mathrm{PA}}(\mu) - v^{\mathrm{MS}} \leq \sum_{i \in \mathcal{I}} (o_i^{\mathrm{P}}(\mu) - v_i^{\mathrm{M}}). \tag{2.9}$$

2.4.2 Some Useful Results for Single-technology Problems

*Reformulation of (**2.7**)*

In problem (2.7), constraints (2.7d) enforce two-stage approach after period μ at each possible outcome in that period. This is equivalent to combining some nodes into a single node after period μ, but still maintaining a tree structure in the capacity expansion decisions, as illustrated in Figure 2.1. Therefore, we can formulate a equivalent multistage model on the "compressed" tree as follows.

$$o_i^{\mathrm{P}}(\mu) = \min \quad \sum_{n \in \hat{\mathcal{T}}} \hat{p}_n \hat{a}_{in} x_{in} \tag{2.10a}$$

$$\text{s.t.} \quad \sum_{m \in \mathcal{P}(n)} x_{im} \geq \lceil \hat{\delta}_{in} \rceil, \quad \forall\, n \in \hat{\mathcal{T}} \tag{2.10b}$$

$$\sum_{m\in\mathcal{P}(n)} x_{im} \le u_i, \quad \forall\, n \in \mathcal{S}_T \tag{2.10c}$$

$$x_{in} \in \mathbb{Z}_+, \quad \forall\, n \in \hat{\mathcal{T}}, \tag{2.10d}$$

where $\hat{\mathcal{T}}$ is the compressed tree, for any node $n \in \hat{\mathcal{T}}$ such that $t_n \le \mu$, we have $\hat{p}_n = p_n$, $\hat{a}_{in} = a_{in}$, $\hat{\delta}_{in} = \delta_{in}$. For any node $n \in \hat{\mathcal{T}}$ such that $t_n > \mu$, let $U_n \subset \mathcal{T}$ consist of the nodes that are "compressed" to node n by constraints (2.7d), and $\hat{p}_n = \sum_{m\in U_n} p_m$, $\hat{a}_{in} = \sum_{m\in U_n} p_m a_{im}$, $\hat{\delta}_{in} = \max\{\delta_{im} : m \in U_n\}$.

*Totally unimodularity (TU) of (**2.10**)*

The following result shows that the feasible region of problem (2.10) is an integral polytope as long as the right-hand-side vector is integral. The boundedness follows directly from upper bound constraints (2.10c) and nonnegativity of x_{in}. As a result, the LP relaxation of problem (2.10) admits integer optimal solutions.

Proposition 1. *In problem* (2.10)*, the left-hand-side coefficient matrix is TU.*

Proof. Let us denote the left-hand-side coefficient matrix corresponding to constraints (2.10b) by D. The left-hand-side coefficient matrix corresponding to constraints (2.10c) are the same as some rows in $-D$. In fact, they correspond to each scenario (path) in the tree $\hat{\mathcal{T}}$. Therefore, it is sufficient to show that D is totally unimodular. We know that every entry of D is either 0 or 1. For each column j, there are exactly $|\hat{\mathcal{T}}(j)|$ 1's, in particular, $D_{ij} = 1$ if $i \in \hat{\mathcal{T}}(j)$. We can traverse the tree by depth-first-search, and rearrange the rows according to the sequence. After rearrangement, D is an interval matrix hence is totally unimodular (cf. Schrijver 1998). □

*Redundancy of constraints (**2.10c**)*

Since $\{\delta_{in}, i \in \mathcal{I}, n \in \mathcal{T}\}$ are defined by an optimal solution $\{x^{\text{MLP}}, y^{\text{MLP}}\}$ to the LP relaxation of multistage model, we can further simplify problem (2.10) by removing the

redundant constraints (2.10c).

Proposition 2. *If $\hat{p}_n \hat{a}_{in} > 0$ for all $n \in \hat{\mathcal{T}}$ and $i \in \mathcal{I}$, then in the LP relaxation of problem* (2.10)*, constraints* (2.10c) *are redundant.*

Proof. It is sufficient to show that any optimal solution to the problem

$$\min \left\{ \sum_{n \in \hat{\mathcal{T}}} \hat{p}_n \hat{a}_{in} x_{in} \quad \text{s.t.} \sum_{m \in \mathcal{P}(n)} x_{im} \geq \lceil \hat{\delta}_{in} \rceil, x_{in} \geq 0, \forall n \in \hat{\mathcal{T}} \right\} \tag{2.11}$$

satisfies constraints (2.10c). Let $\tilde{x}_i$ be an optimal solution to (2.11). Suppose there exists $n_0 \in \hat{\mathcal{T}}$ such that $\sum_{m \in \mathcal{P}(n_0)} \tilde{x}_{im} = w > u_i$. Recall that $u_i \geq \lceil \hat{\delta}_{in} \rceil$ for all $n \in \hat{\mathcal{T}}$, it follows that $\sum_{m \in \mathcal{P}(n_0)} \tilde{x}_{im} > \lceil \hat{\delta}_{in} \rceil$ for all $n \in \hat{\mathcal{T}}(n_0)$. If $\tilde{x}_{in_0} > 0$, since $\hat{p}_{n_0} \hat{a}_{in_0} > 0$, by optimality of $\tilde{x}_i$, we know there must be some $n_0' \in \hat{\mathcal{T}}(n_0)$, such that $\lceil \hat{\delta}_{in_0'} \rceil = w > u_i$, which contradicts the fact that $u_i \geq \lceil \hat{\delta}_{in} \rceil$ for all $n \in \hat{\mathcal{T}}$. If $\tilde{x}_{in_0} = 0$, we traverse back along $\mathcal{P}(n_0)$ to find the first node n_0'' with $\tilde{x}_{in_0''} > 0$. Notice such a node must exist since $\sum_{m \in \mathcal{P}(n_0)} \tilde{x}_{im} > 0$. It follows that $\sum_{m \in \mathcal{P}(n_0'')} \tilde{x}_{im} = w > u_i$. We go back to the first case. Therefore, the result holds. □

As a remark, the result in Proposition 2 also applies to the LP relaxation of problem (2.7), as well as problem (2.8).

2.4.3 An Upper Bound on $o_i^{\mathrm{P}}(\mu) - v_i^{\mathrm{M}}$

There are two main steps in obtaining an upper bound on $o_i^{\mathrm{P}}(\mu) - v_i^{\mathrm{M}}$.

- *Step 1.* We show that $o_i^{\mathrm{P}}(\mu) \leq v_i^{\mathrm{P}}(\mu) + C$, where C is some constant that is dependent on the input data and $\{\delta_{in}\}_{n \in \mathcal{T}}$.
- *Step 2.* We derive upper and lower bounds for $v_i^{\mathrm{P}}(\mu)$ and v_i^{M}.

Then an upper bound for $o_i^{\mathrm{P}}(\mu) - v_i^{\mathrm{M}}$ can be expressed as "upper bound of $v_i^{\mathrm{P}}(\mu)$ − lower bound of $v_i^{\mathrm{M}} + C$".

Step 1 With previous results for the single-technology problems, we have the following result.

Proposition 3. *$o_i^P(\mu) \le v_i^P(\mu) + a_{i1} \cdot \lambda_i$, where $\lambda_i = \max_{n\in\mathcal{T}}\{\lceil\delta_{in}\rceil - \delta_{in}\}$. If $\{\delta_{in}\}_{n\in\mathcal{T}}$ are all integers, the inequality is tight.*

Proof. In fact, with linear programming duality, Proposition 1 and 2, we have

$$
\begin{aligned}
o_i^{\mathrm{P}}(\mu) &= \min\left\{\sum_{n\in\hat{\mathcal{T}}}\hat{p}_n\hat{a}_{in}x_{in} \quad \text{s.t.} \sum_{m\in\mathcal{P}(n)} x_{im} \ge \hat{\delta}_{in},\ \sum_{m\in\mathcal{P}(n)} x_{im} \le u_i,\ x_{in}\in\mathbb{Z}_+,\ \forall n\in\hat{\mathcal{T}}\right\} \\
&\overset{\text{(i)}}{=} \min\left\{\sum_{n\in\hat{\mathcal{T}}}\hat{p}_n\hat{a}_{in}x_{in} \quad \text{s.t.} \sum_{m\in\mathcal{P}(n)} x_{im} \ge \lceil\hat{\delta}_{in}\rceil,\ \sum_{m\in\mathcal{P}(n)} x_{im} \le u_i,\ x_{in}\in\mathbb{R}_+,\ \forall n\in\hat{\mathcal{T}}\right\} \\
&\overset{\text{(ii)}}{=} \min\left\{\sum_{n\in\hat{\mathcal{T}}}\hat{p}_n\hat{a}_{in}x_{in} \quad \text{s.t.} \sum_{m\in\mathcal{P}(n)} x_{im} \ge \lceil\hat{\delta}_{in}\rceil,\ x_{in}\in\mathbb{R}_+,\ \forall n\in\hat{\mathcal{T}}\right\} \\
&\overset{\text{(iii)}}{=} \max\left\{\sum_{n\in\hat{\mathcal{T}}}\lceil\hat{\delta}_{in}\rceil\pi_{in} \quad \text{s.t.} \sum_{m\in\hat{\mathcal{T}}(n)} \pi_{im} \le \hat{p}_n\hat{a}_n,\ \pi_{in}\in\mathbb{R}_+,\ \forall n\in\hat{\mathcal{T}}\right\} \\
&= \max\left\{\sum_{n\in\hat{\mathcal{T}}}(\hat{\delta}_{in} + \lceil\hat{\delta}_{in}\rceil - \hat{\delta}_{in})\pi_{in} \quad \text{s.t.} \sum_{m\in\hat{\mathcal{T}}(n)} \pi_{im} \le \hat{p}_n\hat{a}_n,\ \pi_{in}\in\mathbb{R}_+,\ \forall n\in\hat{\mathcal{T}}\right\} \\
&\le \max\left\{\sum_{n\in\hat{\mathcal{T}}}\hat{\delta}_{in}\pi_{in} \quad \text{s.t.} \sum_{m\in\hat{\mathcal{T}}(n)} \pi_{im} \le \hat{p}_n\hat{a}_n,\ \pi_{in}\in\mathbb{R}_+,\ \forall n\in\hat{\mathcal{T}}\right\} \\
&\quad + \max\left\{\sum_{n\in\hat{\mathcal{T}}}\pi_{in} \quad \text{s.t.} \sum_{m\in\hat{\mathcal{T}}(n)} \pi_{im} \le \hat{p}_n\hat{a}_n,\ \pi_{in}\in\mathbb{R}_+,\ \forall n\in\hat{\mathcal{T}}\right\} \cdot \max_{n\in\mathcal{T}}\{\lceil\hat{\delta}_{in}\rceil - \hat{\delta}_{in}\} \\
&\overset{\text{(iv)}}{=} \min\left\{\sum_{n\in\hat{\mathcal{T}}}\hat{p}_n\hat{a}_{in}x_{in} \quad \text{s.t.} \sum_{m\in\mathcal{P}(n)} x_{im} \ge \hat{\delta}_{in},\ x_{in}\in\mathbb{R}_+,\ \forall n\in\hat{\mathcal{T}}\right\} \\
&\quad + \min\left\{\sum_{n\in\hat{\mathcal{T}}}\hat{p}_n\hat{a}_{in}x_{in} \quad \text{s.t.} \sum_{m\in\mathcal{P}(n)} x_{im} \ge 1,\ x_{in}\in\mathbb{R}_+,\ \forall n\in\hat{\mathcal{T}}\right\} \cdot \max_{n\in\mathcal{T}}\{\lceil\hat{\delta}_{in}\rceil - \hat{\delta}_{in}\} \\
&\overset{\text{(v)}}{\le} v_i^{\mathrm{P}}(\mu) + a_{i1}\cdot\lambda_i. \qquad (2.12)
\end{aligned}
$$

Specifically, (i) follows from Proposition 1; (ii) follows from Proposition 2; (iii) and (iv)

follow from linear programming duality; (v) follows from Proposition 2, the definition of $\{\hat{\delta}_{in}\}_{n\in\hat{T}}$, the fact that $p_1 = 1$, and the optimal solution to a single-technology GEP problem with demand 1 at each node is nothing but building one generator at the beginning of the planning horizon. The tightness of the inequality follows from Proposition 1. □

Step 2 Before presenting lower and upper bounds for $v_i^{\mathrm{P}}(\mu)$ and v_i^{M}, we define some useful parameters and show their relations. For simplicity, we omit the index i in this step.

Let us define

$$\delta^{(\mu-)} := \sum_{n\in S_{\mu-1}} p_n \max_{m\in\mathcal{P}(n)}\{\delta_m\}, \quad \delta^{(\mu)} := \sum_{n\in S_\mu} p_n \max_{m\in\mathcal{P}(n)\cup\mathcal{T}(n)}\{\delta_m\};$$

$$\underline{a}_{\mu-} := \min_{n\in\mathcal{T}:t_n\le\mu}\{a_n\}, \quad \underline{a}_{\mu+} := \min_{n\in\mathcal{T}:t_n\ge\mu}\{a_n\}, \quad a_* = \min_{n\in\mathcal{T}}\{a_n\};$$

$$\bar{a}_{\mu-} := \max_{n\in\mathcal{T}:t_n\le\mu}\{a_n\}, \quad \bar{a}_{\mu+} := \max_{n\in\mathcal{T}:t_n\ge\mu}\{a_n\}, \quad a^* = \max_{n\in\mathcal{T}}\{a_n\}.$$

If $\mu = 1$, we have $\delta^{(1-)} = 0$, $\delta^{(1)} = \max_{n\in\mathcal{T}}\{\delta_n\}$, and $\underline{a}_{1-} = \bar{a}_{1-} = a_1$, $\underline{a}_{1+} = a_*$, $\bar{a}_{1+} = a^*$; if $\mu = T$, then $\delta^{(T)} = \sum_{n\in\mathcal{S}_T} p_n \max_{m\in\mathcal{P}(n)}\{\delta_m\}$, and $\underline{a}_{T-} = a_*$, $\bar{a}_{T-} = a^*$. It is easy to see that as μ increases from 1 to T, $\delta^{(\mu-)}$, $\underline{a}_{\mu+}$ and $\bar{a}_{\mu-}$ are monotone increasing while $\delta^{(\mu)}$, $\underline{a}_{\mu-}$ and $\bar{a}_{\mu+}$ are monotone decreasing. One can treat $\{\delta_n\}_{n\in\mathcal{T}}$ as the "demand" in the single-technology problem, then $\delta^{(1)}$ is the largest demand across the entire scenario tree, and $\delta^{(T)}$ is the average of the maximum demand in each scenario (path). The following proposition reveals the relation between these δ's.

Proposition 4. *For any $\mu \in \{1,\dots,T\}$, the following relation holds, $\delta^{(\mu-)} \le \delta^{(T)} \le \delta^{(\mu)} \le \delta^{(1)}$.*

Proof. Recall that in a scenario tree, the probability associated with a node equals the sums of probabilities of its children nodes. By definition, we have

$$\delta^{(\mu-)} = \sum_{n\in\mathcal{S}_{\mu-1}} p_n \max_{m\in\mathcal{P}(n)}\{\delta_m\}$$

$$= \sum_{n\in\mathcal{S}_{\mu-1}} \left(\sum_{k\in\mathcal{S}_T\cap\mathcal{T}(n)} p_k \max_{m\in\mathcal{P}(n)}\{\delta_m\} \right) \le \sum_{n\in\mathcal{S}_{\mu-1}} \left(\sum_{k\in\mathcal{S}_T\cap\mathcal{T}(n)} p_k \max_{m\in\mathcal{P}(k)}\{\delta_m\} \right)$$

$$(\delta^{(T)}) = \sum_{k\in\mathcal{S}_T} p_k \max_{m\in\mathcal{P}(k)}\{\delta_m\}$$

$$= \sum_{n\in\mathcal{S}_\mu} \sum_{k\in\mathcal{S}_T\cap\mathcal{T}(n)} p_k \max_{m\in\mathcal{P}(k)}\{\delta_m\} \le \sum_{n\in\mathcal{S}_\mu} \left(\sum_{k\in\mathcal{S}_T\cap\mathcal{T}(n)} p_k \right) \max_{m\in\mathcal{P}(n)\cup\mathcal{T}(n)}\{\delta_m\}$$

$$(\delta^{(\mu)}) = \sum_{n\in\mathcal{S}_\mu} p_n \max_{m\in\mathcal{P}(n)\cup\mathcal{T}(n)}\{\delta_m\} \le \sum_{n\in S_\mu} p_n\delta^{(1)} = \delta^{(1)}. \qquad \square$$

Now we derive lower and upper bounds for $v^{\mathrm{P}}(\mu)$, and the bounds for v^{M} can be attained by setting $\mu = T$. We briefly discuss the idea of finding these bounds:

- *Lower bound:* starting from an optimal solution to the LP relaxation of (2.7), relax the coefficients in the objective funtion to reach a lower bound for $v^{\mathrm{P}}(\mu)$;
- *Upper bound:* construct a feasible solution to the LP relaxation of (2.7), and the objective function value given by this solution yields an upper bound for $v^{\mathrm{P}}(\mu)$.

Specifically, we have the following theorem.

Theorem 1. $(\underline{a}_{\mu-} - a_*)\delta^{(\mu-)} + a_*\delta^{(\mu)} \le v^P(\mu) \le (\bar{a}_{\mu-} - \bar{a}_{\mu+})\delta^{(\mu-)} + \bar{a}_{\mu+}\delta^{(\mu)}$.

Proof. We change the notation of decision variables for capacity expansion decisions starting from period μ into a different representation. In particular, for any $n \in \mathcal{S}_\mu$ and $t > \mu$, $\{x_m : m \in \mathcal{T}(n) \cap \mathcal{S}_t\}$ share the same value, let $x_{n,t}$ denote the new variable that represents the common value of these variables. Given a feasible solution x to the LP relaxation of problem (2.7), for any $n \in S_{\mu-1}$, by feasibility we have

$$\sum_{m\in\mathcal{P}(n)} x_m \ge \max_{m\in\mathcal{P}(n)}\{\delta_m\} \Rightarrow \sum_{n\in\mathcal{S}_{\mu-1}} p_n \sum_{m\in\mathcal{P}(n)} x_m \ge \sum_{n\in\mathcal{S}_{\mu-1}} p_n \max_{m\in\mathcal{P}(n)}\{\delta_m\}$$

$$\Leftrightarrow \sum_{t=1}^{\mu-1} \sum_{k\in\mathcal{S}_t} \left(\sum_{m\in S_{\mu-1}\cap\mathcal{T}(k)} p_m \right) x_k \ge \delta_{\mu-}$$

$$\Leftrightarrow \sum_{t=1}^{\mu-1} \sum_{k \in \mathcal{S}_t} p_k x_k \geq \delta_{\mu-},$$

where the first equivalence follows from changing the summation sequence; and the second equivalence follows from the fact that $\sum_{m \in S_\mu \cap \mathcal{T}(k)} p_m = p_k$ for all $k \in \mathcal{T}$ such that $t_k < \mu$. In addition, for any $n \in \mathcal{S}_\mu$, by feasibility we have

$$\sum_{m \in \mathcal{P}(a(n))} x_m + \sum_{t=\mu}^{T} x_{n,t} \geq \max_{\mathcal{P}(n) \cup \mathcal{T}(n)} \{\delta_m\} \quad \Leftrightarrow \quad \sum_{t=\mu}^{T} x_{n,t} \geq \max_{\mathcal{P}(n) \cup \mathcal{T}(n)} \{\delta_m\} - \sum_{m \in \mathcal{P}(a(n))} x_m.$$

Then if x^* is an optimal solution to the LP relaxation of problem (2.7), we have

$$\begin{aligned}
v^{\mathrm{P}}(\mu) &= \sum_{n \in \mathcal{T}} p_n a_n x_n^* && (2.13)\\
&= \sum_{t=1}^{\mu-1} \sum_{n \in \mathcal{S}_t} p_n a_n x_n^* + \sum_{t=\mu}^{T} \sum_{n \in \mathcal{S}_t} p_n a_n x_n^* \\
&\geq \underline{a}_{\mu-} \sum_{t=1}^{\mu-1} \sum_{n \in \mathcal{S}_t} p_n x_n^* + a_* \sum_{t=\mu}^{T} \sum_{n \in \mathcal{S}_t} p_n x_n^* \\
&= \underline{a}_{\mu-} \sum_{t=1}^{\mu-1} \sum_{n \in \mathcal{S}_t} p_n x_n^* + a_* \sum_{n \in \mathcal{S}_\mu} p_n \sum_{t=\mu}^{T} x_{n,t}^* \\
&\geq \underline{a}_{\mu-} \sum_{t=1}^{\mu-1} \sum_{n \in \mathcal{S}_t} p_n x_n^* + a_* \sum_{n \in \mathcal{S}_\mu} p_n \left(\max_{\mathcal{P}(n) \cup \mathcal{T}(n)} \{\delta_m\} - \sum_{m \in \mathcal{P}(a(n))} x_m^* \right) \\
&= (\underline{a}_{\mu-} - a_*) \sum_{t=1}^{\mu-1} \sum_{n \in \mathcal{S}_t} p_n x_n^* + a_* \sum_{n \in S_\mu} p_n \max_{\mathcal{P}(n) \cup \mathcal{T}(n)} \{\delta_m\} \\
&\geq (\underline{a}_{\mu-} - a_*) \delta^{(\mu-)} + a_* \delta^{(\mu)}, && (2.14)
\end{aligned}$$

where the last inequality follows from $\underline{a}_{\mu-} \geq a_*$ and the definitions of $\delta^{(\mu)}$ and $\delta^{(\mu-)}$.

Next, we consider a feasible solution $\hat{x}$ to the LP relaxation of problem (2.7). For any $n \in \mathcal{T}$ such that $t_n \leq \mu - 1$, let $\hat{x}_n = \max\{\delta_m : m \in \mathcal{P}(n)\} - \max\{\delta_m : m \in \mathcal{P}(a(n))\}$, and $\max\{\delta_m : m \in \mathcal{P}(a(1))\} = 0$; for any $n \in S_\mu$, $t \geq \mu$, let $\hat{x}_{n,t} = \max\{\delta_m : m \in$

$\mathcal{P}(a(n)) \cup \mathcal{T}(n,t)\} - \max\{\delta_m : m \in \mathcal{P}(a(n)) \cup \mathcal{T}(n,t-1)\}$. Then we have

$$
\begin{aligned}
v^{\mathrm{P}}(\mu) &\leq \sum_{n\in\mathcal{T}} p_n a_n \hat{x}_n \qquad (2.15)\\
&= \sum_{t=1}^{\mu-1}\sum_{n\in\mathcal{S}_t} p_n a_n \hat{x}_n + \sum_{t=\mu}^{T}\sum_{n\in\mathcal{S}_t} p_n a_n \hat{x}_n \\
&\leq \bar{a}_{\mu-}\sum_{t=1}^{\mu-1}\sum_{n\in\mathcal{S}_t} p_n \hat{x}_n + \bar{a}_{\mu+}\sum_{t=\mu}^{T}\sum_{n\in\mathcal{S}_t} p_n \hat{x}_n \\
&= \bar{a}_{\mu-}\sum_{t=1}^{\mu-1}\sum_{n\in\mathcal{S}_t} p_n \hat{x}_n + \bar{a}_{\mu+}\sum_{n\in\mathcal{S}_\mu} p_n \sum_{t=\mu}^{T} \hat{x}_{n,t} \\
&= \bar{a}_{\mu-}\sum_{t=1}^{\mu-1}\sum_{n\in\mathcal{S}_t} p_n \left(\max_{m\in\mathcal{P}(n)}\{\delta_m\} - \max_{m\in\mathcal{P}(a(n))}\{\delta_m\}\right) \\
&\quad + \bar{a}_{\mu+}\sum_{n\in\mathcal{S}_\mu} p_n \sum_{t=\mu}^{T}\left(\max_{m\in\mathcal{P}(a(n))\cup\mathcal{T}(n,t)}\{\delta_m\} - \max_{m\in\mathcal{P}(a(n))\cup\mathcal{T}(n,t-1)}\{\delta_m\}\right) \\
&= \bar{a}_{\mu-}\sum_{n\in\mathcal{S}_{\mu-1}} p_n \max_{m\in\mathcal{P}(n)}\{\delta_m\} + \bar{a}_{\mu+}\sum_{n\in\mathcal{S}_\mu} p_n \left(\max_{m\in\mathcal{P}(a(n))\cup\mathcal{T}(n)}\{\delta_m\} - \max_{m\in\mathcal{P}(a(n))}\{\delta_m\}\right) \\
&= (\bar{a}_{\mu-} - \bar{a}_{\mu+})\sum_{n\in\mathcal{S}_{\mu-1}} p_n \max_{m\in\mathcal{P}(n)}\{\delta_m\} + \bar{a}_{\mu+}\sum_{n\in S_\mu} p_n \max_{m\in\mathcal{P}(n)\cup\mathcal{T}(n)}\{\delta_m\} \\
&= (\bar{a}_{\mu-} - \bar{a}_{\mu+})\,\delta^{(\mu-)} + \bar{a}_{\mu+}\delta^{(\mu)}, \qquad (2.16)
\end{aligned}
$$

where the third to last equality follows from the fact that the probability of node n equals to the sum of probabilities of its children nodes. □

Setting $\mu = T$ and applying Proposition 4, we immediately have the following corollary.

Corollary 1. $a_*\delta^{(T)} \leq v^M \leq a^*\delta^{(T)}$.

Suppose that the cost parameters a_n are nearly constant, that is, $a^* \approx a_* \approx \underline{a}_{\mu-} \approx \underline{a}_{\mu+} \approx \bar{a}_{\mu-} \approx \bar{a}_{\mu+} \approx a$, then we have $v^{\mathrm{P}}(\mu) \approx a\delta^{(\mu)}$ and $v^{\mathrm{M}} \approx a\delta^{(T)}$. As μ increases from 1 to T, this approximate value of $v^{\mathrm{P}}(\mu) - v^{\mathrm{M}}$ decreases from $a\left(\delta^{(1)} - \delta^{(T)}\right)$ to 0. Combining results in these two steps, we obtain an upper bound of $o^{\mathrm{P}}(\mu) - v^{\mathrm{M}}$, summarized in the following theorem.

Theorem 2. $o^P(\mu) - v^M \leq \left[(\bar{a}_{\mu-} - \bar{a}_{\mu+})\delta^{(\mu-)} + \bar{a}_{\mu+}\delta^{(\mu)} - a_*\delta^{(T)}\right] + a_1 \cdot \lambda_i$, *where* $\lambda_i = \max_{n\in\mathcal{T}}\{\lceil\delta_{in}\rceil - \delta_{in}\}$.

2.4.4 Upper and Lower Bounds for Gap(μ)

Upper bound As discussed before, aggregating the bound in Theorem 2 for each type of capacity expansion technology yields an upper bound for Gap(μ).

Theorem 3. *Let y^{MLP} be the operation level decisions in an optimal solution to the linear relaxation of multistage model Π_{MS}. For each type of expansion technology $i \in \mathcal{I}$, let $\delta_{in} = \max_{k\in\mathcal{K}_{t_n}}\{[A_{nk}y^{MLP}_{nk}]_i\}$, and $\lambda_i = \max_{n\in\mathcal{T}}\{\lceil\delta_{in}\rceil - \delta_{in}\}$. We further define*

$$a_{i,*} = \min_{n\in\mathcal{T}}\{a_{in}\}, \quad \bar{a}_{i,\mu-} = \max_{n\in\mathcal{T}:t_n\leq\mu}\{a_{in}\}, \quad \bar{a}_{i,\mu+} = \max_{n\in\mathcal{T}:t_n\geq\mu}\{a_{in}\},$$
$$\delta_i^{(\mu-)} = \sum_{n\in\mathcal{S}_{\mu-1}} p_n \max_{m\in\mathcal{P}(n)}\{\delta_{im}\}, \quad \delta_i^{(\tau)} = \sum_{n\in\mathcal{S}_\tau} p_n \max_{m\in\mathcal{P}(n)\cup\mathcal{T}(n)}\{\delta_{im}\},$$

for $\tau = \mu, T$. Then for a capacity expansion planning problem with multiple technologies,

$$\textit{Gap}(\mu) \leq \sum_{i\in\mathcal{I}}\left[(\bar{a}_{i,\mu-} - \bar{a}_{i,\mu+})\delta_i^{(\mu-)} + \bar{a}_{i,\mu+}\delta_i^{(\mu)} - a_{i,*}\delta_i^{(T)}\right] + \sum_{i\in\mathcal{I}} a_{i1}\cdot\lambda_i. \tag{2.17}$$

Lower bound Now suppose we start the entire analysis with an optimal solution $\{x^{\text{PLP}}, y^{\text{PLP}}\}$ to the LP relaxation of PA model $\Pi_{\text{PA}}(\mu)$, and let $\gamma_{in} = \max_{k\in\mathcal{K}_{t_n}}\{[A_{nk}y^{\text{PLP}}_{nk}]_i\}$ for all $i \in \mathcal{I}$ and $n \in \mathcal{T}$. Then we have

$$v^{\text{PA}}(\mu) \geq \sum_{n\in\mathcal{T}}\sum_{k\in\mathcal{K}_{t_n}} p_n b^\top_{nk} y^{\text{PLP}}_{nk} + \sum_{i\in\mathcal{I}} v_i^{\text{P}}(\mu), \quad v^{\text{MS}} \leq \sum_{n\in\mathcal{T}}\sum_{k\in\mathcal{K}_{t_n}} p_n b^\top_{nk} y^{\text{PLP}}_{nk} + \sum_{i\in\mathcal{I}} o_i^{\text{M}},$$

where $v_i^{\text{P}}(\mu)$ and o_i^{M} are defined similarly as (2.7) and (2.8) with δ_{in} substituted by γ_{in}. Applying similar techniques as in Theorem 3, we can obtain

$$v^{\text{PA}}(\mu) - v^{\text{MS}} \geq \sum_{i\in\mathcal{I}}(v_i^{\text{P}}(\mu) - o_i^{\text{M}}) \geq \sum_{i\in\mathcal{I}}[v_i^{\text{P}}(\mu) - v_i^{\text{M}} - a_{i1}\cdot\eta_i], \tag{2.18}$$

where $\eta_i = \max_{n\in\mathcal{T}}\{\lceil\gamma_{in}\rceil - \gamma_{in}\}$. Putting Theorem 1, Corollary 1, and (2.18) together, the following result follows immediately.

Theorem 4. *Let y^{PLP} be the operation level decisions in an optimal solution to the linear relaxation of PA model $\Pi_{PA}(\mu)$. For each type of generator $i \in \mathcal{I}$, let $\gamma_{in} = \max_{k\in\mathcal{K}_{t_n}}\{[A_{nk}y^{PLP}_{nk}]_i\}$, and $\eta_i = \max_{n\in\mathcal{T}}\{\lceil\gamma_{in}\rceil - \gamma_{in}\}$. We further define*

$$a_i^* = \max_{n\in\mathcal{T}}\{a_{in}\}, \quad a_{i,*} = \min_{n\in\mathcal{T}}\{a_{in}\}, \quad \underline{a}_{i,\mu-} = \max_{n\in\mathcal{T}:t_n\le\mu}\{a_{in}\},$$
$$\gamma_i^{(\mu-)} = \sum_{n\in\mathcal{S}_{\mu-1}} p_n \max_{m\in\mathcal{P}(n)}\{\gamma_{im}\}, \quad \gamma_i^{(\tau)} = \sum_{n\in\mathcal{S}_\tau} p_n \max_{m\in\mathcal{P}(n)\cup\mathcal{T}(n)}\{\gamma_{im}\},$$

for $\tau = \mu, T$. Then for a capacity expansion planning problem with multiple technologies,

$$\textit{Gap}(\mu) \ge \sum_{i\in\mathcal{I}}\left[(\underline{a}_{i,\mu-} - a_{i,*})\gamma_i^{(\mu-)} + a_{i,*}\gamma_i^{(\mu)} - a_i^*\gamma_i^{(T)}\right] - \sum_{i\in\mathcal{I}} a_{i1}\cdot\eta_i. \tag{2.19}$$

The bounds in Theorem 3 and 4 are dependent on the input data, in particular the investment cost a, and optimal solutions to the LP relaxation of Π_{MS} and $\Pi_{\text{PA}}(\mu)$. Note that these bounds could be weak when investment costs in the first planning period are very large. For general cost structures, however, decision makers can choose a value of μ within $[1, T]$ and calculate these bounds. If these bounds indicate that the current PA model with parameter μ is a good enough approximation to the multistage model, then we can just solve the PA model without bearing addition computational effort. If the bounds are still not appealing, a larger value of μ should be considered. The following small example demonstrates the possibility that the bounds in Theorem 3 and Theorem 4 could be useful in assessing the critical values for μ.

Example 1. Consider a single technology problem. The underlying scenario tree has three stages and each node has two branches with equal probability. Let $\mathcal{S}_1 = \{1\}$, $\mathcal{S}_2 = \{2, 3\}$, $\mathcal{S}_3 = \{4, 5, 6, 7\}$. Assume investment cost $a_1 = 10$, $a_2 = \cdots = a_7 = 8$, and unit generation cost is 1. Let the demand be $d_1 = 1$, $d_2 = 3$, $d_3 = 5$, $d_4 = 4$, $d_5 = 5$, $d_6 = 5$, $d_7 = 6$. We

assume that one unit of demand requires one unit of generator. Thus solving a multistage problem yields an optimal cost of 71, and solving a PA model with $\mu = 2$ yields an optimal cost of 75, thus $Gap(2) = 4$. According to (2.17) and (2.19), we have $4 \leq Gap(2) \leq 6$, the lower bound is actually tight.

2.5 An Approximation Algorithm for Solving Π_{MS}

In this section, we first propose an approximation algorithm based on PA, which recursively traverses the scenario tree and computes a solution to the multistage stochastic optimization model. Then, we identify sufficient conditions under which the obtained solution is indeed optimal for the multistage model.

2.5.1 Algorithm Description

The key idea of the proposed algorithm is to traverse the scenario tree $\mathcal{T}$ and at each node $n \in \mathcal{T}$ solve a PA model on the subtree $\mathcal{T}(n)$, then use the solutions of these PA models to synthesize a multistage solution. To be more precise, let us first introduce some notations. We denote the PA model (2.2) formulated on the subtree $\mathcal{T}(n)$ as $\Pi_{\text{PA}}(\mu, n)$ and $\Pi_{\text{PA}}(\mu)$ always means $\Pi_{\text{PA}}(\mu, 1)$. Also denote the optimal expansion and generation decision of $\Pi_{\text{PA}}(\mu, n)$ as $z^{\mu,n} := \{x_m^{\mu,n}, \{y_{mk}^{\mu,n}\}_{k\in\mathcal{K}_m}\}_{m\in\mathcal{T}(n)}$. A *topological ordering* $\sigma(\mathcal{T})$ on the nodes of a scenario tree $\mathcal{T}$ is a linear ordering such that for every edge (u, v) from parent node u to child node v, u comes before v in the ordering $\sigma(\mathcal{T})$. An important implication is that, when the nodes of the tree are traversed in a topological ordering, a node n is always visited *after* all its ancestor nodes on the path $\mathcal{P}(n)$ from the root node to n are visited.

The proposed algorithm visits each node $n \in \mathcal{T}$ in the order $\sigma(\mathcal{T})$ and solves the PA model $\Pi_{\text{PA}}(\mu, n)$ on the subtree $\mathcal{T}(n)$ rooted at node n. In this process, the root node's optimal solution $z_n^{\mu,n}$ of $\Pi_{\text{PA}}(\mu, n)$ is recorded. Since $\sigma(\mathcal{T})$ is a topological ordering, at node n, all its ancestors k on the path $\mathcal{P}(n)$ have already been traversed by the algorithm, we can use the obtained generation capacity solutions $\{x_m^{\mu,m}\}_{m\in\mathcal{P}(n)}$ of these ancestors to

compute the initial installed generation capacity for the $\Pi_{\text{PA}}(\mu, n)$ problem (cf. constraints (2.2b) in model (2.2)). After the entire tree $\mathcal{T}$ is traversed in this way, the algorithm outputs a solution $\{z_n^{\mu,n}\}_{n=1}^N$, in which the expansion decision has a fully adaptive structure over the entire planning horizon. This procedure is summarized in Algorithm 1.

Algorithm 1 :: Partially Adaptive Recursive Update

Input: A topological ordering $\sigma(\mathcal{T}) = \{1, 2, \ldots, N\}$ of all the nodes in $\mathcal{T}$, a critical time μ, and the initial installed capacity $u_0 = 0$.

1: Solve the PA model $\Pi_{\text{PA}}(\mu, 1)$ at root node 1. Denote the optimal solution as $\{x^{\mu,1}, y^{\mu,1}\}$.
2: Update $x_1 \leftarrow x_1^{\mu,1}$ and $y_1 \leftarrow y_1^{\mu,1}$.
3: **for** $n = 2, \ldots, N$ **do**
4: Solve $\Pi_{\text{PA}}(\mu, n)$ on $\mathcal{T}(n)$ with initial installed capacity computed from x_k for $k \in \mathcal{P}(a(n))$. Denote its optimal solution as $\{x^{\mu,n}, y^{\mu,n}\}$.
5: Update $x_n \leftarrow x_n^{\mu,n}$ and $y_n \leftarrow y_n^{\mu,n}$.
6: **end for**
7: **return** $\{x_n, y_n\}_{n=1}^N$.

We would like to remark that the algorithm can use different critical times μ_n at different nodes n. The algorithm can also be terminated before visiting all the nodes in the scenario tree — the resulting solution is always a feasible multistage solution. Also note that there is flexibility in choosing the topological ordering $\sigma(\mathcal{T})$. We will look into this issue in the computation section.

2.5.2 Optimality of Algorithm **1** for a Special GEP Problem

In this subsection, we present a sufficient condition under which Algorithm 1 recovers an optimal solution for a multistage generation expansion planning problem.

Theorem 5. *For a multistage generation expansion planning problem* (2.4)*, if the instance satisfies the following conditions:*

i) the unit investment costs of each type of generators are stationary, i.e., in (2.1a)*,* $c_{it_n} = c_i$ *for all* $i \in \mathcal{I}$ *and* $n \in \mathcal{T}$*;*

ii) all generators share the same unit generation cost, i.e., in (2.1a)*,* $b_{ink} = b_{nk}$ *for all* $k \in \mathcal{K}_{t_n}$ *and* $n \in \mathcal{T}$*;*

iii) demand must be satisfied by generation, i.e., no penalty is allowed;

then the solution returned by Algorithm 1 is optimal to the multistage problem.

To prove Theorem 5, we need the following lemma.

Lemma 1. *Suppose condition (i) in Theorem 5 is satisfied. Let* $\{x^*, y^*\}$ *be an optimal solution to problem* (2.2)*, then* $x^*_{i1} = \max\{\lceil [A_{1k}y^*_{1k}]_i \rceil : k \in \mathcal{K}_1\}$ *for all* $i \in \mathcal{I}$*.*

Proof. Suppose there exists $i_0 \in \mathcal{I}$ such that $x^*_{i_0 1} > \max\{\lceil [A_{1k}y^*_{1k}]_{i_0} \rceil : k \in \mathcal{K}_1\}$. Since $x^*_{i_0 1} \in \mathbb{Z}_+$, $x^*_{i_0 1} \geq \max\{\lceil [A_{1k}y^*_{1k}]_{i_0} \rceil : k \in \mathcal{K}_1\} + 1$. Let $\tilde{x}_{i_0 1} = x^*_{i_0 1} - 1$, $\tilde{x}_{i_0 n} = x^*_{i_0 n} + 1$ for all $n \in C(1)$, $\tilde{x}_{i1} = x^*_{i1}$ for all $i \neq i_0$, and $\tilde{x}_n = x^*_n$ for n such that $t_n \geq 3$, $\tilde{y}_n = y^*_n$ for $n \in \mathcal{T}$. It is clear that $\{\tilde{x}, \tilde{y}\}$ is still feasible, but it changes the total cost by $-c_{i_0} + \sum_{n \in C(1)} \frac{p_n c_{i_0}}{(1+r)} = -\frac{r}{1+r} c_{i_0} < 0$, where we use the fact that $\sum_{n \in C(1)} p_n = p_1 = 1$. This contradicts the optimality of $\{x^*, y^*\}$. □

Lemma 1 indicates that in every optimal solution to model (2.2), one would never build a generator of any type that is not used for generation at the first planning period.

Proof of Theorem 5. Suppose conditions (i)-(iii) hold, in the objective function of both multistage and PA models, the generation cost becomes

$$\sum_{n \in \mathcal{T}} \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}_{t_n}} \frac{p_n h_{nk} b_{ink}}{(1+r)^{t_n - 1}} v_{ink} = \sum_{n \in \mathcal{T}} \sum_{k \in \mathcal{K}_{t_n}} \frac{p_n b_{nk}}{(1+r)^{t_n - 1}} \sum_{i \in \mathcal{I}} v_{ink} = \sum_{n \in \mathcal{T}} \sum_{k \in \mathcal{K}_{t_n}} \frac{p_n b_{nk} d_{nk}}{(1+r)^{t_n - 1}},$$

which is a constant. This implies that in both multistage and PA models, the choice of generators to satisfy demand will only depend on the investment costs. Moreover, for any subproblem solved during the course of Algorithm 1, it follows from Lemma 1 that both multistage and PA model will expand the capacity in the most economic way to meet the current demand, but will not build any generator that is not used in the current period. In other words, multistage and PA models will make the same capacity expansion decisions

at the root node of the subtree corresponding to that subproblem. Therefore, the solution output by Algorithm 1 is optimal to multistage problem (2.4). □

In Theorem 5, the exactness of Algorithm 1 is established on the assumption that all generators share the same generation cost, which may seem restrictive. However, if all generators considered have similar generation costs, i.e., $b_{ink} \in [\bar{b}_{nk} - \epsilon, \bar{b}_{nk} + \epsilon]$ where ϵ is small for all $i \in \mathcal{I}$, one can use Algorithm 1 to obtain a multistage solution to the problem with generation cost $\{\bar{b}_{nk}\}_{n,k}$, and that solution will have an objective function value of at most $v^{\text{MS}} + \sum_{n\in\mathcal{T}} \sum_{k\in\mathcal{K}_{t_n}} \frac{p_n d_{nk} \epsilon}{(1+r)^{t_n-1}}$, where v^{MS} is the optimal value for the multistage problem with original generation costs. Thus, this result suggests that the algorithm produces nearly optimal solutions for a larger class of problems than proved in the theorem.

2.6 Computational Experiments

In this section, we present extensive computational experiments to evaluate the proposed partially adaptive stochastic model and Algorithm 1.

2.6.1 Experiment Data and Setup

All the data in the experiments are obtained from the real world data collected by Jin et al. (2011). In particular, the authors of Jin et al. (2011) collected hourly electricity demand data from years 1991 to 2007 for the Midwest region from the Midcontinent Independent System Operator (MISO), the natural gas price data from years 1970 to 2006 for the same region from the Energy Information Administration (EIA), and generation build cost data suggested by the Joint Coordinated System Planning Report 2008 (JCSP). The GEP model has the year 2008 as the reference year and a 10-year planning horizon.

The uncertainty in the GEP problem comes from two sources: the natural gas price and the electricity demand. Jin et al. (2011) verified that both of these stochastic processes can be reasonably modeled as geometric Brownian motions with high temporal correlation. Hourly electricity demand is aggregated into three types of sub-period: peak-load, medium-load,

and low-load, according to the load duration curve of the reference year. The following years' load duration curves are assumed to share the same structure as the reference year's with incremental demand growth. There are six types of generators available for capacity expansion, namely Base Load, Combined Cycle (CC), Combined Turbine (CT), Nuclear, Wind, and Integrated Gasification Combined Cycle (IGCC). Among these six types of generators, both CC and CT power plants are fueled by natural gas, which are subject to price uncertainty, and IGCC power plants are fueled by coal, whose price is usually quite stable and thus assumed known.

The 10-year scenario tree is generated by applying a nonlinear programming approach introduced in Høyland and Wallace (2001). In particular, the discrete samples and their associated probability structure for the scenario tree are constructed so that the approximate distribution matches as well as possible the desired statistical properties of the underlying continuous random variables, i.e. electricity demand and fuel prices. The number of branches at each node in the scenario tree is chosen to be 3 to balance the size of the tree and the accuracy of approximation (Jin et al. 2011). For our case, the resulting tree has in total $3^9 = 19,683$ scenarios. The total expected generator investment cost and operation cost are discounted at an annual rate of 8%.

When solving any mixed integer problem, the relative MIP optimality gap is set to 5×10^{-3}. Also, we impose a time limit of 5 hours (18000 sec) on solving any of the two-stage, multistage, and PA models. The same limit is applied to the total computation time of Algorithm 1 as well. Algorithm 1 is implemented in Python 2.7.8 with Gurobi Python interface and Gurobi 5.5.0 as the MIP solver. All numerical experiments are conducted on a Macbook Pro with 8G RAM and a 2.3 GHz Intel Core i5 processor.

2.6.2 Performance of PA Model

In the first set of experiments, we solve the PA model $\Pi_{\text{PA}}(\mu)$ for different planning horizons T and different critical times μ. The experiments aim to answer two questions: (1) What is

the value of the multistage model compared with the two-stage model? (2) How fast can the PA model be solved and how well does the PA model approximate the multistage model as μ increase? Recall again, for a fixed T, $\mu = 1$ corresponds to the two-stage model, and $\mu = T$ corresponds to the multistage model. Results are presented in Table 2.1.

Columns 1 and 2 in Table 2.1 show the length of the planning horizon of each instance and the value of μ, respectively. Column 3 presents two numbers: the left one is the best lower bound on the optimal cost of the $\Pi_{\text{PA}}(\mu)$ model found within the time limit, denoted as $v_L(\mu)$, and the right one is the cost of the best feasible solution for $\Pi_{\text{PA}}(\mu)$, denoted as $v_H(\mu)$. Column 4 computes the gap between these two numbers, namely $(v_H(\mu) - v_L(\mu))/v_L(\mu)$, which measures the quality of the solution of the $\Pi_{\text{PA}}(\mu)$ model. Column 5 computes lower and upper bounds on the gap between the $\Pi_{\text{PA}}(\mu)$ model and the multistage model. In particular, the left number in Column 5 is $(v_L(\mu) - v_H(T))/v_H(T)$, and the right one is $(v_H(\mu) - v_L(T))/v_L(T)$. For example, when $T = 6$, $\mu = 4$, the lower bound is $(6203.99 - 5630.84)/5630.84 \times 100\% = 10.18\%$, and the upper bound is $(6267.00 - 5555.77)/5555.77 \times 100\% = 12.80\%$. Column 6 reports the wall clock time of the MIP solver.

Table 2.1 provides answers to the above two questions. First, there is a significant value in solving the multistage model. In particular, comparing to the two-stage model, the multistage model reduces the total expected cost by more than 30% for all test instances (see Column 5). However, as Column 6 shows, the multistage models for larger instances are difficult to solve within the time limit. This implies that it may be worthwhile to approximate the multistage model by a PA model, which leads to the answer to the second question. Second, as μ increases, the performance of the $\Pi_{\text{PA}}(\mu)$ model improves quite significantly, especially for instances with larger horizon length. For example, for the $T = 9$ instance, solving $\mu = 4$ obtains a gap between PA and multistage models in the range of 17.11-21.84%, and solving $\mu = 7$ further shrinks the gap to no more than 7.23%. For the $T = 10$ instance, the $\Pi_{\text{PA}}(\mu)$ model with $\mu = 5$ obtains a gap no more than 18.50% higher than the

Table 2.1: Solving PA models on GEP instances

T	μ	MIPObj (million \$)	OptGap (%)	$v^{\text{PA}}/v^{\text{MS}} - 1$ (%)	Time (sec.)
3	1	[3065.34, 3079.21]	0.45	[43.34, 44.39]	0.03
	2	[2426.14, 2435.36]	0.39	[13.45, 14.20]	0.06
	3	[2132.58, 2138.57]	0.28	0	0.08
4	1	[4598.61, 4598.61]	0.00	[43.70, 44.42]	0.17
	2	[4046.02, 4066.33]	0.50	[26.43, 27.70]	2.27
	3	[3436.09, 3453.31]	0.50	[7.37, 8.45]	5.45
	4	[3184.19, 3200.11]	0.50	0	4.11
5	1	[6225.63, 6250.71]	0.40	[42.92, 44.57]	0.26
	2	[5667.38, 5685.59]	0.32	[30.11, 31.50]	26.46
	3	[5029.97, 5072.60]	0.85	[15.47, 17.32]	18004.46
	5	[4323.61, 4355.93]	0.75	0	18002.52
6	1	[8018.62, 8055.41]	0.40	[42.41, 44.99]	2.73
	2	[7360.44, 7397.24]	0.50	[30.72, 33.15]	16.56
	4	[6203.99, 6267.00]	1.02	[10.18, 12.80]	18003.85
	6	[5555.77, 5630.84]	1.35	0	18003.95
7	1	[9889.73, 9927.02]	0.38	[41.02, 43.57]	9.72
	2	[9199.12, 9236.99]	0.41	[31.17, 33.59]	13.86
	4	[8033.43, 8097.94]	0.80	[14.55, 17.11]	18010.35
	7	[6914.58, 7013.13]	1.43	0	18002.95
8	1	[11825.95, 11884.75]	0.50	[39.41, 42.37]	86.82
	2	[11135.28, 11189.00]	0.48	[31.27, 34.04]	309.86
	5	[9401.26, 9510.03]	1.16	[10.83, 13.92]	18000.91
	8	[8347.79, 8482.69]	1.62	0	18000.30
9	1	[13717.37, 13735.28]	0.13	[35.56, 40.14]	343.97
	2	[13110.29, 13174.39]	0.49	[29.56, 34.42]	1261.38
	4	[11849.82, 11941.19]	0.77	[17.11, 21.84]	18001.53
	7	[10367.49, 10509.44]	1.37	[2.46, 7.23]	18017.75
	9	[9801.03, 10118.72]	3.24	0	18007.46
10	1	[15606.96, 15680.86]	0.47	[31.42, 38.82]	3730.84
	2	[14998.67, 15087.53]	0.59	[26.30, 33.57]	18002.23
	5	[13225.33, 13385.83]	1.21	[11.36, 18.50]	18020.00
	8	[11793.87, 12127.18]	2.83	[0.00, 7.36]	18004.19
	10	[11295.61, 11875.77]	5.14	0	18083.68

multistage model. Furthermore, for all the test instances, we observe that the PA model with a mid-range value of μ already decreases the gap of the two-stage model ($\mu = 1$) by

more than 50%. This suggests the benefit of solving PA models with small μ values. Indeed, from Column 6, we can see the PA models with large μ including the multistage model are computationally challenging to solve. Thus, recursively traversing the tree by solving PA models with small μ may provide a better multistage solution within a reasonable time limit. This is demonstrated by the next set of experiments.

2.6.3 Performance of Algorithm **1**

This second set of experiments evaluate Algorithm 1 in the following way. On a T-year scenario tree $\mathcal{T}$, Algorithm 1 solves all the PA models $\Pi_{\text{PA}}(\mu, n)$ with $\mu = 2$ for the subtree $\mathcal{T}(n)$, for each node n up to level T_0 in the tree $\mathcal{T}$, where $T_0 < T$. Denote the optimal solution of $\Pi_{\text{PA}}(\mu, n)$ as $z^{\mu,n} := \{x_m^{\mu,n}, \{y_{mk}^{\mu,n}\}_{k\in\mathcal{K}_m}\}_{m\in\mathcal{T}(n)}$. The final solution $\{z_n\}_{n\in\mathcal{T}}$ generated by Algorithm 1 is obtained as follows: $z_n := z_n^{\mu,n}$, for each node n up to level T_0, i.e. $t_n \leq T_0$, and $z_n := z_n^{\mu,m}$ for each node n with $t_n \geq T_0 + 1$, where $m = \mathcal{P}(n) \cap \mathcal{S}_{T_0}$. That is, the output solution z has a multistage expansion plan up to period $T_0 + 1$, and if $T_0 \leq T - 2$, z has a two-stage expansion plan from level $T_0 + 2$ to the end of the planning horizon T. In other words, z has the same decision structure in capacity expansion plan as the solution of a PA model $\Pi_{\text{PA}}(\mu, 1)$ for $\mu = T_0 + 1$, solved at root node 1.

Compared to directly solving the $\Pi_{\text{PA}}(T_0 + 1, 1)$ model, Algorithm 1 solves a sequence of much smaller problems of $\Pi_{\text{PA}}(2, n)$, therefore, may significantly save computation time. Also, the $\Pi_{\text{PA}}(T_0 + 1, 1)$ model may not be solvable to optimality within the time limit, whereas $\Pi_{\text{PA}}(2, n)$ can usually be solved to high precision quickly. In fact, we have already solved the $\Pi_{\text{PA}}(T_0 + 1, 1)$ model for various T and T_0 in the previous experiments. Most of these models for $T \geq 5$ and $T_0 \geq 2$ cannot be solved within the time limit of 5 hours as shown in Table 2.1.

Of course, we also need to evaluate how good the resulting solution z is compared to an optimal (or the best found) solution of $\Pi_{\text{PA}}(T_0 + 1, 1)$. Table 2.1 Column 3 presents the best lower bounds on the optimal costs of $\Pi_{\text{PA}}(T_0 + 1, 1)$ and the best feasible

solution found. Using the notation in the previous subsection, these costs are denoted as $v_L(T_0+1)$ and $v_H(T_0+1)$, respectively. The expected cost of the z solution is given as $v(z) := \sum_{n\in\mathcal{T}} p_n(a_n^\top x_n + \sum_{k\in\mathcal{K}_{t_n}} b_{nk}^\top y_{nk})$, where x and y are the capacity expansion and generation production components of z (cf. objective function (2.2a)). For each T and T_0, define OptGap $:= (v(z) - v_L(T_0+1))/v_L(T_0+1)$, which gives an upper bound on the gap between the Algorithm 1's solution z and the optimal $\Pi_{\text{PA}}(T_0+1,1)$ solution. Also define ImprvGap $:= (v_H(T_0+1) - v(z))/v_L(T_0+1)$, which measures how much the z solution improves on the best solution found by directly solving $\Pi_{\text{PA}}(T_0+1,1)$ (negative value means z solution is worse).

Table 2.2: Computation results of Algorithm 1

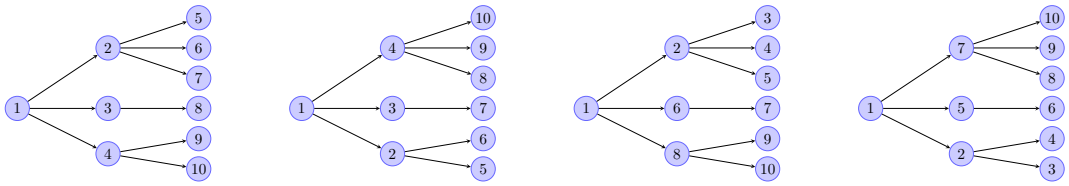
T	T_0	$v(z)$ (million \$)	OptGap (%)	ImprvGap (%)	PA time (sec.)	Alg 1 Time (sec.)
3	2	2138.57	0.28	0.00	0.08	0.35
4	2	3451.26	0.44	0.06	5.45	2.88
	3	3200.11	0.50	0.00	4.11	3.63
5	2	5073.68	0.87	−0.02	18004.46	27.27
	4	4353.10	0.68	0.07	18002.52	32.54
6	3	6255.76	0.83	0.18	18003.85	106.19
	5	5621.10	1.18	0.17	18003.95	119.86
7	3	8081.19	0.59	0.21	18010.35	360.23
	6	6987.25	1.05	0.37	18002.95	519.12
8	4	9463.10	0.66	0.50	18000.91	1324.39
	7	8440.01	1.10	0.51	18000.30	1833.46
9	3	11923.71	0.62	0.15	18001.53	1749.95
	6	10465.37	0.94	0.42	18017.75	6921.59
	8	9951.13	1.53	1.71	18007.46	7141.51
10	4	13302.09	0.58	0.63	18020.00	6838.51
	7	11934.25	1.19	1.64	18004.19	14771.03
	9	11462.33	1.48	3.66	18083.68	15446.97

Table 2.2 clearly shows that Algorithm 1 significantly reduces the computation time compared to directly solving the PA model (see Columns 6 and 7). Also as shown in Column

4 "OptGap", Algorithm 1 is able to produce a z solution that is within at most 1.5% from the optimal $\Pi_{\text{PA}}(T_0+1,1)$ solution. Furthermore, as shown in Column 5 "ImprvGap", the z solution constructed by Algorithm 1 actually improves over the best feasible solution found by directly solving the $\Pi_{\text{PA}}(T_0+1,1)$ for almost all instances, except for $T=5, T_0=2$ where the z solution is slightly worse by 0.02%. In fact, the improvement steadily increases as T and T_0 increase. For the 9-period problem, Algorithm 1 stopping at $T_0=8$ improves over the direct method by 1.71% in expected cost, while the computation time is only 40% of the latter. For the 10-period problem, Algorithm 1 stopping at $T_0=9$ improves over the direct method by 3.66% in expected cost with computation time reduced by 15%. Also note that for these two instances, the $\Pi_{\text{PA}}(T_0+1,1)$ model is the full multistage model. Therefore, the last instance shows that, by recursively traversing the tree with solving small PA models, the resulting solution is within 1.48% from the true optimal full multistage solution for the 10-year planning problem. The computation is done within 4.3 hours.

2.6.4 Effect of Different Node Orderings

As alluded to in Section 2.5.1, Algorithm 1 works for any topological ordering chosen for the nodes of the scenario tree. We distinguish four typical orderings, corresponding to breath-first-search (BFS) and depth-first-search (DFS) on the tree. For the simplicity of exposition, suppose for every node $n \in \mathcal{T}$, all its children nodes are positioned from top to bottom in the order of increasing demand. This is possible because the scenario tree is generated in a way that no two nodes sharing the same parent node have the same demand. Figure 2.2 illustrates four types of typological orderings on a simple example.



(a) `BFS_LOW` (b) `BFS_HIGH` (c) `DFS_LOW` (d) `DFS_HIGH`

Figure 2.2: Four topological orderings on a 3-period tree

In particular, we have the following four orderings and their corresponding search strategies:

(a) `BFS_LOW`: breath-first-search on the tree, and within each level, nodes are visited from top to bottom;

(b) `BFS_HIGH`: breath-first-search, and within each level, nodes are visited from bottom to top;

(c) `DFS_LOW`: depth-first-search, and selects the child node with *lowest* demand;

(d) `DFS_HIGH`: depth-first-search, and selects the child node with *highest* demand.

All the experiments in Section 2.6.3 are conducted with the `BFS_LOW` ordering. It is interesting to see the performance of Algorithm 1 under different node orderings, because traversing the scenario tree in different orders may reveal how fast the solution improves as we explore the node sequence, which may also shed some light on which nodes are important to explore, thus helps the decision maker decide whether to refine the current solution by branching through a specific node.

Figure 2.3 contains the cost improvement curves in an 8-year GEP instance under four different nodes orderings: `BFS_LOW`, `BFS_HIGH`, `DFS_LOW`, and `DFS_HIGH`. The upper-left figure shows how cost of the z solution is improved as allowed computation time limit increases. The other four figures correspond to the cost improvements as the number of visited nodes increases in these four orderings. Indeed, there is one subproblem associated with each node of the tree. The more subproblems Algorithm 1 solves, the lower cost the z solution incurs.

Figure 2.3 shows that, if there is no limit on computation time, i.e. Algorithm 1 is allowed to run until traversing all nodes in the scenario tree, then, as expected, there is no difference in the final policies corresponding to the above four different node orderings. The total computation times required are also almost the same. This is not surprising since all nodes are explored in a topological ordering, and the sets of subproblems solved are identical under each of these four orderings. The only difference is the sequence of solving

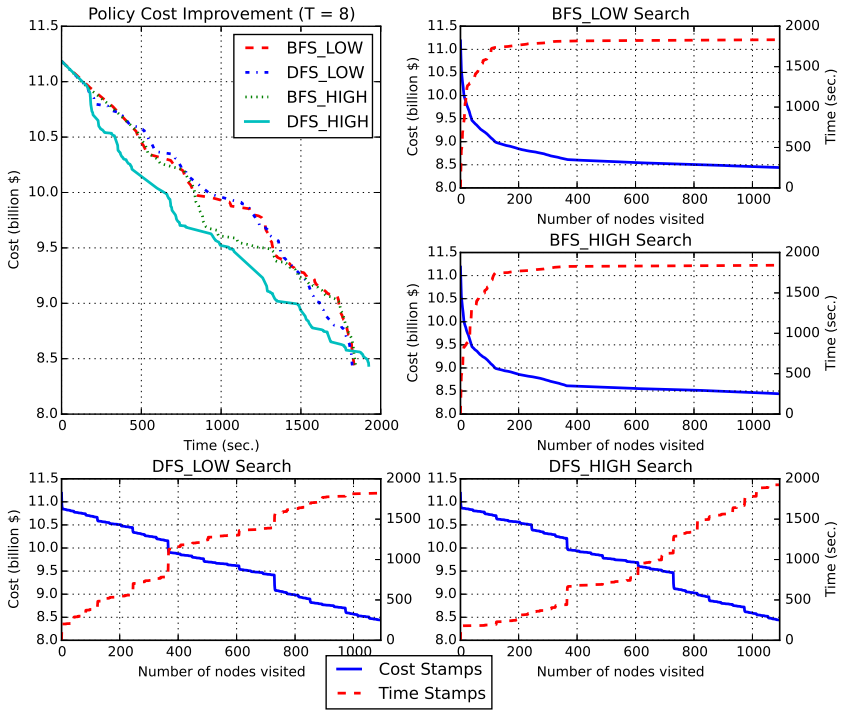


Figure 2.3: Solution cost improvement under four different orderings

these subproblems. However, if computation time is limited, we notice from Figure 2.3 that `DFS_HIGH` produces a better solution than the other three. This indicates that refining the solution on the subtree corresponding to higher demands (as well as higher natural gas prices) improves the solution value faster. Numerical results suggest that as the solution gets refined along this subtree, capacity expansion decisions are postponed. While generation cost is also higher in later periods, the savings of postponing capacity expansion dominates the increment in generation cost.

In the other four figures, we observe that subproblems in earlier periods take more time since they contain more variables and constraints. In addition, subproblems on some nodes take more time or reduce the total expected cost in a larger scale than others, and these two effects usually show up as a pair. These nodes correspond to the ones in early periods of the scenario tree. However, there do exist some nodes whose subproblems take a long time to solve but do not substantially improve the total cost. In other words, the subproblems on

such nodes do not contribute to a significant improvement to the overall solution. It could be interesting if the decision maker can identify such nodes beforehand and skip over them when implementing Algorithm 1.

2.7 Concluding Remarks

We consider a long-term power generation expansion planning problem and propose a new framework of partially adaptive stochastic mixed integer optimization models for the GEP problem and generic capacity expansion planning problem. Our model unifies the two-stage and multistage approaches, and provides the decision maker with the flexibility to adjust the adaptivity of the capacity expansion decisions with respect to uncertainty realizations. Since an optimal solution to the PA model is always feasible to the multistage model, we present nontrivial bounds for the gap between these two models. Furthermore, we propose an approximation algorithm which recursively solves a sequence of PA models and returns a feasible multistage solution. We identify a set of sufficient conditions under which the algorithm produces an optimal multistage solution.

We conduct extensive computational experiments on the PA models and the proposed recursive algorithm on a realistic scale generation expansion planning problem. Numerical results show that PA model provides significant value to the long-term generation expansion problem. It considerably reduces the expected total cost of the GEP problem comparing to the traditional two-stage model. Computation also shows that, with a small amount of flexibility in the expansion decision (i.e. small μ in the $\Pi_{\text{PA}}(\mu)$ model), the PA model can approximate the multistage model fairly well. Computational experiments further shows that it is not necessary to directly solve the PA model, but rather recursively traversing the scenario tree and solving a sequence of small sized PA models can produce a near-optimal solution with a much reduced computation time. We also explore the impact of different search orderings on the performance of the algorithm and its implications.

CHAPTER 3
NESTED DECOMPOSITION OF MULTISTAGE STOCHASTIC INTEGER PROGRAMS WITH BINARY STATE VARIABLES

3.1 Introduction

In general MSIP problems, if integer decisions are present, the nonconvexity of integer programming value functions makes it impossible to directly adapt nested decomposition algorithms such as Benders decomposition or its stochastic variant, stochastic dual dynamic programming (SDDP), to MSIP. In this chapter we develop effective decomposition algorithms for MSIP with *binary* state variables. We focus on binary state variables because any MSIP problem can be approximated as such under mild conditions. By exploiting the binary nature of state variables we develop valid (stochastic) nested decomposition algorithms for MSIP. The key contributions are summarized below.

1. **Extension of nested Benders decomposition algorithm.** We extend the nested Benders decomposition algorithm to solve general MSIP problems with binary state variables. We define a precise notion of valid, tight, and finite cuts, and prove that the algorithm admits finite convergence to an optimal solution if the cuts satisfy these three conditions. This algorithm provides a general framework of solving MSIP problems to optimality and redirects the question to constructing valid and tight cuts for non-convex expected cost-to-go functions at each node.

2. **A new class of Lagrangian cuts.** In considering a reformulation of the node problem and solving its Lagrangian dual problem, we propose a new collection of cutting planes, termed Lagrangian cuts. In such a reformulation, we make local copies of the state variables, and the corresponding constraints are relaxed in the Lagrangian dual. We show that these cuts are valid and tight. A simplified version of a Lagrangian cut

is a strengthened version of the standard Benders' cut. While strengthened Benders' cuts are not necessarily tight, our computational experience indicates that they provide significant benefits.

3. **SND and SDDiP algorithms.** We propose a stochastic variant of the nested decomposition algorithm, namely Stochastic Nested Decomposition (SND) algorithm and its practical realization, Stochastic Dual Dynamic Integer Programming (SDDiP) algorithm, when stochasticity satisfies stage-wise independence, to solve general MSIP problems with binary state variables. We give a rigorous proof of its finite convergence with probability one to an optimal policy as long as valid, tight, and finite cuts are used, and sampling is done with replacement.

4. **Effectiveness of SDDiP algorithm.** Extensive numerical tests are presented to demonstrate the effectiveness of the SDDiP algorithm. In particular, we apply SDDiP with different combinations of cutting planes to three classes of large-scale MSIP problems that have practical importance: a power generation capacity planning problem, a multistage portfolio optimization problem, and an airline revenue management problem. A particularly notable feature is that we transform non-binary state variables in these problems, either integer or continuous, to binary state variables. The promising results demonstrate the applicability of SDDiP for solving MSIP with general (i.e., not necessarily binary) state variables.

3.2 Related Work

In MSLP, the value function $Q_n(\cdot)$ defined in (1.4) and therefore the expected cost-to-go function $\mathcal{Q}_n(\cdot)$ is piecewise linear and convex. This allows for these functions to be under approximated by linear cuts as in nested Benders or L-shaped decomposition (Birge 1985). This algorithm approximates the convex cost-to-go functions by adding Benders' cuts, and converges in finite steps to an optimal solution. When the scenario tree is large,

however, it may be computationally impractical to solve the problem using nested Benders decomposition. Often the underlying stochastic process and the constructed scenario tree is stage-wise independent, i.e., for any two nodes n and n' in $\mathcal{S}_t$ the set of children nodes $\mathcal{C}(n)$ and $\mathcal{C}(n')$ are defined by identical data and conditional probabilities. Then the value functions and expected cost-to-go functions depend only on the stage rather than the nodes, i.e., we have $\mathcal{Q}_n(\cdot) \equiv \mathcal{Q}_t(\cdot)$ for all $n \in \mathcal{S}_t$. This allows for considerable reduction in the number of DP equations (1.4). By exploiting stage-wise independence, a sampling-based nested decomposition method – Stochastic Dual Dynamic Programming (SDDP) is proposed in Pereira and Pinto (1991). This algorithm iterates between forward and backward steps. In the forward step, a subset of scenarios is sampled from the scenario tree and optimal solutions for each sample path are computed for each of them independently. Then in the backward step, starting from the last stage, the algorithm adds supporting hyperplanes to the approximate cost-to-go functions of the previous stage. These hyperplanes are Benders' cuts evaluated at the optimal solutions from the previous stage. After solving the problem at the first stage, a lower bound on the policy value can be obtained. It is then compared against a statistical upper bound computed from the forward step. Various proofs of almost sure convergence of SDDP under mild assumptions have been proposed (see e.g., Chen and Powell 1999; Philpott and Guan 2008; Shapiro 2011; Girardeau et al. 2014). The SDDP algorithm has also been embedded in the scenario tree framework (Rebennack 2013), and extended to risk averse multistage linear programming problems (Shapiro 2011; Shapiro et al. 2013).

While enormous amount of work has been done in both theory and solution strategies for two-stage ($T = 2$) stochastic integer programs, the progress on multistage stochastic integer programming is somewhat limited (cf. Ahmed 2010; Römisch and Schultz 2001). In MSIP, due to the presence of integer variables, the convexity and continuity of the expected cost-to-go functions are lost. A natural way to tackle such a problem is to consider the extensive form of the problem, and then relax the coupling constraints so that it can be

decomposed into scenario-based or component-based subproblems. Different decomposition algorithms involving Lagrangian relaxation (Takriti et al. 1996; Carøe and Schultz 1999; Nowak and Römisch 2000; Chen et al. 2002) and column generation (Lulli and Sen 2004; Sen et al. 2006; Singh et al. 2009) have been successful in solving various classes of MSIP problems. MSIP problems with binary state variables are studied in Alonso-Ayuso et al. (2003), and a branch-and-fix coordination approach is proposed, which coordinates the selection of the branching nodes and branches variables in the scenario subproblems such that they will be jointly optimized. All of the above approaches are based on the extensive form (1.2) of MSIP and do not scale well to large scenario trees.

Existing attempts at extending the nested decomposition and SDDP approaches for the dynamic programming formulation (1.3)-(1.4) for MSIP and other nonconvex problem are based on convex relaxations of the cost-to-go functions. For example, relaxing the integrality constraints so that the problem becomes an MSLP problem (Newham and Wood 2007; Flach et al. 2010; Löhndorf et al. 2013); and combining stochastic dynamic programming and SDDP methods to retain the convexity (Gjelsvik et al. 1999; Helseth et al. 2015). Another way of dealing with nonconvexity is to approximate the cost-to-go functions directly. For instance, approximating the bilinear relationship between variables using McCormick envelope is studied in Cerisola et al. (2012). This approach is further improved by optimizing the Lagrangian multipliers, which results in tighter cuts (Thomé et al. 2013). More recently, the concept of locally valid cuts is introduced and integrated in the SDDP framework (Abgottspon et al. 2014). Note that all the above methods produce solutions to different forms of relaxations rather than the original problem. In Philpott et al. (2016), authors propose a new extension of SDDP, which, rather than cutting planes, uses step functions to approximate the value function.

The remainder of this chapter is organized as follows. In Section 3.3, we describe the class of MSIP problems we consider in this work. In Section 3.4, we summarize the exact nested decomposition algorithm with valid, tight and finite cuts and prove its finite

convergence. Section 3.5 contains the development of Lagrangian cuts as well as the proof of its validity and tightness. We present the SND and SDDiP algorithms and prove their finite convergence with probability one in Section 3.6. Numerical experiments together with discussions are included in Section 3.7. Finally, we provide some concluding remarks in Section 3.8.

3.3 MSIP with Binary State Variables

We consider multistage stochastic mixed integer linear programming problems, i.e., we make the following assumption regarding the MSIP (1.2).

(A1) The objective function $f_n(x_n, y_n)$ in each node n is a linear function in x_n and y_n, and the constraint set X_n is a nonempty compact mixed integer polyhedral set.

The results in this chapter can be easily extended to settings with nonlinear objective functions and constraint sets under mild regularity conditions. However, to make the main idea clear, we focus on the linear case.

A key requirement of our developments is that the state variables x_n in (1.2) are binary. The local variables y_n, however, can be general mixed integer. Recall that, in the presence of integer local variables, the value functions and expected cost-to-go functions are nonconvex with respect to the state variables. Existing nested decomposition algorithms use piecewise convex polyhedral representations of these functions. In general, it is impossible to construct such convex polyhedral representations of the nonconvex value functions that are tight at the evaluated state variable values. On the other hand, any function of binary variables can be represented as a convex polyhedral function. We exploit this fact to develop exact nested decomposition algorithms for MSIP with binary state variables.

Next we show that under a reasonable assumption any MSIP with mixed integer state variables can be approximated to desired precision with an MSIP with binary state variables without increasing the problem size by too much.

Definition 1. We say that an MSIP of the form (1.2) has *complete continuous recourse* if,

for any value of the state variables and the local integer variables, there exist values for the continuous local variables such that the resulting solution is feasible to (1.2). That is, suppose $y_n = (u_n, v_n)$ where $u_n \in \mathbb{Z}_+^{\ell_1}$ and $v_n \in \mathbb{R}_+^{\ell_2}$, then given any $(\hat{x}_{a(n)}, \hat{x}_n, \hat{u}_n)$, there exists $\hat{v}_n \in \mathbb{R}_+^{\ell_2}$ such that $(\hat{x}_{a(n)}, \hat{x}_n, (\hat{u}_n, \hat{v}_n)) \in X_n$ for all $n \in \mathcal{T}$.

In addition to (A1) we also make the following assumption.

(A2) Problem (1.2) has *complete continuous recourse*.

The above assumption can always be achieved by adding nonnegative auxiliary continuous variables and penalizing them in the objective function.

Theorem 6. *For an MSIP with general mixed integer state variables satisfying assumptions (A1) and (A2) we can construct an approximate MSIP that has binary state variables such that any optimal solution to the approximating MSIP is an ε-optimal solution to the original MSIP, and the number, k, of the binary state variables per node in the approximating MSIP satisfies*

$$k \leq d(\lfloor \log_2(M\sqrt{d}/\varepsilon) \rfloor + 1)$$

where d is the number of the state variables per node in the original MSIP and M is a positive constant depending on the problem data.

Proof. Consider an MSIP with $d := d_1 + d_2$ mixed-integer state variables per node:

$$\begin{aligned} \min_{x_n, y_n} \quad & \sum_{n \in \mathcal{T}} p_n f_n(x_n, y_n) \\ \text{s.t.} \quad & (x_{a(n)}, x_n, y_n) \in X_n \quad \forall\, n \in \mathcal{T} \\ & x_n \in \mathbb{Z}_+^{d_1} \times \mathbb{R}_+^{d_2} \qquad \forall\, n \in \mathcal{T}. \end{aligned} \tag{3.1}$$

Since the state variables are bounded by (A1), we can assume that $x_n \in [0, U]^d$ for some positive integer U for all $n \in \mathcal{T}$.

We approximate (3.1) as follows. For an integer state variable $x \in \{0, \ldots, U\}$, we substitute by its binary expansion: $x = \sum_{i=1}^{\kappa} 2^{i-1}\lambda_i$ where $\lambda_i \in \{0, 1\}$ and $\kappa =$

$\lfloor \log_2 U \rfloor + 1$. For a continuous state variable $x \in [0, U]$, we approximate it by binary approximation to a precision of $\epsilon \in (0, 1)$, i.e. $x = \sum_{i=1}^{\kappa} 2^{i-1} \epsilon \lambda_i$ where $\lambda_i \in \{0, 1\}$ and $\kappa = \lfloor \log_2(U/\epsilon) \rfloor + 1$ (see e.g., Glover 1975). Note that $|x - \sum_{i=1}^{\kappa} 2^{i-1} \epsilon \lambda_i| \leq \epsilon$. The total number k of binary variables introduced to approximate the d state variables thus satisfies $k \leq d(\lfloor \log_2(U/\epsilon) \rfloor + 1)$. We then have the following approximating MSIP with binary variables $\lambda_n \in \{0, 1\}^k$

$$\begin{aligned} \min_{\lambda_n, y_n} \quad & \sum_{n \in \mathcal{T}} p_n f_n(A\lambda_n, y_n) \\ \text{s.t.} \quad & (A\lambda_{a(n)}, A\lambda_n, y_n) \in X_n \quad \forall\, n \in \mathcal{T} \\ & \lambda_n \in \{0, 1\}^k \qquad\qquad\quad \forall\, n \in \mathcal{T}, \end{aligned} \tag{3.2}$$

where the $d \times k$ matrix A encodes the coefficients of the binary expansion.

Recall that the local variables are mixed integer, i.e. $y_n = (u_n, v_n)$ with $u_n \in \mathbb{Z}_+^{\ell_1}$ and $v_n \in \mathbb{R}_+^{\ell_2}$. Given $x := \{x_n \in \mathbb{Z}^{d_1} \times \mathbb{R}^{d_2}\}_{n \in \mathcal{T}}$, let

$$\begin{aligned} \phi(x) &:= \min_{u,v} \left\{ \sum_{n \in \mathcal{T}} f_n(x_n, (u_n, v_n)) : (x_{a(n)}, x_n, (u_n, v_n)) \in X_n,\ \forall n \in \mathcal{T} \right\} \\ &= \sum_{n \in \mathcal{T}} \min_{u_n, v_n} \left\{ f_n(x_n, (u_n, v_n)) : (x_{a(n)}, x_n, (u_n, v_n)) \in X_n \right\} \\ &= \sum_{n \in \mathcal{T}} \min_{u_n \in \mathcal{U}_n} \left\{ \psi_n(x_{a(n)}, x_n, u_n) \right\}, \end{aligned}$$

where

$$\psi_n(x_{a(n)}, x_n, u_n) = \min_{v_n \in \mathbb{R}_+^{\ell_2}} \left\{ f_n(x_n, (u_n, v_n)) : (x_{a(n)}, x_n, (u_n, v_n)) \in X_n \right\},$$

and $\mathcal{U}_n$ is the finite set of integer values the local variable u_n can take. By the compactness assumption (A1) and the complete continuous recourse assumption (A2), the function ψ_n is the value function of a linear program that is feasible and bounded for all values of $(x_{a(n)}, x_n, u_n)$. By Hoffman's lemma (Hoffman 1952), there exists a constant $C_n(u_n)$ which

is dependent on the data defining (f_n, X_n) and u_n, such that $\psi_n(x_{a(n)}, x_n, u_n)$ is Lipschitz continuous with respect to $(x_{a(n)}, x_n)$ with this constant. It follows that $\phi(x)$ is Lipschitz continuous with respect to x with constant $C = \sum_{n\in T} \max_{u_n\in U_n} C_n(u_n)$, i.e.,

$$|\phi(x) - \phi(x')| \leq C\|x - x'\| \ \ \forall\, x, x'.$$

Let $(\tilde{\lambda}, \tilde{y})$ be an optimal solution to problem (3.2) and w_2 be its optimal value. Define $\tilde{x}_n = A\tilde{\lambda}_n$ for all $n \in \mathcal{T}$, then $(\tilde{x}, \tilde{y})$ is a feasible solution to (3.1) and has the objective value of w_2. From the definition of ϕ we have that $w_2 = \phi(\tilde{x})$. Now let $(\hat{x}, \hat{y})$ be an optimal solution of (3.1) and w_1 be its optimal value. Note that $w_1 = \phi(\hat{x})$. Let us construct a solution $(\hat{\lambda}, \hat{y}')$ such that

$$\|\hat{x} - A\hat{\lambda}\| \leq \sqrt{|\mathcal{T}|d}\epsilon, \ \text{ and } \ \hat{y}'_n = \text{argmin}_{y_n} \left\{ f(A\hat{\lambda}_{a(n)}, A\hat{\lambda}_n, y_n) : (A\hat{\lambda}_{a(n)}, A\hat{\lambda}_n, y_n) \in X_n \right\}.$$

Then $(\hat{\lambda}, \hat{y}')$ is clearly a feasible solution to (3.2) and has the objective value $\phi(A\hat{\lambda})$. Thus we have the following inequalities

$$\phi(\hat{x}) \leq \phi(\tilde{x}) \leq \phi(A\hat{\lambda}).$$

Thus

$$0 \leq \phi(\tilde{x}) - \phi(\hat{x}) \leq |\phi(A\hat{\lambda}) - \phi(\hat{x})| \leq C\|A\hat{\lambda} - \hat{x}\| \leq C\sqrt{|\mathcal{T}|d}\epsilon = C'\sqrt{d}\epsilon,$$

where $C' = C\sqrt{|\mathcal{T}|}$. By choosing $\epsilon = \varepsilon/C'\sqrt{d}$ and $M = UC'$ we have that $(\tilde{x}, \tilde{y})$ is a ε-optimal solution of (3.1) and $k \leq d(\lfloor \log_2(M\sqrt{d}/\varepsilon) \rfloor + 1)$ as desired. □

Based on the above result, for the remainder of the chapter we consider MSIP with binary state variables. Next, we introduce a simple, but key reformulation of (1.2) based on making local copies of the state variables. That is, we introduce an auxiliary variable z_n for

each node n and equate it to the parent node's state $x_{a(n)}$. The resulting formulation is

$$\begin{aligned}
\min_{x_n, y_n, z_n} \quad & \sum_{n \in \mathcal{T}} p_n f_n(x_n, y_n) \\
\text{s.t.} \quad & (z_n, x_n, y_n) \in X_n \quad \forall\, n \in \mathcal{T} && \text{(3.3a)} \\
& z_n = x_{a(n)} \qquad\quad \forall\, n \in \mathcal{T} && \text{(3.3b)} \\
& z_n \in [0,1]^d \qquad\quad \forall\, n \in \mathcal{T} && \text{(3.3c)} \\
& x_n \in \{0,1\}^d \qquad\quad \forall\, n \in \mathcal{T}. && \text{(3.3d)}
\end{aligned}$$

This reformulation turns out to be crucial for the development of a class of valid and tight inequalities to approximate the cost-to-go functions. Detailed study of (3.3), especially a certain strong duality property, will be given in Section 3.5.3. The important role of the redundant constraint (3.3c) will become clear there. However, except in Section 3.5.3, we will fold constraint (3.3c) into X_n to save space.

Now we can write down the DP equations for the optimal value function of the multistage problem (3.3) at node $n \in \mathcal{T}$ as follows:

$$\begin{aligned}
(P_1): \quad \min_{x_1, y_1, z_1} \quad & f_1(x_1, y_1) + \sum_{m \in \mathcal{C}(1)} q_{1m} Q_m(x_1) && \text{(3.4)} \\
\text{s.t.} \quad & (z_1, x_1, y_1) \in X_1 \\
& z_1 = x_{a(1)} \\
& x_1 \in \{0,1\}^d.
\end{aligned}$$

where for each node $n \in \mathcal{T} \setminus \{1\}$,

$$\begin{aligned}
(P_n): \quad Q_n(x_{a(n)}) := \min_{x_n, y_n, z_n} \quad & f_n(x_n, y_n) + \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(x_n) && \text{(3.5)} \\
\text{s.t.} \quad & (z_n, x_n, y_n) \in X_n
\end{aligned}$$

$$z_n = x_{a(n)}$$
$$x_n \in \{0,1\}^d.$$

3.4 Nested Decomposition

In this section, we present a Nested Decomposition (ND) algorithm for solving the MSIP (3.3) with binary state variables. The proposed ND algorithm solves the DP recursion (3.5) by iteratively strengthening a convex piecewise polyhedral lower approximation of the expected cost-to-go function $\mathcal{Q}_n(\cdot)$ at each node $n \in \mathcal{T}$. The key to the success of such an ND algorithm lies in a certain notion of tightness of the lower approximation of the value functions achieved by valid linear inequalities, which we will precisely define. In the following, we will first outline the ND algorithm, and then introduce the sufficient cut conditions, and prove the finite convergence of the ND algorithm to a global optimal solution of problem (3.3) under these conditions.

3.4.1 The ND Algorithm

The proposed ND algorithm can be outlined as follows. In each iteration i, the ND algorithm consists of a forward step and a backward step. The forward step proceeds stage-wise from $t = 1$ to T by solving a DP equation with an approximate expected cost-to-go function at each node n. In particular, at node n with the parent node's state $x^i_{a(n)}$, the DP recursion (3.5) is approximated by the following forward problem

$$(P^i_n(x^i_{a(n)}, \psi^i_n)): \quad \underline{Q}^i_n(x^i_{a(n)}, \psi^i_n) := \min_{x_n, y_n, z_n} \quad f_n(x_n, y_n) + \psi^i_n(x_n) \tag{3.6a}$$
$$\text{s.t.} \quad (z_n, x_n, y_n) \in X_n \tag{3.6b}$$
$$z_n = x^i_{a(n)} \tag{3.6c}$$
$$x_n \in \{0,1\}^d, \tag{3.6d}$$

where $\psi_n^i(\cdot)$ is defined as:

$$\psi_n^i(x_n) := \min \big\{\theta_n : \quad \theta_n \geq L_n, \tag{3.7a}$$

$$\theta_n \geq \sum_{m \in \mathcal{C}(n)} q_{nm}(v_m^\ell + (\pi_m^\ell)^\top x_n),\ \forall \ell = 1, \ldots, i-1\big\}. \tag{3.7b}$$

In other words, the forward problem in iteration i is characterized by $x_{a(n)}^i$, which is obtained from solving its parent node $a(n)$'s forward problem, as well as by $\psi_n^i(\cdot)$ defined by (3.7a)–(3.7b), which provides a piecewise-linear convex lower-approximation of the expected cost-to-go function $\mathcal{Q}_n(x_n)$. Here, we assume there is a lower bound L_n in (3.7a) to avoid unboundedness of the forward problem. An optimal solution of the state variable in $(P_n^i(x_{a(n)}^i, \psi_n^i))$, denoted as x_n^i, is passed on to the forward problems $(P_m^i(x_n^i, \psi_m^i))$ of its children nodes $m \in \mathcal{C}(n)$. In other words, the forward step updates the state variable solution x_n^i for each $n \in \mathcal{T}$.

When all the forward problems are solved in iteration i, the backward step starts from the last stage T. The goal of the backward step is to update the lower approximation ψ_n^i for each node n. In particular, in a last-stage node $n \in \mathcal{S}_T$, a suitable relaxation of the forward problem $(P_n^i(x_{a(n)}^i, \psi_n^i))$, denoted as (R_n^i), is solved, which produces a linear inequality that lower approximates the true value function $Q_n(x_{a(n)}^i)$. Note that the last stage problem does not have a cost-to-go function, therefore $\psi_n^i \equiv 0$ for all i. Going back one stage, at a node $n \in \mathcal{S}_{T-1}$, all the linear inequalities generated from n's children nodes are aggregated in the form of (3.7b) and added to update its lower approximation from $\psi_n^i(\cdot)$ to $\psi_n^{i+1}(\cdot)$. Then, a suitable relaxation of the updated problem $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ is solved in the backward step at node n. This generates a new linear inequality, which will be aggregated to its parent's node. The backward step continues in this way until it reaches back to the root node of the tree.

Since the linear cuts in (3.7a)-(3.7b) are under-approximations of the true expected cost-to-go function, the optimal value of the forward problem (P_1^i) in node 1 provides a

lower bound to the true optimal value of (3.3). Once all the forward problems in the tree are solved in iteration i, we obtain a feasible solution $\{(x_n^i, y_n^i, z_n^i)\}_{n\in\mathcal{T}}$ to the original multistage problem (3.3), whose total objective value, $\sum_{n\in\mathcal{T}} f_n(x_n^i, y_n^i)$, provides an *upper* bound to the true optimal value of (3.3). If the lower and upper bounds coincide, the ND algorithm terminates; otherwise, another iteration starts. The steps of the ND algorithm are summarized in Algorithm 2. Note that Algorithm 2 is identical to the standard Nested Benders Decomposition for MSLP (Birge and Louveaux 2011), except here we solve suitable relaxations of the stage problems to generate cuts in the backward step.

3.4.2 Sufficient Cut Conditions

The ND algorithm has different implementations according to how the relaxation problem (R_n^i) is formed and how the cut coefficients are obtained in the backward step (Line 14). However, regardless of detailed mechanisms for relaxation and cut generation, the ND algorithm is valid as long as the cuts satisfy the following three sufficient conditions, namely, they are *valid*, *tight*, and *finite*, as defined below.

Definition 2. Let $\{(v_n^i, \pi_n^i)\}_{n\in\mathcal{T}}$ be the cut coefficients obtained from the backward step of the i-th iteration of the ND algorithm (Line 14). We say such a collection of cuts is

(i) *valid*, if for all $n \in \mathcal{T}$ and all iteration i,

$$Q_n(x_{a(n)}) \geq v_n^i + (\pi_n^i)^\top x_{a(n)} \quad \forall\, x_{a(n)} \in \{0,1\}^d, \tag{3.8}$$

(ii) *tight*, if for all $n \in \mathcal{T}$ and all iteration i,

$$\underline{Q}_n^i(x_{a(n)}^i, \psi_n^{i+1}) = v_n^i + (\pi_n^i)^\top x_{a(n)}^i, \tag{3.9}$$

where $\underline{Q}_n^i(x_{a(n)}^i, \psi_n^{i+1})$ is defined in (3.6) and $x_{a(n)}^i$ is the solution of state variable $x_{a(n)}$ obtained from the forward step in iteration i, and

Algorithm 2 :: Nested Decomposition

set $i = 1$, $LB = -\infty$, $UB = \infty$, and an initial lower approximation $\{\psi_n^1(\cdot)\}_{n\in\mathcal{T}}$
while $UB - LB > 0$ **do**
  /* Forward step */
  for $t = 1, \ldots, T$ **do**
    for $n \in \mathcal{S}_t$ **do**
      solve forward problem $P_n^i(x_{a(n)}^i, \psi_n^i)$
      collect solution $(x_n^i, y_n^i, z_n^i, \theta_n^i = \psi_n^i(x_n^i))$
    end for
  end for

  /* Backward step */
  for $t = T - 1, \ldots, 1$ **do**
    for $n \in \mathcal{S}_t$ **do**
      for $m \in \mathcal{C}(n)$ **do**
        solve a suitable relaxation (R_m^i) of the updated problem $P_m^i(x_n^i, \psi_m^{i+1})$ and collect cut coefficients (v_m^i, π_m^i)
      end for
      add cut (3.7b) using the coefficients $\{(v_m^i, \pi_m^i)\}_{m\in\mathcal{C}(n)}$ to ψ_n^i to get ψ_n^{i+1}
    end for
  end for

  /* Lower and upper bounds update */
  $LB = f_1(x_1^i, y_1^i) + \theta_1^i$ and
  $UB = \min\{UB, \sum_{n\in\mathcal{T}} p_n f_n(x_n^i, y_n^i)\}$
  incumbent solution $\{(x_n^*, y_n^*)\}_{n\in\mathcal{T}} \leftarrow \{x_n^i, y_n^i\}_{n\in\mathcal{T}}$
  $i \leftarrow i + 1$
end while
return $\{(x_n^*, y_n^*)\}_{n\in\mathcal{T}}$

(iii) *finite*, if in each iteration i of the ND algorithm, solving the relaxation problem (R_n^i) of $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ can only generate finitely many different cut coefficients (v_n^i, π_n^i).

It is easy to see that valid cuts are needed. The tightness of the cuts means that the cut generated from solving a *relaxation* of $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ needs to exactly recover the objective value $\underline{Q}_n^i(x_{a(n)}^i, \psi_n^{i+1})$ of $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ at $x_{a(n)}^i$. The tightness property alludes to a certain strong duality of the cuts that we introduce in Section 3.5.3, and is crucial in

ensuring the convergence of the ND algorithm. The finiteness condition is important to guarantee finite convergence. In Section 3.5, we discuss various types of relaxations and associated cuts that can be used in the context of the ND algorithm.

3.4.3 Finite Convergence

Next we show that under the sufficient cut conditions identified above, the ND algorithm produces an optimal solution in a finite number of iterations. Before we dive into the details of the proof, we first give a high-level description of the proof. In particular, we prove the convergence of the ND algorithm in two steps. In step 1, we show that the approximation function $\psi_n^i(\cdot)$ obtained from the backward steps of the ND algorithm converges to a certain piecewise linear convex function after a finite number of steps for each node n. Once $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}}$ converges, we prove by induction that the cuts generated in the backward steps are not only tight at the lower estimate of the value functions as in (3.9), but also tight at the *true* value functions, $Q_n^i(x_{a(n)}^i, \psi_n^{i+1})$'s, evaluated at the forward step solutions, which is a stronger tightness property. In step 2, by exploiting the finiteness of the set of stage variable values and using the stronger property of tightness of the generated cuts, we prove that the lower and upper bounds coincide, i.e., the algorithm terminates.

Theorem 7. *If the linear cuts used in the Nested Decomposition algorithm are valid, tight, and finite, then the ND algorithm terminates in a finite number of iterations with an optimal solution to the multistage stochastic program* (3.3)*.*

Proof. We first prove the following claim. (In this chapter, claims are numbered globally for cross reference.)

Claim 1. *For any T-stage problem* (3.3)*, after a finite number (i_T^*) of iterations, the solutions $\{x_n^i\}_{n\in\mathcal{T}}$ generated in the forward steps (in Line 7) and the cuts $\{(v_n^i, \pi_n^i)\}_{n\in\mathcal{T}}$ obtained from the backward steps (in Line 14) of the ND algorithm satisfy the following equality for*

all $n \in \mathcal{T}$ *and* $i \geq i_T^*$,

$$Q_n(x_{a(n)}^i) = v_n^i + (\pi_n^i)^\top x_{a(n)}^i. \tag{3.10}$$

Proof of Claim 1: First notice that x_n is a binary vector thus can take at most 2^d different values for all $n \in \mathcal{T}$, where d is the dimension of x_n. For any $n \in \mathcal{S}_T$ and $i \geq 1$, $\psi_n^i(\cdot) \equiv 0$. For $n \in \mathcal{S}_{T-1}$, since x_n only has finitely many different values, and the cuts used in the ND algorithm satisfy the finite property, it follows that the total number of distinct cuts that may be added to node n in the backward steps (Line 14) of the ND algorithm is finite, hence there are only finitely many different polyhedral models for $\psi_n^i(\cdot)$ for any $i \geq 1$. Similarly, using each of them as the approximate cost-to-go function can only generate finitely many distinct cuts for n's parent node. Continuing this way to node 1, we know there are only finitely many cuts that can be added to any $n \in \mathcal{T}$. Since the number of cuts in $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}}$ is a monotone nondecreasing sequence with respect to iteration i, there exists $i_T^* < \infty$ such that $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}} \equiv \{\psi_n^*(\cdot)\}_{n\in\mathcal{T}}$ for all $i \geq i_T^*$.

Next, we want to show that, after $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}}$ converges to $\{\psi_n^*(\cdot)\}_{n\in\mathcal{T}}$, the lower estimate of the value function, $\underline{Q}_n^i(x_{a(n)}^i, \psi_n^*)$ in (3.6), will support the true value function $Q_n(x_{a(n)}^i)$ given in (3.5) evaluated at the forward solution $x_{a(n)}^i$ at all $n \in \mathcal{T}$ after $i \geq i_T^*$. Therefore, the cuts $\{(v_n^i, \pi_n^i)\}_{n\in\mathcal{T}}$ generated from the backward steps satisfy a stronger notion of tightness, namely, they are not only tight at the lower estimate of the value function as described in (3.9), but also tight at the true value function as shown in (3.10).

We prove (3.10) by induction over the stages for any T-stage problem. In particular, at any node n in the last stage T, i.e., $n \in \mathcal{S}_T$, the cost-to-go function $\psi_n^i \equiv 0$. Therefore, $\underline{Q}_n^i(x_{a(n)}^i, \psi_n^i) = Q_n^i(x_{a(n)}^i)$ for all $i \geq 1$ and all binary $x_{a(n)}^i$. Then, (3.10) follows from the tightness property of the cuts (3.9).

For the induction step, consider a node $n \in \mathcal{S}_t$ with $t \leq T-1$, and assume that the cuts $\{(v_m^i, \pi_m^i)\}_{m\in\mathcal{C}(n)}$ generated at its children nodes $m \in \mathcal{C}(n)$ satisfy (3.10), i.e.,

$Q_m(x_n^i) = v_m^i + (\pi_m^i)^\top x_n^i$. We want to show the cut (v_n^i, π_n^i) generated at node n by solving a suitable relaxation (R_n^i) of the updated forward problem $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ also satisfies (3.10). We have the following relations:

$$
\begin{aligned}
Q_n(x_{a(n)}^i) &\geq v_n^i + (\pi_n^i)^\top x_{a(n)}^i && \text{(3.11a)}\\
&= \underline{Q}_n^i(x_{a(n)}^i, \psi_n^{i+1}) && \text{(3.11b)}\\
&= \underline{Q}_n^i(x_{a(n)}^i, \psi_n^*) && \text{(3.11c)}\\
&= f_n(x_n^i, y_n^i) + \psi_n^*(x_n^i) && \text{(3.11d)}\\
&\geq f_n(x_n^i, y_n^i) + \sum_{m \in \mathcal{C}(n)} q_{nm}(v_m^i + (\pi_m^i)^\top x_n^i) && \text{(3.11e)}\\
&= f_n(x_n^i, y_n^i) + \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(x_n^i) && \text{(3.11f)}\\
&\geq Q_n(x_{a(n)}^i). && \text{(3.11g)}
\end{aligned}
$$

The inequality in (3.11a) follows from the validity of the cut (v_n^i, π_n^i) from (3.8). The equality in (3.11b) follows from the fact that (v_n^i, π_n^i) is a tight cut for the relaxation problem of $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ and uses the definition of tight cuts given in (3.9). Notice that after i_T^* iterations, the lower approximation functions ψ_n^i converged to ψ_n^*. Therefore, the problem $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$ in the backward step is identical to the forward problem $(P_n^i(x_{a(n)}^i, \psi_n^i))$ in iteration i, which in turn is the same as $(P_n^i(x_{a(n)}^i, \psi_n^*))$. Thus, we have the equality in (3.11c). Since (x_n^i, y_n^i) is an optimal solution of the forward problem $(P_n^i(x_{a(n)}^i, \psi_n^*))$, we also have the equality in (3.11d). Since the cuts $\{(v_m^i, \pi_m^i)\}_{m \in \mathcal{C}(n)}$ are already contained in the description of ψ_n^*, the inequality in (3.11e) follows from the construction of ψ_n^* in (3.7b). The equality in (3.11f) holds due to the induction hypothesis. Lastly, (3.11g) follows, because (x_n^i, y_n^i) is a feasible solution of the problem (P_n) with the parent state $x_{a(n)}^i$ as defined in (3.5). This closes the induction step and proves Claim 1. $\diamond$

Since there are finitely many feasible state vectors $\{x_n\}_{n \in \mathcal{T}}$, in a finite number of iterations after i_T^*, the ND algorithm will repeat a solution $\{x_n^i\}_{n \in \mathcal{T}}$ with $i \geq i_T^*$ that has

been obtained in a previous iteration $j \geq i_T^*$. (Note that there could be multiple optimal solutions, so this may not happen in the $(i_T^* + 1)$-th iteration.) Consider $x_n^j = x_n^i = \hat{x}_n$ for $j < i$ and all $n \in \mathcal{T}$. Note that the upper bound at the end of iteration j is

$$UB^j \leq \sum_{n \in \mathcal{T}} p_n f_n(\hat{x}_n, \hat{y}_n), \tag{3.12}$$

where

$$\hat{y}_n \in \operatorname{argmin}_{y_n} \{f_n(\hat{x}_n, y_n) : (\hat{x}_{a(n)}, \hat{x}_n, y_n) \in X_n\}. \tag{3.13}$$

The cuts generated in the backward step of iteration j are of the form (c.f. (3.7b))

$$\theta_n \geq \sum_{m \in \mathcal{C}(n)} q_{nm}(\hat{v}_m^\ell + (\hat{\pi}_m^\ell)^\top x_n),$$

which are tight at $\hat{x}_n$ by (3.10), i.e., $Q_m(\hat{x}_n) = \hat{v}_m^\ell + (\hat{\pi}_m^\ell)^\top \hat{x}_n$. Since a cut is never discarded and $\hat{x}_n$ is an optimal solution of the forward problem $P_n^i(x_{a(n)}^i, \psi_n^*)$ for all n in iteration i, we must have $\theta_n^i \geq \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(\hat{x}_n)$ for all $n \in \mathcal{T}$. In particular, the lower bound at iteration i is

$$LB^i = f(\hat{x}_1, \hat{y}_1) + \theta_1^i \geq f(\hat{x}_1, \hat{y}_1) + \sum_{m \in \mathcal{C}(1)} q_{1m} Q_m(\hat{x}_1). \tag{3.14}$$

Now note that with fixed state vector $\{\hat{x}_n\}_{n \in T}$, the multistage stochastic problem (3.3) separates by nodes, thus we have

$$Q_n(\hat{x}_{a(n)}) = f_n(\hat{x}_n, \hat{y}_n) + \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(\hat{x}_n) \quad \forall\, n \in \mathcal{T}, \tag{3.15}$$

where $\hat{y}_n$ satisfies (3.13). Combining (3.15) with (3.14), we have

$$LB^i \geq \sum_{n \in \mathcal{T}} p_n f_n(\hat{x}_n, \hat{y}_n) \geq UB^j \geq UB^i,$$

where the second inequality follows from (3.12) and the third inequality follows from the fact that the upper bounds are nonincreasing. Thus the algorithm will terminate at the end of iteration i. Upon termination, due to the complete continuous recourse assumption, the solution vector $\{(x_n^i, y_n^i, z_n^i)\}_{n\in\mathcal{T}}$ is feasible and has an objective value equal to a valid lower bound, and hence is optimal to the multistage stochastic program (3.3). This completes the proof of the theorem. □

In the above description of the algorithm, we only use one direction to traverse the scenario tree – proceed forward in time all the way, and then update all the node subproblems by cuts and start again from stage 1. Alternatively one can consider various other tree traversal schemes, e.g., alternating between generating solutions for next stage and then passing cuts back for some steps before proceeding forward (Gassmann 1990).

3.5 Cut families

In this section, we discuss various types of cuts that can be used within the ND algorithm. We discuss the well known Benders' and integer optimality cuts, and introduce the *Lagrangian* cuts derived from a Lagrangian relaxation corresponding to the reformulation (3.3), where local copies of state variables are introduced, and an associated collection of *strengthened Benders'* cuts.

3.5.1 Benders' Cut

A well known family of cuts is the Benders' cut (Benders 1962), where the relaxation (R_n^i) solved in the backward step is the LP relaxation of problem $(P_n^i(x_{a(n)}^i, \psi_n^{i+1}))$. Therefore, the cost coefficients (v_n^i, π_n^i) are computed based on the optimal value of the LP relaxation and a basic optimal dual solution. Specifically, the cut added to node n in the backward step

evaluated at a forward solution x_n^i takes the following form

$$\theta_n \geq \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m^{LP}(x_n^i) + \sum_{m \in \mathcal{C}n} q_{nm} (\pi_m^i)^\top (x_n - x_n^i), \tag{3.16}$$

where $Q_m^{LP}(x_n^i)$ is the optimal LP relaxation objective function value of problem $(P_m^i(x_n^i, \psi_m^{i+1}))$ and π_m^i is a basic optimal dual solution corresponding to constraints $z_m = x_n^i$. This is the cut family used in nested decomposition algorithms for MSLP. For MSIP, Benders' cut are valid and finite (when basic dual optimal solutions are used) but not tight in the sense of (3.9) in general. Accordingly, for MSIP, the ND algorithm is not guaranteed to produce an optimal solution using only Benders' cuts.

3.5.2 Integer Optimality Cut

Another interesting collection of cutting planes is introduced by Laporte and Louveaux 1993 and is designed for solving two-stage stochastic programs with binary first-stage variables. It is generated by evaluating the subproblem at a feasible first-stage solution and coincides with the true expected cost-to-go function at the proposed first-stage solution. We present a natural extension of them to the ND algorithm for the multistage setting.

Let x_n^i be a solution to the problem $(P_n^i(x_{a(n)}^i \psi_n^i))$ solved in iteration i at node n in the forward step. The relaxations solved in the backward step are the original problems themselves. That is, let v_m^{i+1} be the optimal objective value of problem $(R_m^i) = (P_m^i(x_n^i, \psi_m^{i+1}))$ given x_n^i for all $m \in \mathcal{C}(n)$. Then the integer optimality cut added to $(P_n^i(x_{a(n)}^i, \psi_n^i))$ in the backward step takes the following form

$$\theta_n \geq (\bar{v}_n^{i+1} - L_n) \left(\sum_j (x_{n,j}^i - 1) x_{n,j} + \sum_j (x_{n,j} - 1) x_{n,j}^i \right) + \bar{v}_n^{i+1}, \tag{3.17}$$

where $\bar{v}_n^{i+1} = \sum_{m \in \mathcal{C}(n)} q_{nm} v_m^{i+1}$. It is easy to verify that integer optimality cuts are valid, tight and finite. Thus the ND algorithm with this cut family is an exact approach for

solving MSIP with binary state variables. However, these cuts are only tight at the proposed binary solution x_n^i and could be very loose at other solutions, and hence may not perform satisfactorily.

3.5.3 Lagrangian Cut

We consider another class of cuts obtained by solving a Lagrangian dual of the node forward problems. The relaxation solved in the backward step of iteration i in node n in this case is:

$$(R_n^i): \quad \max_{\pi_n} \left\{ \mathcal{L}_n^i(\pi_n) + \pi_n^\top x_{a(n)}^i \right\} \tag{3.18}$$

where

$$\begin{aligned} \mathcal{L}_n^i(\pi_n) \;=\; \min_{x_n, y_n, z_n, \theta_n} \quad & f_n(x_n, y_n) + \theta_n - \pi_n^\top z_n \\ \text{s.t.} \quad & (z_n, x_n, y_n) \in X_n \\ & x_n \in \{0,1\}^d \\ & z_n \in [0,1]^d \\ & \theta_n \geq L_n \\ & \theta_n \geq \sum_{m \in \mathcal{C}(n)} q_{nm}(v_m^\ell + (\pi_m^\ell)^\top x_n) \quad \forall \ell = 1, \ldots, i. \end{aligned} \tag{3.19}$$

We will denote the feasible region defined by the first four constraint systems of $\mathcal{L}_n^i(\pi_n)$ as X_n' and that defined by all five constraint systems as X_n''.

Given any $\{x_n^i\}_{n \in \mathcal{T}}$ with $x_n^i \in \{0,1\}^d$, a collection of cuts given by the coefficients $\{(v_n^i, \pi_n^i)\}_{n \in \mathcal{T}}$ is generated in the backward step of iteration i, where π_n^i is an optimal solution to the Lagrangian dual problem (R_n^i) and $v_n^i = \mathcal{L}_n^i(\pi_n^i)$ for all $n \in \mathcal{T}$. We call this collection of cuts the *Lagrangian cuts*.

Theorem 8. *Given any $\{x_n^i\}_{n \in \mathcal{T}}$ with $x_n^i \in \{0,1\}^d$, let π_n^i be an optimal solution to the Lagrangian dual problem (R_n^i) in* (3.18) *and $v_n^i = \mathcal{L}_n^i(\pi_n^i)$. Then, the collection of Lagrangian cuts $\{(v_n^i, \pi_n^i)\}_{n \in \mathcal{T}}$ is valid and tight in the sense of* (3.8)-(3.9).

Proof. First, we prove that the Lagrangian cuts generated in iteration i of the ND algorithm are tight at the forward solution $\{x_n^i\}_{n\in\mathcal{T}}$. The tightness of the Lagrangian cuts is essentially implied by a strong duality between the Lagrangian relaxation defined by (3.18)-(3.19) and the forward problem $(P_n^i(x_{a(n)}^i, \psi_n^i))$ defined in (3.6). Then, we prove by induction that they are also valid cuts.

Take any node $n \in \mathcal{T}$. Let π_n^i be an optimal dual solution of (3.18). Then, we have the following equalities:

$$\begin{aligned}\mathcal{L}_n^i(\pi_n^i) + (\pi_n^i)^\top x_{a(n)}^i &= \min\left\{f_n(x_n, y_n) + \theta_n - (\pi_n^i)^\top(z_n - x_{a(n)}^i) : (z_n, x_n, y_n, \theta_n) \in X_n''\right\} \\ &= \min\left\{f_n(x_n, y_n) + \theta_n : (z_n, x_n, y_n, \theta_n) \in \operatorname{conv}(X_n''), z_n = x_{a(n)}^i\right\}, \end{aligned} \tag{3.20}$$

where (3.20) follows from Theorem 6.2 in Nemhauser and Wolsey 1999. Let $(\hat{z}_n, \hat{x}_n, \hat{y}_n, \hat{\theta}_n) \in \operatorname{conv}(X_n'')$ be an optimal solution of (3.20). Then there exists $\{(\hat{z}_n^k, \hat{x}_n^k, \hat{y}_n^k, \hat{\theta}_n^k)\}_{k\in K} \in X_n''$ such that $(\hat{z}_n, \hat{x}_n, \hat{y}_n, \hat{\theta}_n) = \sum_{k\in K} \lambda_k \cdot (\hat{z}_n^k, \hat{x}_n^k, \hat{y}_n^k, \hat{\theta}_n^k)$, where K is a finite set, $\lambda_k \geq 0$ for all $k \in K$, and $\sum_{k\in K} \lambda_k = 1$. Since $x_{a(n)}^i \in \{0,1\}^d$ and $\hat{z}_n^k \in [0,1]^d$ for all k, we have that $\sum_{k\in K} \lambda_k \hat{z}_n^k = \hat{z}_n = x_{a(n)}^i$, which implies that $\hat{z}_n^k = x_{a(n)}^i$ for all k. Thus $(\hat{z}_n, \hat{x}_n, \hat{y}_n, \hat{\theta}_n) \in \operatorname{conv}(X_n'' \wedge \{z_n = x_{a(n)}^i\})$ and

$$\begin{aligned}\mathcal{L}_n^i(\pi_n^i) + (\pi_n^i)^\top x_{a(n)}^i &= \min\left\{f_n(x_n, y_n) + \theta_n : (z_n, x_n, y_n, \theta_n) \in \operatorname{conv}(X_n'' \wedge \{z_n = x_{a(n)}^i\})\right\} \\ &= \min\left\{f_n(x_n, y_n) + \theta_n : (z_n, x_n, y_n, \theta_n) \in X_n'', z_n = x_{a(n)}^i\right\} \\ &= f_n(x_n^i, y_n^i) + \theta_n^i = \underline{Q}_n^i(x_{a(n)}^i, \psi_n^i), \end{aligned}$$

where the second equality follows since $f_n(x_n, y_n)$ is linear. This proves the tightness of the Lagrangian cuts according to (3.9).

Next, we show by induction that the Lagrangian cuts are valid. For the base case, we consider any node $n \in \mathcal{S}_T$. Note that $\psi_n^i \equiv 0$ in the last stage problem. Relaxing the constraint $z_n = x_{a(n)}$ in the definition (3.5) of $Q_n(x_{a(n)})$ using the optimal multiplier π_n^i of

(3.18), we have for any $x_{a(n)} \in \{0,1\}^d$,

$$\begin{aligned} Q_n(x_{a(n)}) &\geq \min\{f_n(x_n, y_n) - (\pi_n^i)^\top (z_n - x_{a(n)}) : (z_n, x_n, y_n) \in X_n'\} \\ &= \min\{f_n(x_n, y_n) - (\pi_n^i)^\top z_n : (z_n, x_n, y_n) \in X_n'\} + (\pi_n^i)^\top x_{a(n)} \\ &= \mathcal{L}_n^i(\pi_n^i) + (\pi_n^i)^\top x_{a(n)}. \end{aligned}$$

Thus the Lagrangian cut is valid at any $n \in \mathcal{S}_T$. For the induction step, consider a node $n \in \mathcal{S}_t$ with $t \leq T-1$, and assume that the Lagrangian cuts defined by $\{(v_m^i, \pi_m^i)\}_{m \in \mathcal{C}(n)}$ are valid. Note that

$$Q_n(x_{a(n)}) = \min\Big\{ f_n(x_n, y_n) + \theta_n : (z_n, x_n, y_n) \in X_n,\ z_n = x_{a(n)},\ \theta_n \geq \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(x_n) \Big\}. \tag{3.21}$$

Since the cuts defined by $\{(\pi_m^i, v_m^i)\}_{m \in \mathcal{C}(n)}$ are valid, i.e. $Q_m(x_n) \geq v_m^i + (\pi_m^i)^\top x_n$ for any $x_n \in \{0,1\}^d$, X_n'' with these cuts is a relaxation of the feasible region of (3.21). Therefore, we have

$$\begin{aligned} Q_n(x_{a(n)}) &\geq \min\left\{ f_n(x_n, y_n) + \theta_n : (z_n, x_n, y_n, \theta_n) \in X_n'', z_n = x_{a(n)} \right\} \\ &\geq \min\left\{ f_n(x_n, y_n) + \theta_n - (\pi_n^i)^\top z_n : (z_n, x_n, y_n, \theta_n) \in X_n'' \right\} + (\pi_n^i)^\top x_{a(n)} \\ &= \mathcal{L}_n^i(\pi_n^i) + (\pi_n^i)^\top x_{a(n)}, \end{aligned}$$

where the second inequality is by relaxing the constraint $z_n = x_{a(n)}$. Thus the Lagrangian cut defined by (π_n^i, v_n^i) is valid. This completes the proof of the theorem. □

If we restrict the set of dual optimal solutions π_n^i of (R_n^i) to be basic, then the set of Lagrangian cuts is also finite. Accordingly, the ND algorithm with this cut family is guaranteed to produced an optimal solution to MSIP with binary state variables in a finite number of iterations.

3.5.4 Strengthened Benders' Cut

The Lagrangian problem is an unconstrained optimization problem, thus for any fixed π_n, solving (3.19) to optimality yields a valid cut. Therefore, one can strengthen Benders' cut by solving a node mixed integer program. More concretely, we solve (3.19) at all $m \in \mathcal{C}(n)$ with π_m equal to a basic optimal LP dual solution π_m^i corresponding to the constraints $z_m = x_n^i$. Upon solving all these node subproblems, we can construct a valid cut which is parallel to the regular Benders' cut,

$$\theta_n \geq \sum_{m \in \mathcal{C}(n)} q_{nm} \mathcal{L}_m(\pi_m^i) + \sum_{m \in \mathcal{C}n} q_{nm} (\pi_m^i)^\top x_n. \tag{3.22}$$

Indeed, we have $\mathcal{L}_m(\pi_m^i) \geq Q_m^{LP}(x_n^i) - (\pi_m^i)^\top x_n^i$, thus (3.22) is at least as tight as Benders' cuts (3.16). For this reason, we call these cuts *strengthened Benders' cuts*. The strengthened Benders' cuts are valid and finite but are not guaranteed to be tight according to (3.9). Nonetheless these cuts afford significant computational benefits as demonstrated in Section 3.7.

Even though Lagrangian cuts are tight, whereas strengthened Benders' cuts are not in general, the latter are not necessarily dominated by the previous one, as shown in the following example.

Example 2. Consider the following two-stage program with only 1 scenario,

$$\min_x \ \{x_1 + x_2 + Q(x_1, x_2) : x_1, x_2 \in \{0, 1\}\}$$

where $Q(x_1, x_2) = \min\{4y : y \geq 2.6 - 0.25x_1 - 0.5x_2, y \leq 4, y \in \mathbb{Z}_+\}$. It is easy to compute that $Q(0, 0) = 12$. The Benders' cut described in (3.16) is $\theta \geq 10 - x_1 - 2x_2$; the strengthened Benders' cut described in (3.22) is $\theta \geq 11 - x_1 - 2x_2$; and the Lagrangian cut is $\theta \geq 12 - 4x_2$. We see that the Lagrangian cut supports function $Q(x_1, x_2)$ at $(0, 0)$, while the other two do not. Also, it is clear that the strengthened Benders' cut strictly improves

the Benders' cut, and the strengthened Benders' cut and the Lagrangian cut do not dominate each other. $\diamond$

3.6 Stochastic Nested Decomposition

The number of nodes in a scenario tree, in most cases, can be enormous. Therefore, traversing the tree back and forth in every iteration of the ND algorithm can be computationally expensive. In this section, we present a Stochastic Nested Decomposition (SND) algorithm and its special case, Stochastic Dual Dynamic Integer Programming, or SDDiP, when the stochasticity satisfies stage-wise independence, for solving the MSIP (3.1) with binary state variables.

3.6.1 The SND Algorithm

In contrast to the ND algorithm, the SND algorithm does not solve all the forward problems in each iteration. Instead, a subset of scenarios, i.e., a set of paths from root to a subset of leaf nodes, is sampled from the tree in each forward step. In particular, we consider the following sampling procedure: in each iteration of the SND algorithm, M nodes, denoted as $\{n_{j_1}, \ldots, n_{j_M}\}$, out of all the N nodes in the last stage of the scenario tree are sampled based on the distribution $\{p_n : n \in \mathcal{S}_T\}$. Let $\mathcal{P}(n_{j_k})$ denote the scenario path from root to the leaf node n_{j_k}. The set $\{\omega_k := \mathcal{P}(n_{j_k})\}_k$ contains all the corresponding scenario paths for all $k = 1, \ldots, M$. The sampling can be done with or without replacement, and there is no significant practical difference between them as M is usually much smaller than N. As in ND, each iteration of the SND algorithm consists of a forward step and a backward step. In the forward step, we solve forward problems defined in (3.6) along each sampled scenario path and collect forward solutions. We call them *candidate solutions*. In the backward step, we only add cuts to the subproblems at the nodes which were traversed in the previous forward step and keep all subproblems at other nodes the same as in the previous iteration. The full algorithm is described in Algorithm 3.

Algorithm 3 :: Stochastic Nested Decomposition

Initialize: $LB \leftarrow -\infty$, $UB \leftarrow +\infty$, $i \leftarrow 1$, and an initial lower approximation $\{\psi_n^1(\cdot)\}_{n\in\mathcal{T}}$
while some stopping criterion is not satisfied **do**
  Sample M scenarios $\Omega^i = \{\omega_1^i, \ldots, \omega_M^i\}$

  /* Forward step */
  for $k = 1, \ldots, M$ **do**
    for $n \in \omega_k^i$ **do**
      solve forward problem $P_n^i(x_{a(n)}^i, \psi_n^i)$
      collect solution $(x_n^i, y_n^i, z_n^i, \theta_n^i = \psi_n^i(x_n^i))$
    end for
    $u^k \leftarrow \sum_{n\in\omega_k^i} f_n(x_n^i, y_n^i)$
  end for

  /* (Statistical) upper bound update */
  $\hat{\mu} \leftarrow \frac{1}{M}\sum_{k=1}^M u^k$ and $\hat{\sigma}^2 \leftarrow \frac{1}{M-1}\sum_{k=1}^M (u^k - \hat{\mu})^2$
  $UB \leftarrow \hat{\mu} + z_{\alpha/2}\frac{\hat{\sigma}}{\sqrt{M}}$

  /* Backward step */
  for $t = T-1, \ldots, 1$ **do**
    for $n \in \mathcal{S}_t$ **do**
      if $n \in \omega_k^i$ for some k **then**
        for $m \in \mathcal{C}(n)$ **do**
          solve a suitable relaxation (R_n^i) of the updated problem $P_n^i(x_{a(n)}^i, \psi_n^{i+1})$ and collect cut coefficients (v_m^i, π_m^i)
        end for
        add cut (3.7b) using the coefficients $\{(v_m^i, \pi_m^i)\}_{m\in\mathcal{C}(n)}$ to ψ_n^i to get ψ_n^{i+1}
      else
        $\psi_n^{i+1} \leftarrow \psi_n^i$
      end if
    end for
  end for

  /* Lower bound update */
  solve $P_1^i(\bar{x}_0, \psi_1^{i+1})$ and set LB be the optimal value
  $i \leftarrow i + 1$
end while

The SND algorithm (Algorithm 3) does not specify a termination condition. One possibility is to stop when the upper bound UB and lower bound LB are sufficiently

close. It is important to note that the upper bound is a statistical upper bound. Its validity is guaranteed with certain probability provided that M is not too small (e.g., $M > 30$). However, no matter how large M is, it could still happen that this upper bound is smaller than the valid lower bound evaluated in the backward step. As a result, one needs to be careful when using the stopping criterion $UB - LB \leq \epsilon$. A conservative test is to compare the lower bound with the estimated upper bound plus two standard deviations. Other stopping criteria are also used in the literature, e.g., stop the algorithm when the lower bounds become stable and the statistical upper bound given by a large sample size is close to the lower bound; or enforce a limit on the total number of iterations (Shapiro et al. 2013; Bruno et al. 2016).

3.6.2 Convergence

In this section, we prove the convergence of the SND algorithm. In particular, we show that, with probability one, the approximate cost-to-go functions constructed using valid, tight, and finite cuts define an optimal solution to MSIP with binary state variables in a finite number of iterations. We have the following technical assumption.

(A3) In any node $n \in \mathcal{T}$ and iteration i in the SND algorithm, given the same parent solution $x^i_{a(n)}$ and the same approximate cost-to-go function ψ^i_n, the node problem $P^i_n(x^i_{a(n)}, \psi^i_n)$ is always solved to the same optimal solution x^i_n.

This assumption is to avoid the situation, where the algorithm for solving the same node problem keeps generating different optimal solutions (if they exist). Most deterministic MIP solvers, e.g. CPLEX and Gurobi, satisfy (A3). Therefore, it is a practical assumption. However, notice that we do not assume the node problem $P^i_n(\cdot)$ has a unique optimal solution.

Theorem 9. *Suppose the sampling procedure in the forward step is done with replacement, the cuts generated in the backward step are valid, tight, and finite, and the algorithm for solving the node problems $\{P^i_n(\cdot)\}_{n\in\mathcal{T}}$ satisfies (A3), then with probability one, the forward step of the SND algorithm defines an optimal solution to the multistage stochastic program*

(3.3) *after a finite number of iterations.*

Proof. First, notice that each binary state variable x_n in (3.3) can only take at most 2^d different values and the cutting planes used in the backward steps are finite (see Definition 2), it follows that there are finitely many possible realizations (polyhedral models) for the approximate expected cost-to-go functions $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}}$ for all $i \geq 1$.

At the beginning of any iteration $i \geq 1$, the current approximate expected cost-to-go functions $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}}$ define a solution (x_n^i, y_n^i) over the tree obtained by the forward step of iteration i, i.e.,

$$(x_n^i, y_n^i) \in \text{argmin} \left\{ \begin{array}{ll} \min\limits_{x_n, y_n} & f_n(x_n, y_n) + \psi_n^i(x_n) \\ \text{s.t.} & (x_{a(n)}^i, x_n, y_n) \in X_n \quad \forall n \in \mathcal{T} \end{array} \right\}. \tag{3.23}$$

It is worth noting that during a particular iteration, the SND algorithm does not compute all of these solutions but only those along the sampled paths (scenarios). We first prove the following claim, which gives a sufficient condition under which the solution defined in (3.23) is optimal to the original problem.

Claim 2. *If, at iteration i of the SND algorithm, $\psi_n^i(x_n^i) = \mathcal{Q}_n(x_n^i)$ for all $n \in \mathcal{T}$, then the forward solution $\{x_n^i, y_n^i\}_{n\in\mathcal{T}}$ is optimal to problem* (3.3)*.*

Proof of Claim 2: Since the cuts generated in backward steps are valid, $\{\psi_n^i(\cdot)\}_{n\in\mathcal{T}}$ is a lower approximation to the true expected cost-to-go functions, i.e., $\psi_n^i(x_n) \leq \mathcal{Q}_n(x_n)$ for all $x_n \in \{0,1\}^d$ and $n \in \mathcal{T}$. Therefore, $\underline{Q}_n^i(x_{a(n)}^i, \psi_n^i) \leq Q_n(x_{a(n)}^i)$ (cf. (3.5) and (3.6)). Furthermore, we have

$$\underline{Q}_n^i(x_{a(n)}^i, \psi_n^i) = f_n(x_n^i, y_n^i) + \psi_n^i(x_n^i) \tag{3.24a}$$

$$= f_n(x_n^i, y_n^i) + \mathcal{Q}_n(x_n^i) \tag{3.24b}$$

$$\geq Q_n(x_{a(n)}^i), \tag{3.24c}$$

where (3.24a) is true because x_n^i by definition is an optimal solution of $(P_n^i(x_{a(n)}^i, \psi_n^i))$, (3.24b) follows the assumption $\psi_n^i(x_n^i) = \mathcal{Q}_n(x_n^i)$, and (3.24c) holds because (x_n^i, y_n^i) is feasible for the true DP recursion (3.5). Therefore, (x_n^i, y_n^i) is also optimal for the true DP recursion (3.5) for all $n \in \mathcal{T}$, thus (x_n^i, y_n^i) is optimal for (3.3). This completes the proof of Claim 2. $\diamond$

Suppose the solution defined by (3.23) at the beginning of iteration i is not optimal, then there must exist some $n \in \mathcal{T}$ such that $\psi_n^i(x_n^i) < \mathcal{Q}_n(x_n^i)$. Any iteration $j \geq i$ can be characterized as either one of the following two types:

(a) $\{\psi_n^{j+1}(\cdot)\}_{n\in\mathcal{T}} \neq \{\psi_n^j(\cdot)\}_{n\in\mathcal{T}}$, i.e., at least one $\psi_n^j(\cdot)$ changes during the backward step;

(b) $\{\psi_n^{j+1}(\cdot)\}_{n\in\mathcal{T}} = \{\psi_n^j(\cdot)\}_{n\in\mathcal{T}}$, i.e., all $\psi_n^j(\cdot)$ remain the same after the backward step.

It is possible that consecutive iterations after i may belong to Type-a or Type-b iterations. Let us denote I_a^k and I_b^k as the k-th such set of consecutive Type-a and Type-b iterations, respectively. Let $K = \sup\{i : \{x_n^i, y_n^i\}_{n\in\mathcal{T}} \text{ is not optimal}\}$, and let K_a and K_b respectively be the total number of sets of consecutive Type-a and Type-b iterations, when the forward tree solution $\{x_n^i, y_n^i\}_{n\in\mathcal{T}}$ is not optimal. Let us also denote $|I_a^k|$ and $|I_b^k|$ as the cardinality of the k-th set of consecutive Type-a and Type-b iterations, respectively. Since there are only finitely many cuts that can be added, both K_a and each $|I_a^k|$ must be finite. As will be shown below, each I_b^k occurrence before the SND algorithm converges is followed by a Type-a iteration. Therefore, $K_b \leq K_a$, hence K_b is also finite. We next show that each $|I_b^k|$ is finite with probability 1.

Claim 3. *With probability 1, $|I_b^k|$ is finite for all $1 \leq k \leq K_b$.*

Proof of Claim 3: For any $1 \leq k \leq K_b$, let j_k be the iteration when I_b^k starts, since $\{\psi_n^{j_k+1}(\cdot)\}_{n\in\mathcal{T}} = \{\psi_n^{j_k}(\cdot)\}_{n\in\mathcal{T}}$ and by assumption (A3), we have $\{x_n^{j_k+1}, y_n^{j_k+1}\}_{n\in\mathcal{T}} = \{x_n^{j_k}, y_n^{j_k}\}_{n\in\mathcal{T}}$. Because the solution $\{x_n^{j_k}, y_n^{j_k}\}_{n\in\mathcal{T}}$ is not optimal, by Claim 2, there exists $n_{j_k} \in \mathcal{T}$ such that $\psi_{n_{j_k}}^{j_k}(x_{n_{j_k}}^{j_k}) < \mathcal{Q}_{n_{j_k}}(x_{n_{j_k}}^{j_k})$. Choose such an node n_{j_k} so that $t(n_{j_k})$ is the largest, hence for all $m \in \mathcal{C}(n_{j_k})$, $\psi_m^{j_k}(x_m^{j_k}) = \mathcal{Q}_m(x_m^{j_k})$. The sampling in the forward step is

done with replacement, thus each scenario is sampled independently. Since there are finitely many scenarios, and each one is sampled with a positive probability, we know that with probability 1, after finitely many number of iterations, a scenario that contains node n_{j_k} will be sampled in an iteration, say j'_k. In the backward step of iteration j'_k, the same state vector $x^{j'_k}_{n_{j_k}} = x^{j_k}_{n_{j_k}}$ will be evaluated at all children nodes of n_{j_k}, and a cut will be added to $\psi^{j'_k}_{n_{j_k}}(\cdot)$. We want to show that $\psi^{j'_k+1}_{n_{j_k}}(x^{j_k}_{n_{j_k}}) = \mathcal{Q}_{n_{j_k}}(x^{j_k}_{n_{j_k}})$ after adding this cut.

Note that we have the following relations:

$$\psi^{j'_k+1}_{n_{j_k}}(x^{j_k}_{n_{j_k}}) \geq \sum_{m\in\mathcal{C}(n_{j_k})} q_{n_{j_k}m}(v^{j_k}_m + (\pi^{j_k}_m)^\top x^{j_k}_{n_{j_k}}) \tag{3.25a}$$

$$= \sum_{m\in\mathcal{C}(n_{j_k})} q_{n_{j_k}m}\underline{Q}^{j_k}_m(x^{j_k}_{n_{j_k}}, \psi^{j_k}_m) \tag{3.25b}$$

$$= \sum_{m\in\mathcal{C}(n_{j_k})} q_{n_{j_k}m}(f_m(x^{j_k}_m, y^{j_k}_m) + \psi^{j_k}_m(x^{j_k}_m)) \tag{3.25c}$$

$$= \sum_{m\in\mathcal{C}(n_{j_k})} q_{n_{j_k}m}(f_m(x^{j_k}_m, y^{j_k}_m) + \mathcal{Q}^{j_k}_m(x^{j_k}_m)) \tag{3.25d}$$

$$\geq \sum_{m\in\mathcal{C}(n_{j_k})} q_{n_{j_k}m}Q^{j_k}_m(x^{j_k}_{n_{j_k}}, \psi^{j'_k}_m) \tag{3.25e}$$

$$= \mathcal{Q}_{n_{j_k}}(x^{j_k}_{n_{j_k}}). \tag{3.25f}$$

The inequality in (3.25a) follows from the construction of $\psi^{j'_k+1}_{n_{j_k}}(x^{j_k}_{n_{j_k}})$ in (3.7). The equality in (3.25b) follows from the fact that $(v^{j_k}_m, \pi^{j_k}_m)$ is a tight cut for the relaxation problem of $(P^{j_k}_m(x^{j_k}_{n_{j_k}}, \psi^{j_k+1}_m))$ and uses the definition of tight cuts given in (3.9). The equality in (3.25c) follows from the definition of $\underline{Q}^{j_k}_m$ in (3.6). The equality (3.25d) holds due to the fact for all $m \in \mathcal{C}(n_{j_k})$, $\psi^{j_k}_m(x^{j_k}_m) = \mathcal{Q}_m(x^{j_k}_m)$. Then, (3.25e) follows because $(x^{j_k}_m, y^{j_k}_m)$ is a feasible solution of the problem $(P^{j_k}_m(x^{j_k}_{n_{j_k}}, \psi^{j_k+1}_m))$ with the parent state $x^{j_k}_{n_{j_k}}$ as defined in (3.5). Lastly, (3.25f) is the definition of $\mathcal{Q}_{n_{j_k}}(x^{j_k}_{n_{j_k}})$.

Since $\psi^{j'_k+1}_{n_{j_k}}(x^{j_k}_{n_{j_k}}) = \mathcal{Q}_{n_{j_k}}(x^{j_k}_{n_{j_k}})$, a new Type-a occurrence starts from the j'_k-th iteration. In other words, when the SND algorithm has not converged, i.e., $(x^i_n, y^i_n)_{n\in\mathcal{T}}$ is not optimal,

each consecutive Type-b occurrence is followed by a Type-a iteration. This proves $K_b \leq K_a$. Therefore, the number of iterations in I_b^k for $1 \leq k \leq K_b$ is finite with probability 1. $\diamond$

It follows from Claim 3 that the condition in Claim 2 will hold after $K = \sum_{k=1}^{K_a} |I_a^k| + \sum_{k=1}^{K_b} |I_b^k|$ iterations. We have the following relations.

$$\begin{aligned} 1 &\geq Pr\left(\sum_{k=1}^{K_a} |I_a^k| + \sum_{k=1}^{K_b} |I_b^k| < \infty\right) \\ &= Pr\left(\sum_{k=1}^{K_b} |I_b^k| < \infty\right) \\ &= Pr\left(|I_b^k| < \infty, \forall\, 1 \leq k \leq K_b\right) = 1, \end{aligned}$$

where the first equality follows from the finiteness of $\sum_{k=1}^{K_a} |I_a^k|$ and the second is due to $K_b < \infty$ for sure, and the last follows from Claim 3. Hence $Pr(K < \infty) = 1$. Therefore, the SND algorithm converges to an optimal solution of problem (3.3) in a finite number of iterations with probability 1. $\square$

3.6.3 The SDDiP Algorithm

We now propose the SDDiP algorithm for the setting where the scenario tree satisfies stage-wise independence, i.e., for any two nodes n and n' in $\mathcal{S}_t$ the set of children nodes $\mathcal{C}(n)$ and $\mathcal{C}(n')$ are defined by identical data and conditional probabilities. In this case, the value functions and expected cost-to-go functions depend only on the stage rather than the nodes, i.e., we have $\mathcal{Q}_n(\cdot) \equiv \mathcal{Q}_t(\cdot)$ for all $n \in \mathcal{S}_t$. As a result, only one problem is maintained per stage, and cuts generated from different candidate solutions are added to the same problem.

We consider the setting where the scenario tree is created by sampling a stage-wise independent stochastic process. Let N_t be the number of realizations of uncertain parameters at stage $t = 2, \ldots, T$, each outcome has an equal probability of $1/N_t$. The total number of scenarios is $N = \prod_{t=2}^{T} N_t$. For any $1 \leq t \leq T$ and $i \geq 1$, let $\psi_t^i(\cdot)$ be the approximate expected cost-to-go function in stage t at the beginning of iteration i (cf. (3.6)-(3.7)). For a

particular uncertain data realization $\xi_t^k (1 \leq k \leq N_t)$ in stage t, let $(P_t^i(x_{t-1}^{ik}, \psi_t^i, \xi_t^k))$ be the corresponding stage problem given state variable x_{t-1}^{ik} at the beginning of iteration i, and denote its optimal solution by $(x_t^{ik}, y_t^{ik}, z_t^{ik}, \theta_t^{ik})$. In the backward step, given a candidate solution x_{t-1}^{ik}, let (R_t^{ik}) be a suitable relaxation of the updated problem $(P_t^i(x_{t-1}^{ik}, \psi_t^{i+1}, \xi_t^j))$ for some $1 \leq j \leq N_t$, and (v_t^{ij}, π_t^{ij}) be the corresponding cut coefficients collected from solving the relaxation problem. Since each outcome of the uncertain data process has the same probability, the cut (3.7b) is obtained by taking the average of all generated cut coefficients, i.e.,

$$\theta_t \geq \frac{1}{N_t} \sum_{j=1}^{N_t} (v_t^{ij} + (\pi_t^{ij})^\top x_{t-1}). \tag{3.26}$$

The SDDiP algorithm is described in Algorithm 4, and its almost sure convergence immediately follows from Theorem 9.

For the problem with right hand side uncertainty, simple stage-wise dependency, e.g., p-th order autoregressive model, can be transformed into the independent case by adding additional decision variables (Shapiro et al. 2013). However this approach in general does not extend to the situation where uncertainty exists in the objective coefficients or left hand side matrix of constraints because bilinear terms will be introduced but cannot be handled by the standard SDDP method. In our setting, however, these bilinear terms are products of two binary variables after reformulation using binary expansion or approximation, which can be easily reformulated as linear constraints. This is another significant advantage of considering the 0-1 state space.

3.7 Computational Experiments

In this section, we present computational experiments to evaluate the SDDiP Algorithm 4 on a power generation expansion planning problem, a financial portfolio optimization problem, and an airline revenue management problem. Algorithm 4 is implemented in C++ with CPLEX 12.6.0 to solve the MIP and LP subproblems. The Lagrangian dual problem is

Algorithm 4 :: Stochastic Dual Dynamic Integer Programming

Initialize: $LB \leftarrow -\infty$, $UB \leftarrow +\infty$, $i \leftarrow 1$, and an initial lower approximation $\{\psi_t^1(\cdot)\}_{t=1,\ldots,T}$
while some stopping criterion is not satisfied **do**
  Sample M scenarios $\Omega^i = \{\xi_1^k, \ldots, \xi_T^k\}_{k=1\ldots,M}$

  /* Forward step */
  for $k = 1, \ldots, M$ **do**
    for $t = 1, \ldots, T$ **do**
      solve forward problem $P_t^i(x_{t-1}^{ik}, \psi_t^i, \xi_t^k)$
      collect solution $(x_t^{ik}, y_t^{ik}, z_t^{ik}, \theta_t^{ik} = \psi_t^i(x_t^{ik}))$
    end for
    $u^k \leftarrow \sum_{t=1,\ldots,T} f_t(x_t^{ik}, y_t^{ik}, \xi_t^k)$
  end for

  /* (Statistical) upper bound update */
  $\hat{\mu} \leftarrow \frac{1}{M}\sum_{k=1}^{M} u^k$ and $\hat{\sigma}^2 \leftarrow \frac{1}{M-1}\sum_{k=1}^{M}(u^k - \hat{\mu})^2$
  $UB \leftarrow \hat{\mu} + z_{\alpha/2}\frac{\hat{\sigma}}{\sqrt{M}}$

  /* Backward step */
  for $t = T, \ldots, 2$ **do**
    for $k = 1, \ldots, M$ **do**
      for $j = 1, \ldots, N_t$ **do**
        solve a suitable relaxation (R_t^{ij}) of the updated problem $P_t^i(x_{t-1}^{ik}, \psi_n^{i+1}, \xi_t^j)$ and collect cut coefficients (v_t^{ij}, π_t^{ij})
      end for
      add cut (3.26) to ψ_{t-1}^i to get ψ_{t-1}^{i+1}
    end for
  end for

  /* Lower bound update */
  solve $P_1^i(\bar{x}_0, \psi_1^{i+1})$ and set LB to the optimal value
  $i \leftarrow i + 1$
end while

solved to optimality using a basic subgradient algorithm (see e.g., Bertsekas 1999, Sec. 6.3) with an optimality tolerance of 10^{-4}. All other relative MIP tolerance is set to 10^{-4} except when specified. All computations are conducted on a Linux (Fedora 22) desktop with four 2.4GHz processors and 8GB RAM.

3.7.1 Long-term Generation Expansion Planning

In a power generation expansion planning (GEP) problem, one seeks to determine a long-term construction and generation plan for different types of generators, taking into account the uncertainties in future demand and natural gas prices. Suppose there are n types of expansion technologies available. Let x_t be a vector representing numbers of different types of generators to be built in stage t, and y_t be a vector of the amount of electricity produced by each type of generator per hour in stage t. A deterministic formulation is as follows.

$$
\begin{aligned}
\min \quad & \sum_{t=1}^{T} (a_t^\top x_t + b_t^\top y_t) && \text{(investment cost + generation cost)} \\
\text{s.t.} \quad & \forall\, t = 1, \ldots, T \\
& \sum_{s=1}^{t} x_s \geq A_t y_t && \text{(generation capacity)} \\
& \sum_{s=1}^{t} x_s \leq \bar{u} && \text{(limitation on total number of generators)} \\
& \mathbf{1}^\top y_t = d_t && \text{(demand satisfaction)} \\
& x_t \in \mathbb{Z}_+^n,\ y_t \in \mathbb{R}_+^n.
\end{aligned}
$$

In the above formulation, a_t and b_t are investment and generation cost at stage t, respectively. Matrix A_t contains maximum rating and maximum capacity information of generators, $\bar{u}$ is a pre-determined construction limits on each type of generators due to resource and regulatory constraints, and d_t is the electricity demand at stage t.

Scenario generation Among all data, $\{b_t\}_{t=1,\ldots,T}$ and $\{d_t\}_{t=1,\ldots,T}$ are subject to uncertainty. All data (except demand and natural gas price) used in this numerical study can be found in Jin et al. (2011), where demand and natural gas price are modeled as two correlated geometric Brownian motions. We simplify the stochastic processes of electricity demand and natural gas price as follows. We assume that both processes are stage-wise

independent. At each stage, electricity demand follows a uniform distribution, and natural gas price follows a truncated normal distribution with known first and second moments. In addition, these two processes are considered as independent to each other. There are six types of generators available for capacity expansion, namely Coal, Combined Cycle (CC), Combined Turbine (CT), Nuclear, Wind, and Integrated Gasification Combined Cycle (IGCC). Among these six types of generators, both CC and CT power generators are fueled by natural gas.

In the implementation, we create a new set of general integer variables s_t, representing the cumulative numbers of different types of generators built until stage t. After binary expansion, there are 48 binary state variables per stage. The local variables are x_t and y_t, containing 6 general integer variables and 7 continuous variables, respectively.

Performance Comparison We first consider an instance of the GEP problem with 10 decision stages. At each stage, three realizations of the uncertainty parameters are drawn, thus in total there are $3^9 = 19683$ scenarios with equal probability. We construct the extensive formulation on the scenario tree and use CPLEX to solve the problem as one large MIP. This formulation contains nearly 620,000 binary variables and 207,000 continuous variables. CPLEX returns an incumbent solution with an objective function value 7056.7, and the best bound 6551.6, i.e., a 7.16% gap remains after two hours.

We solve the same instance using SDDiP algorithm with seven different combinations of cutting planes and compare their performance. Each of the combinations includes at least one collection of tight cuts. The stopping criterion used in this numerical test is to terminate the algorithm once lower bounds obtained in the backward steps become stable, and the computation time limit is set to be 5 hours. After the lower bounds become stable, we evaluate the objective function value for 1500 forward paths independently, and construct a 95% confidence interval. The right endpoint of this interval is reported as the statistical upper bound of the optimal value. The seven combinations of cuts are specified below:

(1) Integer optimality cut (I);

(2) Lagrangian cut (L);

(3) Benders' cut + Integer optimality cut (B + I);

(4) Benders' cut + Lagrangian cut (B + L);

(5) Strengthened Benders' cut + Integer optimality cut (SB + I);

(6) Strengthened Benders' cut + Lagrangian cut (SB + L);

(7) Strengthened Benders' cut + Integer optimality cut + Lagrangian cut (SB + I + L).

In Table 3.1, we compare the performance of the SDDiP algorithm with integer optimality cuts (I) and Lagrangian cuts (L). The first column indicates the type of cuts; Column 2 represents the number of forward paths sampled in the forward step; Column 3 contains the best lower bound computed by the algorithm when stopping criterion (or computation time limit) is reached; Column 4 shows the average number of iterations used; Column 5 contains a 95%-confidence statistical upper bound on the optimal value; Column 6 shows the gap between the statistical upper bound and the best lower bound in Column 2; and the last two columns contain the average total computation time and time used per iteration for each experiment setting.

From Table 3.1 we can see that, if only integer optimality cuts are used in the backward step, the lower bound improves very slowly. As a result, it takes a long time for the algorithm to stop. In fact, none of the experiments converges within 5 hours of computation time and large gaps are observed between the lower and upper bounds on the optimal values. In comparison, if only Lagrangian cuts are used, the algorithm converges much faster. The lower bounds obtained are also significantly higher than those attained only with integer optimality cuts. In addition, for the Lagrangian cuts, the gap between the statistical upper bound and the deterministic lower bound is very small in all experiments with different choices of the number of forward sample paths. The reason behind these results should be clear from the construction of integer optimality cuts. Namely, they are much looser than

Table 3.1: SDDiP algorithm with a single class of cutting planes

cuts	# FW	best LB	# iter	stat. UB	gap	time (sec.)	time/iter.
	1	4261.1	4041	8999.1	52.65%	18000	4.5
	2	4184.5	2849	9005.5	53.53%	18000	6.3
	3	4116.2	2426	10829.9	61.99%	18000	7.4
I	5	3970.4	1908	9730.0	59.19%	18000	9.4
	10	3719.8	1384	9868.5	62.31%	18000	13.0
	20	3427.8	969	10011.1	65.76%	18000	18.6
	50	3055.8	603	10002.9	69.45%	18000	29.9
	1	6701.1	110	6762.4	0.91%	1810	16.5
	2	6701.1	57	6781.9	1.19%	1021	18.0
	3	6701.0	45	6769.5	1.01%	1595	35.5
L	5	6701.1	36	6851.8	2.20%	741	20.6
	10	6701.3	34	6796.6	1.40%	1223	36.0
	20	6701.2	28	6803.3	1.50%	1274	45.5
	50	6701.1	30	6801.6	1.48%	2092	69.7

Lagrangian cuts everywhere else except at the candidate solution being evaluated.

Table 3.2 presents similar computational results but in addition to using a single class of tight cuts (i.e. I or L), we further adopt either Benders' cuts or strengthened Benders' cuts (i.e. B or SB). We have the following comparisons.

1. *(B+I) v.s. I*: It is observed that adding Benders' cuts together with integer optimality cuts (B+I) leads to a significant improvement of the algorithm performance, comparing to the performance of only integer optimality cuts (I) in Table 3.1. Not only all experiments converge within 5 hours, the quality of the solutions is also very satisfactory, i.e., the gap between the statistical upper bound and deterministic lower bound is small ($\leq 2\%$) in most cases.
2. *(B+I) v.s. (SB+I) and (B+L)*: Another significant improvement on the algorithm performance can be observed by comparing (B + I) and (SB + I) of Table 3.2, where we substitute Benders' cuts with strengthened Benders' cuts. We can still attain small gaps, i.e., good estimations on the optimal value. Moreover, the number of iterations, the total time, and the average computation time all significantly decrease due to

Table 3.2: SDDiP algorithm with multiple classes of cutting planes

cuts	# FW	best LB ($MM)	# iter	stat. UB ($MM)	gap	time (sec.) (sec.)	time/iter. (sec.)
B + I	1	6701.1	399	6874.7	2.53%	3905	9.8
	2	6701.1	263	6757.1	0.83%	3524	13.4
	3	6701.0	204	6755.8	0.81%	3594	17.6
	5	6701.1	173	6799.5	1.44%	4457	25.8
	10	6701.1	146	6752.9	0.77%	5579	38.1
	20	6701.1	137	6874.3	2.52%	8167	59.8
	50	6701.1	135	6840.1	2.03%	14719	109.0
B + L	1	6701.1	70	6772.7	1.06%	467	7.1
	2	6701.1	56	6753.9	0.78%	632	14.8
	3	6701.1	38	6831.0	1.90%	546	15.7
	5	6701.2	34	6807.0	1.56%	752	20.8
	10	6701.0	24	6818.6	1.72%	737	32.7
	20	6700.9	23	6838.3	2.01%	952	39.1
	50	6701.1	21	6843.5	2.08%	1230	60.5
SB + I	1	6700.3	178	6808.1	1.58%	461	2.6
	2	6701.0	114	6825.9	1.82%	643	5.7
	3	6701.1	95	6800.6	1.46%	618	6.5
	5	6701.1	35	6768.4	0.99%	624	9.5
	10	6701.1	31	6763.0	0.91%	760	14.9
	20	6701.1	25	6803.9	1.51%	814	20.7
	50	6701.1	27	6860.6	2.32%	1239	32.4
SB + L	1	6701.0	61	6808.5	1.58%	401	6.6
	2	6701.0	40	6788.5	1.29%	457	11.6
	3	6701.0	33	6766.3	0.97%	496	14.9
	5	6701.1	29	6827.9	1.86%	621	21.8
	10	6701.0	22	6768.9	1.00%	611	28.1
	20	6701.1	20	6761.2	0.89%	767	37.7
	50	6701.1	20	6783.9	1.22%	1083	53.3
SB + I + L	1	6701.0	57	6800.5	1.46%	437	7.6
	2	6701.0	42	6763.5	0.92%	582	14.0
	3	6701.0	30	6817.1	1.70%	404	13.8
	5	6701.1	27	6783.4	1.21%	527	19.3
	10	6701.0	21	6835.1	1.96%	580	28.1
	20	6701.1	21	6796.8	1.41%	772	36.7
	50	6701.1	20	6813.3	1.65%	960	47.2

the tighter strengthened Benders' cuts. Comparing (B + I) with (B + L) suggests that replacing integer optimality cuts with Lagrangian cuts also results in a major improvement in both the total number of iterations and computation time.

3. *(SB+I) v.s. (SB+L) and (SB+I+L)*: No significant improvement is observed between (SB + I) and (SB + L). This is because the optimal Lagrangian dual multipliers do not deviate much from the LP dual optimal in these instances. Therefore, strengthened Benders' cuts and Lagrangian cuts are "similar" in this sense. Finally, adding integer optimality cuts in addition to the strengthened Benders' and Lagrangian cuts (SB + I + L) does not significantly affect algorithm performance, because integer optimality cuts do not contribute much in approximating the expected cost-to-go functions except at the candidate solutions.

As we increase the number of sample paths evaluated in the forward step, the total computation time as well as the time used per iteration increase in general. The more scenarios are selected in the forward step, the more subproblems need to be solved, and it is often the case that more candidate solutions will be generated and evaluated in the backward step. A significant advantage of using only 1 sample path in the forward step was reported in Shapiro et al. (2013). Similar results can be observed in our experiments. Though for some instances (e.g., B + I), a slightly bigger number (e.g., 3) of forward paths results in better performance of SDDiP algorithm. In general, the best choice of forward sample size remains small (1, 2, or 3). Moreover, in the experiments where Lagrangian cuts are used, the time used per iteration is usually longer. Since generating integer optimality cuts only requires solving the subproblem as an integer program, whereas one needs to solve a Lagrangian dual problem to get a Lagrangian cut, and the basic subgradient method usually takes more time. A visualization summary of the final gap and computation time for each cut combination discussed above can be found in Figure A1 and A2 in Appendix.

To better illustrate the contribution of each type of cuts, Figure 3.1 contains the deterministic lower bounds and stochastic upper bounds in the first 50 iterations of the

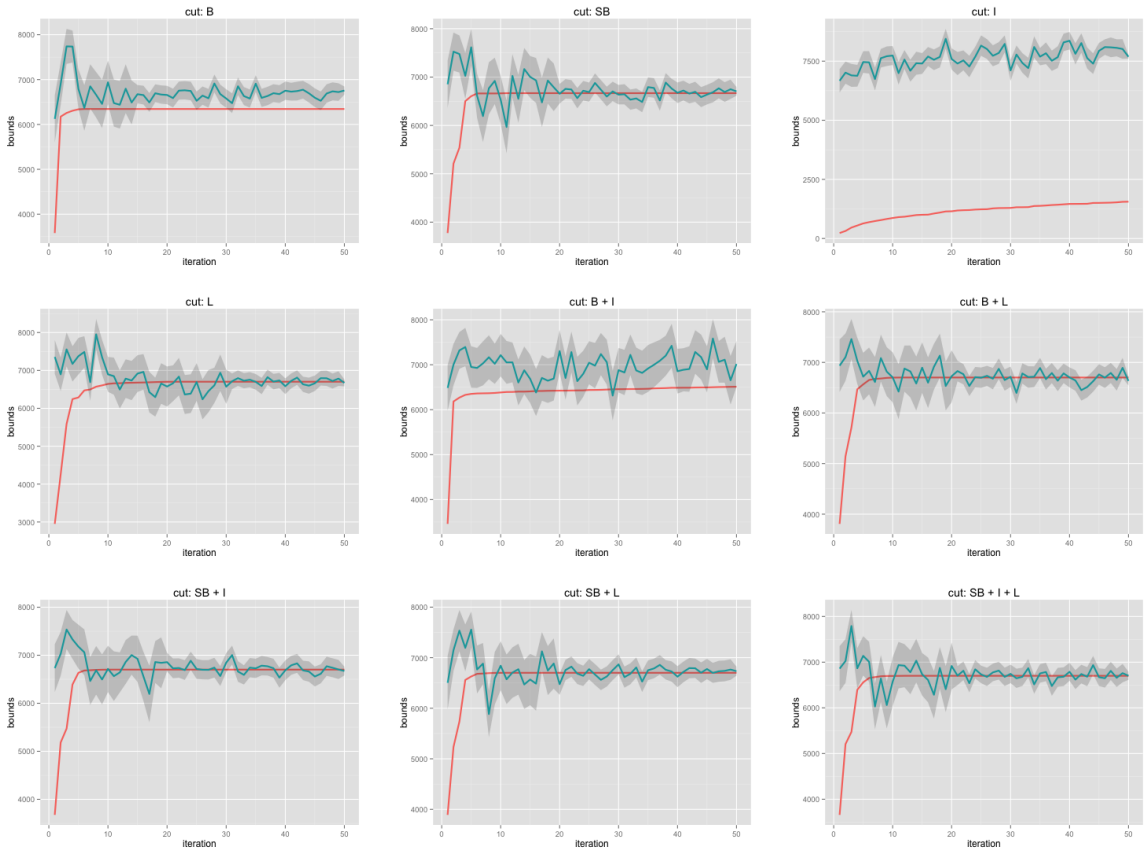


Figure 3.1: Bounds improvement with different cut combinations

SDDiP algorithm, for different cut combinations. In addition to the seven combinations mentioned above, we also include using only Benders' or strengthened Benders' cuts. In Figure 3.1, the 95% confidence intervals for the upper bounds are constructed by evaluating 50 forward sample paths each iteration till the lower bounds become stable, then the number is increased to 300 afterwards. If the algorithm converges before the 50th iteration, to make each subfigure have the same horizontal axis, we keep evaluating the upper bounds by sampling 300 forward sample paths per iteration. It is clear that using only Benders' cuts in the SDDiP algorithm does not close the gap between the upper and lower bounds. Even though strengthened Benders' cuts are not tight in general, the lower and upper bounds are much closer comparing to using only Benders' cuts. Using a single collection of integer optimality cuts results in a very slow improvement of the lower bounds. Finally, while using only Lagrangian cuts leads to a faster convergence, the computation time requirement

indicated in Table 3.1 makes it less appealing. Further computational improvement to obtain the Lagrangian cuts will be discussed in Chapter 4.

In summary, cut combinations (B + L), (SB + I), (SB + L), and (SB + I + L), appear to be good choices for the SDDiP algorithm. In the case where the Lagrangian dual problem is difficult to solve, strengthened Benders' cuts and integer optimality cuts yield a better performance.

Scalability To further test the scalability of the algorithm, we generate several large-scale instances with planning horizons ranging from 5 to 9, and each period contains 30 to 50 realizations of the uncertain parameters, which are sampled independently from their distributions.The extensive scenario tree formulation (3.3) for these instances contains as many as 11 trillion binary variables, so it is impossible to expect any solver can solve such a problem as a single MIP. However, the SDDiP algorithm is able to estimate the optimal values of these instances with very high accuracy, as shown in Table 3.3.

In Table 3.3, Column 1 indicates the planning horizon of the corresponding instance, Column 2 shows the number of branches of each node in the scenario tree, and Column 3 indicates the cut combinations used in the backward step. In these instances, we do not enforce computation time limit, the algorithm stops when the lower bounds become stable. In all experiments, we achieve good estimates on the optimal value (small gaps between upper and lower bounds) within a reasonable computation time. Notice that the reduction in the number of iterations and computation time from cut combination (B + I) to (SB + I) or (B + L) is significant. Moreover, the time per iteration is also significantly reduced even though SB and L require solving additional integer subproblems. This is perhaps because the later iterations, where more cuts are accumulated, take longer time, and using SB and L reduces the iteration count. The difficulty and time requirement for solving Lagrangian dual problems can be observed by comparing cut combination (SB + I + L) with (SB + I). Although the number of iterations decreases after adding Lagrangian

Table 3.3: SDDiP algorithm on some large instances of GEP

T	# branch	cuts	best LB ($MM)	# iter	stat. UB ($MM)	gap	time (hr.)	time/iter (sec.)
5	50	B + I	2246.4	92	2260.7	0.63%	0.96	37.6
		SB + I	2246.4	34	2278.2	1.39%	0.09	9.4
		B + L	2246.4	34	2279.6	1.45%	0.19	20.3
		SB + L	2246.4	21	2276.4	1.32%	0.14	23.4
		SB + I + L	2246.4	25	2279.4	1.45%	0.11	15.4
6	50	B + I	2818.8	237	2840.6	0.77%	2.24	34.0
		SB + I	2818.9	74	2855.8	1.29%	0.60	29.0
		B + L	2818.9	63	2848.5	1.04%	0.96	54.7
		SB + L	2818.9	56	2849.2	1.06%	0.70	45.2
		SB + I + L	2818.9	50	2820.7	0.06%	1.03	73.9
7	50	B + I	3564.5	239	3614.8	1.39%	8.10	122.0
		SB + I	3564.4	111	3588.9	0.68%	1.08	34.9
		B + L	3564.5	100	3569.1	0.13%	2.48	89.2
		SB + L	3564.5	66	3576.9	0.35%	2.37	129.0
		SB + I + L	3564.5	69	3577.6	0.37%	1.95	101.6
8	30	B + I	4159.4	340	4254.2	2.23%	7.78	82.4
		SB + I	4159.4	152	4207.5	1.14%	1.53	36.3
		B + L	4159.6	147	4227.7	1.61%	4.00	97.9
		SB + L	4159.6	87	4218.9	1.41%	2.55	105.4
		SB + I + L	4159.6	103	4278.0	2.77%	2.72	94.9
9	30	B + I	5058.0	520	5081.5	0.46%	19.85	137.4
		SB + I	5058.6	230	5102.0	0.85%	2.57	40.2
		B + L	5058.7	218	5108.6	0.98%	8.01	132.3
		SB + L	5058.7	120	5145.3	1.68%	2.96	88.8
		SB + I + L	5058.9	119	5079.4	0.40%	4.39	132.8

cuts (which implies that these cuts provide better approximation than integer optimality cuts), both the total computation time and time used per iteration increase considerably. We finally point out that in all our experiments, the combination of strengthened Benders' cuts and integer optimality cuts (SB + I) outperforms other combinations in terms of total computation time. A visualization summary of number of iterations and computation time for each cut combination discussed above can be found in Figure A3 and A4 in Appendix.

These computational results demonstrate that the SDDiP algorithm with the proposed

cuts successfully estimates the optimal value of large-scale generation capacity expansion problems with high accuracy and reasonable computation time.

3.7.2 Multi-period Portfolio Optimization

In this section, we test SDDiP algorithm on a multi-period portfolio optimization problem (see e.g., Dantzig and Infanger 1993), where the uncertain parameters are the returns of different assets in each period. In this problem, the objective is to maximize the expected return over a fixed length of time periods, by adjusting the holding position of each type of asset. Each completed transaction will incur a certain amount of fee, referred as transaction cost, which is assumed to be a proportional cost to the total value of assets involved in the corresponding transaction. At any time period, the total number of assets possessed is restricted to be less than some prescribed threshold.

In particular, we consider n types of stocks and a risk-free asset (the $(n+1)$-th asset) over a T-period investment horizon. Let x_t be a vector denoting the values of assets at period t, and z_t be a binary vector, representing whether the account holder owns each asset at period t. The account holder decides how much of each stock to buy (b_t) or sell (s_t) at period t, with return information $r_0, \ldots, r_{t-1}$ which have been realized. We assume that the initial risk-free asset value is $\bar{x}_0$ and all others are 0. A deterministic model is as follows:

$$
\begin{aligned}
\max \quad & r_T^\top x_T \\
\text{s.t.} \quad & \forall t = 1, \ldots, T, \\
& x_{ti} = r_{t-1,i} x_{t-1,i} + b_{t,i} - s_{t,i} \;\; \forall i = 1, \ldots, n, && \text{(transaction flow balance)} \\
& x_{t,n+1} = r^f x_{t-1,n+1} - (\mathbf{1} + \alpha_b)^\top b_t + (\mathbf{1} - \alpha_s)^\top s_t, && \text{(self-financing)} \\
& x_t \le M z_t, \; s_{ti} \le r_{t-1,i} x_{t-1,i}, \; \forall i = 1, \ldots, n, && \text{(variable relationships)} \\
& \mathbf{1}^\top z_t \le K, && \text{(number of assets possessed)} \\
& x_0 = [0, \ldots, 0, \bar{x}_0]^\top,
\end{aligned}
$$

$$z_t \in \{0,1\}^n, 0 \leq b_t, s_t \leq u, 0 \leq x_t \leq v,$$

where α_b and α_s are the transaction cost coefficients for buy and sell, respectively, and u, v are implied bounds on variables. For the stochastic model, the uncertainty is in the return vector r.

Scenario Generation We test the problem on all the stocks from the S&P 100 index. The optimization problem has an investment horizon of 5 to 12 periods, each of which is a two-week (10 business days) span. The scenarios of returns for each stock are generated using historical returns data without assuming specific distributions. In particular, we collect 500 bi-weekly returns over the past 2 years for each stock, and regard these 500 overlapping returns as the universe of all possible return realizations for each investment period. Then we sample (with replacement) a subset of realizations at each period independently to form a recombining scenario tree. To preserve the correlation between different stocks, the sampled scenario contains a return vector in which all components correspond to the same time span. In the scenario tree, the number of branches ranges from 10 to 20.

Note that in this problem, x_t are continuous state variables. We will resort to the binary approximation discussed in Section 3.3. We assume that at the beginning the account holder has 100 units of cash and none of the stocks. The continuous state variables are approximated using the binary expansion with approximation accuracy $\varepsilon = 10^{-2}$. Each stage subproblem contains approximately 1500 binary state variables. The local variables are z_t, b_t, and s_t, each has a dimension of 100.

Algorithm Performance Table 3.4 summarizes the performance of SDDiP algorithm on the test instances. Since this is a maximization problem, the negation of the lower bound reported by SDDiP algorithm is a valid upper bound on the true optimal value (Column 5). The algorithm also produces a statistical lower bound on the expected return (Column 6), obtained by evaluating 500 sample paths independently after the upper bounds become

Table 3.4: SDDiP algorithm on portfolio optimization

T	# branch	# scen	# FW	Best UB	Stat. LB	gap	time (sec)
4	10	1000	1	108.1	105.7	0.66%	185
			2	108.1	106.4	1.33%	210
			5	108.1	106.3	1.41%	313
			10	108.1	106.1	1.00%	456
	15	3375	1	106.9	105.1	1.10%	309
			2	106.9	104.4	1.42%	356
			5	106.9	104.6	1.07%	518
			10	106.9	104.3	0.36%	884
	20	8000	1	108.1	106.2	1.05%	418
			2	108.1	106.1	1.63%	423
			5	108.1	105.0	1.25%	630
			10	108.1	106.1	1.49%	1027
5	10	10000	1	116.1	112.9	1.49%	343
			2	116.1	112.0	1.79%	414
			5	116.1	112.8	1.30%	580
	15	50625	1	109.6	106.9	1.65%	567
			2	109.6	106.6	0.98%	686
			5	109.6	106.3	2.07%	933
	20	160000	1	109.0	106.9	1.45%	425
			2	109.0	106.1	1.49%	715
			5	109.0	106.3	1.54%	1156
6	20	3.2×10^6	1	112.2	109.1	1.14%	704
			2	112.2	109.8	1.58%	1091
			5	112.2	108.2	2.08%	1573
7	20	6.4×10^7	1	116.5	112.8	1.71%	938
			2	116.5	112.8	1.24%	1201
			5	116.5	112.8	1.64%	2008
8	15	1.7×10^8	1	120.57	119.29	1.08%	1182
10	10	10^9	1	125.21	122.43	2.27%	1032
12	10	10^{11}	1	129.79	126.83	2.33%	1299

stable. Column 1 shows the time horizon of the test instances; Columns 2 and 3 contain information of the scenario tree, i.e., number of branches of each node and total number of scenarios; Column 4 indicates how many forward sample paths are used in the forward step; Columns 7 and 8 report the gaps between the lower and upper bounds on the optimal values,

and the total computation time, respectively.

The stopping criterion remains the same, i.e., the algorithm stops when the deterministic upper bounds become stable. Among all test instances, the algorithm reaches the stopping criterion within 10 iterations, and gaps between the upper bound and the statistical lower bound are all small. We solve the extensive scenario tree formulation for the first two instances $T = 4$, #branch $= 10$ and 15 as two examples to demonstrate the accuracy of attained upper bounds. The first instance has an optimal value of 108.02 and the second is 106.8. The gap between the lower and upper bounds mostly come from the evaluation of lower bounds, and can be made smaller by evaluating more forward paths. Similar to the generation capacity expansion example, we observe that it is more efficient to use a small number of sample paths in the forward iteration. Note that we generate a different scenario tree for each instance (T, #branch), thus the optimal values are not necessary monotone.

3.7.3 Airline Revenue Management

In the airline industry, revenue management usually refers to dynamic pricing and controlling seat sales based on the passenger demand forecast in a flight network. In this section, we focus on the latter approach. The objective is to maximize the revenue generated from ticket sales. We consider a multistage stochastic model which is similar to the one in Möller et al. (2008). A deterministic formulation of such a problem is given as follows.

$$
\begin{aligned}
\max \quad & \sum_{t=1}^{T} \left[(f_t^b)^\top b_t - (f_t^c)^\top c_t \right] \\
\text{s.t.} \quad & \forall t = 1, \ldots, T, \\
& B_t = B_{t-1} + b_t,\ C_t = C_{t-1} + c_t \\
& C_t = \lfloor \Gamma_t B_t + 0.5 \rfloor \\
& A(B_t - C_t) \leq R,\ b_t \leq d_t \\
& B_0 = \bar{B}_0,\ C_0 = \bar{C}_0
\end{aligned}
$$

$$B_t, C_t, b_t, c_t \in \mathbb{Z}_+^m.$$

In the above formulation, T is the number of booking intervals. The numbers of fulfilled bookings (resp. cancellations) of period t and cumulative fulfilled bookings (resp. cancellations) up to period t are denoted by b_t (resp. c_t) and B_t (resp. C_t). Each of these quantities is an m-dimensional vector, whose components correspond to particular origin-destination itineraries and fare classes. f_t^b and f_t^c are the booking price and refund for cancellation at period t, respectively. The matrix Γ_t is a diagonal matrix, whose elements are the cancellation rate of each type of tickets. Passenger demand is denoted by d_t, which is subject to uncertainty. The seat capacity on each leg is denoted by R, and A is a 0-1 matrix that indicates whether a booking request for a particular itinerary and fare class fills up one unit of capacity of each leg.

Scenario Generation The underlying flight network contains a single hub and three destinations. There are in total 6 legs and 12 itineraries. Ticket prices and refund are fixed over booking intervals. Cancellation rates for different fare classes are also given as constants. All data can be found in Möller et al. (2008). As proposed in the literature (see e.g., Boer et al. 2002; Chen and Mello 2010), the booking process is modeled by a non-homogeneous Poisson process. The total number of cumulative booking request G over the entire booking horizon for a particular itinerary and fare class is assumed to follow a Gamma distribution $G \sim \Gamma(k, \theta)$, and the corresponding arrival pattern β follows a Beta distribution $\beta \sim \text{Beta}(a, b)$. The arrival pattern determines an allocation of total booking requests among booking intervals. The cumulative booking requests up to time $t \in [0, T]$ can be represented by $D(t) = G \cdot F_\beta(t, a, b)$, where $F_\beta(t, a, b)$ is the cumulative density function of the Beta distribution. We generate the scenario tree as follows. First, we generate N_0 realizations for the cumulative booking request for each itinerary and class fare combination, and allocate them according to the corresponding arrival patterns into each

booking interval. Then, for each booking interval, N_b samples are drawn independently out of the N_0 realizations, where N_b is the number of branches of each node in the scenario tree. In this way, we obtain a recombining scenario tree which preserves stage-wise independence. It has T stages, each of which contains N_b nodes, hence there are N_b^{T-1} scenarios in total.

In this problem, the state variables are B_t and C_t, and local variables are b_t and c_t. After binary expansion, the stage problem contains about 3000 binary state variables, and the local variables are general integers with dimension 144.

Algorithm Performance We divide the booking horizon of 182 days into different numbers of booking intervals (stages), from 6 to 14 (not necessarily evenly divided), and generate scenario trees separately for each of them. The scenario tree information is contained in the first three columns of Table 3.5. We test SDDiP algorithm on these 5 instances. During the experiment, we notice that the stage subproblem is more difficult to solve than in the previous two examples, hence we relax the relative MIP tolerance from the default (10^{-4}) to 0.05. In addition, we enforce limits on both the number of total iterations (120) and computation time (5 hours).

Table 3.5: SDDiP algorithm on network revenue management

T	# branch	# scen	# iter	Best UB	Stat. LB	gap	time (sec)
6	10	10^5	120	214357	204071	5.04%	10983
8	10	10^7	120	214855	201099	6.84%	12095
10	10	10^9	120	215039	199896	7.58%	14674
12	10	10^{11}	120	210110	196237	7.07%	15413
14	10	10^{13}	120	210012	196280	7.00%	15241

Table 3.5 summarizes the results for these 5 instances. All of them terminate because of reaching the limit on number of iterations. We observe relatively larger but acceptable gaps between the lower and upper bounds on the optimal values. These relatively larger gaps could be a consequence of early termination due to the difficulty of solving the stage

problems, or possibly because the 5% relative MIP error accumulated over the stages. We would also like to note that, due to the very large scale of the underlying multistage stochastic programs, the extensive form problems can not be solved by existing solvers. Therefore, the SDDiP algorithm with proposed cuts provides a viable and systematic way to tackle these extremely challenging problems in network revenue management.

3.8 Concluding Remarks

We consider a large class of multistage stochastic integer programs in which the variables that carry information from one stage to the next are purely binary. By exploiting the binary nature of the state variables, we propose an exact and finite nested decomposition algorithm and its stochastic variants. We remark that the binary feature of the state variables and making a local copy of state variables are the key elements to the success of the approach. It allows us to construct supporting hyperplanes to the expected cost-to-go functions, which is crucial for the correctness of the method. Extensive computational experiments on three classes of real-world problems, namely electric generation expansion, financial portfolio management, and network revenue management, show that the proposed methodology may lead to significant improvement on solving large-scale, multistage stochastic optimization problems in real-world applications.

There are several interesting directions worth investigating for future research. Improvements to the integer optimality cut for two-stage stochastic integer programs are recently proposed in Angulo et al. (2016), and this may be considered for extension to the multistage setting. In addition, since we have observed that the stage problem is sometimes not very easy to solve, to further improve performance, one needs to explore the problem substructure and tailor the algorithm according to specific problems. Effective cut management strategies could be explored to keep the problem sizes small, especially in later iterations. Finally, extension of the proposed approach to the risk averse setting would be valuable. Most previous work in risk averse multistage stochastic programming is

restricted to the linear or convex settings (Shapiro 2009; Shapiro 2012; Philpott and Matos 2012; Shapiro et al. 2013; Bruno et al. 2016), it is intriguing to study how the nonlinearity of risk in the presence of integer variables affect the problem structure.

CHAPTER 4

MULTISTAGE STOCHASTIC UNIT COMMITMENT PROBLEM USING SDDIP

4.1 Introduction

Unit commitment (UC) is one of the key problems in power system operations. It is often used in a deregulated market by an independent system operator, to decide a commitment schedule of generation units for the next 24 hours or a longer period of time, under which the forecast demand can be met in the most cost efficient way. Besides satisfying the load requirements, the commitment decisions also need to satisfy certain physical constraints, e.g., generator capacity, minimum up/down time, ramping limit, as well as the flow limit of each transmission line. To ensure the reliability and security requirement of UC solutions, certain form of reserved generation capacity and $N-1$ contingency are often enforced. Due to the presence of binary variables, which are used to model generator status, together with other constraints, UC is proven to be NP-hard (Tseng 1996).

In recent years, an increasing penetration of renewable energy has cast another layer of complexity to the UC problem. Due to the intermittent nature renewable energy, the grid system needs to be more flexible when dealing with uncertainty. Stochastic optimization approaches have been utilized in UC problems to achieve this goal. There are typically two modeling approaches, namely *two-stage* and *multistage*. In a two-stage model, the commitment decisions are determined day-ahead thus the generator commitment schedule is fixed regardless what happens in real time. To accommodate uncertainty in load and renewable output, system reserve requirement is often imposed. In contrast, a multistage model handles uncertainty more dynamically. A solution to the multistage model is referred to as a *policy*, which system operator can use in real time to adjust the generator status according to actual load and renewable output. There has been a great amount of work to

solve the UC problem in both two-stage and multistage settings. We refer the reader to several comprehensive surveys Tahanan et al. 2015; Zheng et al. 2015 for the progress in the two-stage setting, and we will summarize the existing work in the multistage setting in the next section.

In this chapter, we consider an MSUC problem with uncertain net load, i.e., the difference between the total demand and the renewable output. It captures the uncertainty from both the demand and supply sides. We solve the MSUC problem with an objective to minimize the total expected cost of start-up, shut-down, generation, and possible penalties. The key contributions are summarized below.

1. **A new solution approach for MSUC.** We propose a new type of decomposition algorithm based on the SDDiP algorithm to solve large-scale MSUC problems. To the best of our knowledge, this is the first attempt to tackle the MSUC problem using a stage-wise decomposition framework. Upon termination of the algorithm, a multistage, implementable policy is returned. Operators can use such a policy in real time to deal with system uncertainties.

2. **Computational enhancement of SDDiP.** We propose several enhancement of the SDDiP algorithm to improve running time, including using the Level Method to compute the Lagrangian cuts, a "hybrid" mixed-integer and linear modeling approach with the notion of "breakstage", and a parallel implementation of the backward step in SDDiP.

3. **Computational study in large-scale MSUC applications.** Extensive computational experiments are conducted on the IEEE 14-bus and 118-bus systems. We study the effectiveness of three types of cuts for solving the MSUC problem and the impact of breakstage. Our experiments show that the proposed method can handle MSUCs with a huge number of scenarios that were considered impossible before.

4.2 Related Work

To capture the uncertainty dynamics, a scenario tree (cf. Ruszczynski and Shapiro 2003) is usually used to model the underlying stochastic process. Previous work has considered various types of uncertainty in the UC model, such as demand (Carpentier et al. 1996; Takriti et al. 1996; Sen et al. 2006), renewable output (Uçkun et al. 2016; Jiang et al. 2016), unit failure and outage (Carpentier et al. 1996; Shiina and Birge 2004), price of electricity, fuel, and reserve (Takriti et al. 2000; Li et al. 2007).

The advantage of multistage model is accompanied by the significantly higher computation requirement. As the number of decision stages increases, the number of scenarios in a scenario tree grows exponentially. Solving the deterministic mixed integer programming formulation under the scenario tree becomes almost computationally infeasible. Existing literature has addressed this issue from different perspectives. Various scenario reduction algorithms have been proposed (Dupačová et al. 2003; Gröwe-Kuska et al. 2003; Heitsch and Römisch 2003). These algorithms usually compare the scenarios according to some probabilistic metric and reduce the total number of scenarios without over compromising the approximation accuracy. Uçkun et al. (2016) divide the planning horizon into several time blocks and propose two different ways to construct "buckets" for wind scenarios in each time block. The commitment decisions are required to be the same for the scenarios within the same bucket. Such a model is essentially an intermediate formulation between the two-stage and multistage settings.

A second stream of efforts is to develop advanced decomposition algorithms for the multistage UC problem. Two different approaches have been proposed, namely *unit decomposition* and *scenario decomposition*. In the unit decomposition, the demand and spinning reserve constraints are relaxed so that each subproblem corresponds to a single generation unit. Carpentier et al. (1996) first consider such a decomposition scheme. They also observe the benefit of using augmented Lagrangian relaxation compared to the classic

version. Takriti et al. (2000) study an extension by incorporating electricity contracts and spot-market prices into the model. Nowak and Römisch (2000) uses a similar decomposition, but the single generator subproblem is solved by stochastic dynamic programming and the Lagrangian dual problem is solved by a faster and more stable proximal bundle method (cf. Kiwiel 1983). Bacaud et al. (2001) also utilizes bundle method to solve the Lagrangian dual problem. They associate a set of weights to each scenario using their probability information, these weights are reported to be critical to the algorithm performance. A Dantzig-Wolfe decomposition approach is studied in Shiina and Birge (2004), where the single-unit subproblems are solved by dynamic programming and their schedules are added back to the restricted master problem. Alternatively, the scenario decomposition approach attempts to relax the coupling constraints among scenarios, usually referred to as non-anticipativity constraints. The resulting subproblems then become deterministic, each of which corresponds to a single scenario. Different methods have been used to solve the relaxed problem, such as Progressive Hedging algorithm(Takriti et al. 1996) and Dantzig-Wolfe decomposition (Wu et al. 2007; Schulze et al. 2015).

Most of the works above has demonstrated the benefit (profit gain/cost reduction) of using the multistage approach compared to the deterministic setting where a conservative spinning reserve is usually used. A few other papers focus on evaluating the benefit of using a multistage model over a deterministic approach (Tuohy et al. 2009; Sturt and Strbac 2012; Schulze and McKinnon 2016). These simulations are mostly carried out in a rolling horizon framework.

Besides the development of scenario reduction and decomposition techniques, there have been efforts to improve the solver performance by strengthening the formulation using effective cutting planes. A majority of these works focus on the two-stage setting (e.g., Rajan and Takriti 2005; Ostrowski et al. 2012; Morales-España et al. 2013, etc.). Recently, some new results have been developed for the multistage model. Pan and Guan (2016) derive cross-scenario strong valid inequalities. They show that the proposed inequalities

describe the convex hull for some special cases. Jiang et al. (2016) extend the results in Rajan and Takriti (2005) into the stochastic setting and propose several families of strong valid inequalities via lifting schemes for the ramping and load balance polytopes. In both works, a branch-and-cut algorithm is used to demonstrate the effectiveness of the proposed cutting planes. However, existing cutting plane approaches are limited to relatively small trees.

Pereira and Pinto (1991) propose a stochastic variant of nested Benders decomposition, known as Stochastic Dual Dynamic Programming (SDDP) to solve multistage stochastic linear programs. It has been successfully applied to multistage stochastic hydro-thermal scheduling problems since then (e.g. Rotting and Gjelsvik 1992; Archibald et al. 1999; Shapiro et al. 2013). SDDP is a sampling-based algorithm and benefits from stage-wise independence of the underlying stochastic process. Due to the presence of integer recourse decisions in MSUC, such Benders-type decomposition algorithm has its own limitation, thus has only been used in a two-stage setting (see e.g. Cerisola et al. 2009b; Wang et al. 2013; Zheng et al. 2013). An extension of SDDP to solve MSIP problems, called Stochastic Dual Dynamic integer Programming (SDDiP), is proposed in Chapter 3. The algorithm is designed for MSIP with binary state variables, and can also accommodate general state variables under mild conditions. In SDDiP algorithm, state variables refer to those whose values will be passed to the next stage problem as inputs, and the rest are referred to as local variables. A new family of cuts, termed Lagrangian cuts, is proposed in this work and is shown to be the key to close optimality gap and to guarantee almost sure convergence of SDDiP.

The remainder of this chapter is organized as follows. In Section 4.3, we describe the SDDiP algorithm and different cut families to make this chapter self-contained. In Section 4.4, we present the mathematical formulation for MSUC. In Section 4.5, we describe various computational enhancements of SDDiP for solving MSUC. Sections 4.6-4.7 discuss experiment settings and detailed computational results. Finally, we provide some concluding

remarks in Section 4.8.

4.3 Stochastic Dual Dynamic Integer Programming

We start with a scenario tree formulation of an MSIP problem with binary state variables.

$$\begin{aligned} \min_{x_n, y_n, z_n} \quad & \sum_{n \in \mathcal{T}} p_n g_n(x_n, y_n) \\ \text{s.t.} \quad & \forall n \in \mathcal{T} \\ & (z_n, x_n, y_n) \in X_n && \text{(4.1a)} \\ & z_n = x_{a(n)}, z_n \in [0,1]^d && \text{(4.1b)} \\ & x_n \in \{0,1\}^d. && \text{(4.1c)} \end{aligned}$$

In the above formulation, $x_n \in \{0,1\}^d$ is the state variable of node n in the scenario tree $\mathcal{T}$ and y_n is the local variable. In addition, $z_n \in [0,1]^d$ is another continuous local variable which is a copy of the state variable from previous stage. Note that in this formulation only successive stages are linked together. This can be always be ensured by a proper reformulation of the problem. We assume that (4.1) has complete continuous recourse, i.e. given any value of the state and local integer variables, there exist a value of continuous local variables such that (4.1) is feasible. If the state variables are not binary, we can use binary expansion/approximation to transform them into binary. Suppose $x \in [0, U]$ is a continuous state variables, we substitute it by $\sum_{i=0}^{\kappa} 2^{i-1}\epsilon\lambda_i$, where $\lambda_i \in \{0,1\}$, $\kappa = \lfloor \log_2(U/\epsilon) \rfloor + 1$, and ϵ is the approximation accuracy. In this way $|x - \sum_{i=0}^{\kappa} 2^{i-1}\epsilon\lambda_i| \leq \epsilon$. For general integer variables, setting $\epsilon = 1$ results in an exact representation.

Now we can write down the DP equations for the optimal value function of the multistage problem (4.1) as follows:

$$(P_1) \quad \min \quad g_1(x_1, y_1) + \sum_{m \in \mathcal{C}(1)} q_{1m} Q_m(x_1)$$

$$\text{s.t.} \quad \text{(4.1a)} - \text{(4.1c)} \text{ (for } n = 1)$$

and for each node $n \in \mathcal{T} \setminus \{1\}$,

$$(P_n) \;\; Q_n(x_{a(n)}) := \min \;\; g_n(x_n, y_n) + \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(x_n)$$
$$\text{s.t. (4.1a)} - \text{(4.1c)},$$

where q_{nm} is the conditional probability from n to its children node m. We will refer to $Q_n(\cdot)$ as the optimal value function (of $x_{a(n)}$) at node n and denote the function $\mathcal{Q}_n(\cdot) := \sum_{m \in \mathcal{C}(n)} q_{nm} Q_m(\cdot)$ as the expected cost-to-go (ECTG) function at node n.

In the SDDiP algorithm, the scenario tree satisfies stage-wise independence, i.e., for any two nodes n and n' in stage t, the set of their children nodes $\mathcal{C}(n)$ and $\mathcal{C}(n')$ are defined by identical data and conditional probabilities. In this case, the ECTG functions depend only on the stage rather than the nodes, i.e., we have $\mathcal{Q}_n(\cdot) \equiv \mathcal{Q}_t(\cdot)$ for all n in stage t. More specifically, let N_t be the number of possible data realizations at stage $t = 2, \ldots, T$, each outcome has an equal probability of $1/N_t$. Then total number of scenarios is $N = \prod_{t=2}^{T} N_t$.

By exploiting the stage-wise independence of the underlying stochastic process, SDDiP algorithm proceeds stage-wise from $t = 1$ to T by solving a dynamic programming equation with an approximated ECTG function at each stage. The ECTG function at each stage is approximated from below by a convex piece-wise linear function, and these approximations are improved through iterations. In particular, the stage problem in the i-th iteration $P_t^i(x_{t-1}^i, \psi_t^i, \xi_t^k)$ is of the following form:

$$\underline{Q}_t^i(x_{t-1}^i, \psi_t^i, \xi_t^k) := \min_{x_t, y_t, z_t} \;\; g_t(x_t, y_t, \xi_t^k) + \psi_t^i(x_t) \tag{4.2a}$$
$$\text{s.t.} \;\; (x_t, y_t, z_t) \in X_t(\xi_t^k) \tag{4.2b}$$
$$z_t = x_{t-1}^i \tag{4.2c}$$

$$x_t \in \{0,1\}^d,\ z_t \in [0,1]^d, \tag{4.2d}$$

where $\psi_t^i(\cdot)$ is defined as:

$$\psi_t^i(x_t) := \min \big\{\theta_t : \theta_t \geq L_t, \tag{4.3a}$$

$$\theta_t \geq \frac{1}{N_{t+1}} \sum_{j=1}^{N_{t+1}} (v_{t+1}^{\ell j} + (\pi_{t+1}^{\ell j})^\top x_t), \quad \forall \ell = 1, \ldots, i-1\big\}. \tag{4.3b}$$

The function $\psi_t^i(x_t)$ is the current convex lower-approximation of the true ECTG function $\mathcal{Q}_t(x_t)$.

Given a solution x_{t-1}^i from the previous stage, the current approximation $\psi_t^i(\cdot)$, and a particular uncertainty realization $\xi_t^k (1 \leq k \leq N_t)$ at stage t, the forward problem in iteration i is fully characterized. We assume that a lower bound exists for the ECTG function to avoid unboundedness. Once a forward iteration is completed, we have obtained a feasible solution $\{(x_t^i, y_t^i)\}_{t=1}^T$ for the corresponding scenario.

The backward step starts from the last stage T. Given the solution x_{T-1}^i from iteration i and a particular uncertainty realization $\xi_T^j (1 \leq j \leq N_T)$, let R_T^{ij} be a relaxation of the forward problem $P_T^i(x_{T-1}^i, \psi_T^{i+1}, \xi_T^j)$. Note that $\psi_T^i \equiv 0$ for all $i \geq 0$. Solving R_T^{ij} for each j produces a linear inequality defined by (v_T^{ij}, π_T^{ij}) and it is valid for the value function $Q_T(x_{T-1}, \xi_T^j)$. Then the inequalities are aggregated to obtain one of form (4.3b), which is valid for ECTG function $\mathcal{Q}_{T-1}(x_{T-1})$. The lower approximation of the ECTG function is updated from $\psi_{T-1}^i(\cdot)$ to $\psi_{T-1}^{i+1}(\cdot)$. The backward step then proceeds to stage $T-1$. When the first stage is completed, since we have solved a lower approximation of the original problem, the optimal value of the first stage problem is an exact lower bound of the original problem.

One can also generate more than one scenario in the forward step. These single scenario cost can be used to compute a confidence interval of their mean value. The mean value of these costs is usually referred as a statistical upper bound for the original problem. In fact,

common stopping criteria for SDDP-type algorithm is often based on this statistical upper bound and the exact lower bound obtained from the backward step. A complete description of the SDDiP algorithm and the proof of almost surely convergence can be found in Section 3.6.

4.3.1 Cut Families in Backward Step

Depending on the relaxation problems R_t^{ij} solved in the backward step, different families of linear inequalities can be obtained. In this chapter, we implement the SDDiP algorithm with standard Benders' cuts, a type of Lagrangian cuts obtained from a particular reformulation, and strengthened Benders' cuts, which are a byproduct of the Lagrangian cuts.

Benders' cut (Benders 1962)

These cuts are derived from the LP relaxation of $P_t^i(x_{t-1}^i, \psi_t^{i+1}, \xi_t^j)$, and the cut coefficients (v_t^{ij}, π_t^{ij}) correspond to the optimal value of the LP relaxation and a basic feasible dual solution. To be precise, the cut added to ψ_{t-1}^i takes the form:

$$\theta_{t-1} \geq \frac{1}{N_t} \sum_{j=1}^{N_t} Q_{LP,t}^{ij} + \frac{1}{N_t} \sum_{j=1}^{N_t} (\pi_{LP,t}^{ij})^\top (x_{t-1} - x_{t-1}^i), \tag{4.4}$$

where $Q_{LP,t}^{ij}$ is the optimal LP relaxation objective function value of problem $P_t^i(x_{t-1}^i, \psi_t^{i+1}, \xi_t^j)$, and $\pi_{LP,t}^{ij}$ is a basic optimal dual solution in connection with the constraints $z_t = x_{t-1}^i$. Benders' cut are in general not tight when integer variables are present. Therefore almost sure convergence is not guaranteed if only these cuts are used.

Lagrangian cut

This family of cuts is based on solving a Lagrangian dual of $P_t^i(x_{t-1}^i, \psi_t^{i+1}, \xi_t^j)$. In particular, the relaxation problem R_t^{ij} is

$$(R_t^{ij}): \quad \max_{\pi_t} \left\{ \mathcal{L}_t^{ij}(\pi_t) + \pi_t^\top x_{t-1}^i \right\} \tag{4.5}$$

where

$$\begin{aligned} \mathcal{L}_t^{ij}(\pi_t) \;=\; & \min \quad g_t(x_t, y_t, z_t) + \psi_t^i - \pi_t^\top z_t \\ & \text{s.t.} \quad (x_t, y_t, \xi_t^j) \in X_t(\xi_t^j) \\ & \qquad x_t \in \{0,1\}^d,\ z_t \in [0,1]^d \end{aligned} \tag{4.6}$$

The cut coefficients (v_t^{ij}, π_t^{ij}) are then equal to $\mathcal{L}_t^{ij}(\pi_{LG,t}^{ij})$ and $\pi_{LG,t}^{ij}$, where $\pi_{LG,t}^{ij}$ is an optimal solution of the Lagrangian dual problem R_t^{ij}. It can be proven that the Lagrangian dual problem has zero duality gap and almost sure convergence is guaranteed if these cuts are used in the backward step (see Theorem 8).

Strengthened Benders' cut.

Instead of solving the Lagrangian dual problem to optimality, we can solve (4.6) with $\pi_t = \pi_{LP,t}^{ij}$. The cut then takes the form

$$\theta_{t-1} \geq \frac{1}{N_t} \sum_{j=1}^{N_t} \mathcal{L}_t^{ij}(\pi_{LP,t}^{ij}) + \frac{1}{N_t} \sum_{j=1}^{N_t} (\pi_{LP,t}^{ij})^\top (x_{t-1} - x_{t-1}^i).$$

The cut is valid because $\pi_{LP,t}^{ij}$ is feasible to (4.5). It is parallel to and at least as tight as standard Benders' cuts.

4.4 Multistage Stochastic UC

4.4.1 Problem Formulation

We present an extensive formulation for MSUC where the uncertain net load is modeled by a scenario tree. Below is a summary of notation.

Indices

n, m	Node in the scenario tree
i	Generation unit
b	Load bus
ℓ	Transmission line
t	Decision stage
$t(n)$	Decision stage of node n

Sets

$\mathcal{T}$	Scenario tree
$\mathcal{B}$	Set of all buses
$\mathcal{G}$	Set of generation units
$\mathcal{G}_b$	Set of generation units at bus b
$\mathcal{D}$	Set of demand bus
$\mathcal{L}$	Set of all transmission lines
$\mathcal{P}(n,t)$	Path from the t-th ancestor node of n to n

Parameters

p_n	Probability associated with node n
$\overline{S}_i$, $\underline{S}_i$	Start-up/shut-down cost of generation unit i
C_p	Penalty cost for unsatisfied demand and over generation
D_{nb}	Load demand of bus b at node n
F_l	Maximum flow capacity of line ℓ
K_l	Vector of the shift vectors of transmission line ℓ

$\overline{P}_i$, $\underline{P}_i$	Maximum/minimum output of generation unit i
R_t	Reserve requirement in period t
$\overline{\Delta}_i$, $\underline{\Delta}_i$	Regular ramp up/down rate for generation unit i
Δ^{SU}, Δ^{SD}	Start-up/shut-down ramp rate for generation unit i
UT_i, DT_i	Minimum up/down time for generation unit i

Decision variables

x_{ni}	State of generator i at node n, equals 1 if it is on, 0 otherwise
y_{ni}	Electricity produced by generation unit i at node n
u_{ni}	Start up decision for generation unit i, equals 1 if it is turned on at node n, 0 otherwise
v_{ni}	Shut down decision for generation unit i, equals 1 if it is turned off at node n, 0 otherwise
r_{ni}	Reserved spinning capacity from generation unit i at node n
δ_n^+	Unsatisfied demand across the network at node n
δ_n^-	Over-generation across the network at node n

A multistage stochastic programming formulation can be written as follows.

$$\min \quad \sum_{n\in\mathcal{T}} p_n \left[\sum_{i\in\mathcal{G}} \left(\overline{S}_i u_{ni} + \underline{S}_i v_{ni} + f_i(y_{ni})\right) + C_p(\delta_n^+ + \delta_n^-)\right] \tag{4.7a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{G}} y_{ni} + \delta_n^+ - \delta_n^- = \sum_{b\in\mathcal{D}} D_{nb}, \quad \forall n \in \mathcal{T} \tag{4.7b}$$

$$\sum_{i\in\mathcal{G}} r_{ni} \geq R_{t(n)}, \quad \forall n \in \mathcal{T} \tag{4.7c}$$

$$-F_\ell \leq \sum_{b\in\mathcal{B}} K_{\ell b} \left(\sum_{i\in\mathcal{G}_b} y_{ni} - D_{nb}\right) \leq F_\ell, \quad \forall n \in \mathcal{T}, \ell \in \mathcal{L} \tag{4.7d}$$

$$y_{ni} + r_{ni} \leq \overline{P}_i x_{ni}, \quad \forall n \in \mathcal{T}, i \in \mathcal{G} \tag{4.7e}$$

$$y_{ni} \geq \underline{P}_i x_{ni}, \quad \forall n \in \mathcal{T}, i \in \mathcal{G} \tag{4.7f}$$

$$y_{ni} - y_{a(n),i} \leq \Delta_i^{\text{SU}} u_{ni} + \overline{\Delta}_i x_{a(n),i}, \quad \forall n \in \mathcal{T}, i \in \mathcal{G} \tag{4.7g}$$

$$y_{a(n),i} - y_{ni} \leq \Delta_i^{\text{SD}} v_{ni} + \underline{\Delta}_i x_{ni}, \quad \forall n \in \mathcal{T}, i \in \mathcal{G} \tag{4.7h}$$

$$x_{ni} - x_{a(n),i} \leq u_{ni}, \quad \forall n \in \mathcal{T}, i \in \mathcal{G} \tag{4.7i}$$

$$x_{ni} - x_{a(n),i} = u_{ni} - v_{ni}, \quad \forall n \in \mathcal{T}, i \in \mathcal{G} \tag{4.7j}$$

$$\sum_{m \in \mathcal{P}(n, UT_i - 1)} u_{mi} \leq x_{ni}, \quad \forall n \in \mathcal{T}, i \in \mathcal{G} \tag{4.7k}$$

$$\sum_{m \in \mathcal{P}(n, DT_i - 1)} v_{mi} \leq 1 - x_{ni}, \quad \forall n \in \mathcal{T}, i \in \mathcal{G} \tag{4.7l}$$

$$x_{ni}, u_{ni}, v_{ni} \in \{0, 1\}, y_{ni}, r_{ni} \geq 0, \quad \forall n \in \mathcal{T}, i \in \mathcal{G}. \tag{4.7m}$$

In the above formulation, the objective function (4.7a) consists of the expected start-up cost, shut-down cost, generation cost (denoted by $f_i(y_{ni})$), and the possible penalty cost from electricity shortage or over generation. Network load balance constraint is enforced in (4.7b), and constraints (4.7c) indicates the requirement for total reserved capacity at each stage. Linearized power flow equations using shift factors are imposed by constraints (4.7d). Constraints (4.7e) and (4.7f) specify the output capacities for each generator. Ramping constraints are shown in (4.7g) and (4.7h). Constraints (4.7i) and (4.7j) link the generator states with commitment decisions. Minimum up and down time constraints are enforced by (4.7k) and (4.7l). These constraints are proposed by Rajan and Takriti (2005) in the two-stage setting and the authors show that such a formulation is a convex hull representation of the minimum up and down time polytope. Jiang et al. (2016) verify that such formulation is still tight under the stochastic setting. Lastly, (4.7m) contains the binary and non-negativity constraints for decision variables. At the time of solving this problem, any initial state of the system can be accommodated by constraints (4.7g)–(4.7l). In our numerical experiments, we assume that each generator is off and has met the minimum downtime requirement thus can be turned on immediately.

4.4.2 Stage-wise Independence in Net Load

The demand process and renewable output are usually correlated across different hours. However, they exhibit certain patterns throughout a day. We assume that a base net load profile (24-hour resolution) is given and the true net load deviates from this profile according to a given distribution. The deviations at each hour are assumed to be independent. Specifically, let $\{D_t\}_{t=1}^{T}$ be the nominal hourly net load profile. For each $t = 1, \ldots, T$, the true net load $\tilde{D}_t$ is generated as follows:

$$\tilde{D}_t = D_t \cdot \xi_t, \tag{4.8}$$

where $\xi_t \sim \Xi_t$, and Ξ_t is the inferred distribution from historical data. The total net load across the network is then allocated to each load bus according to the ratio implied from the base load profile, i.e., the proportion of the net load at each bus is the same for all realizations within the same stage. Table 4.1 shows the base net load profiles for two different systems.

Table 4.1: Net load base profile (Unit: MW)

#bus	12am	1am	2am	3am	4am	5am	6am	7am	8am	9am	10am	11am
14	259	243	215	148	184	223	259	287	303	326	329	311
118	4242	3987	3521	2418	3012	3648	4242	4709	4963	5345	5387	5090
#bus	12pm	1pm	2pm	3pm	4pm	5pm	6pm	7pm	8pm	9pm	10pm	11pm
14	295	282	326	334	313	329	347	363	370	334	321	303
118	4836	4624	5345	5472	5133	5387	5684	5939	6066	5472	5260	4963

4.4.3 State Variables

In formulation (4.7), decisions at node n are dependent on information passed from node n's ancestors. This information includes the generator state $x_{a(n)}$ of the last stage, the generation level $y_{a(n)}$ at the previous stage, and a sequence of commitment decisions from earlier stages,

i.e., $\{u_m, m \in \mathcal{P}(n, UT-1)\}$, and $\{v_m, m \in \mathcal{P}(n, DT-1)\}$. They are the state variables, and have a dimension of $\sum_{i\in\mathcal{G}}(UT_i + DT_i)$.

One obstacle of applying SDDiP directly to MSUC is the minimum up and down time constraints. They link variables from more than two stages, while SDDiP requires that the current stage problem depend only on the previous stage. To resolve this problem, we reformulate these constraints by making copies of decisions from nodes in $\mathcal{P}(n, UT_i - 1)$ and $\mathcal{P}(n, DT_i - 1)$. More specifically, we create two sets of new variables at each node n: $\{u_{ni}^{(k)}, k = 0, \ldots, UT_i - 1\}$ and $\{v_{ni}^{(k)}, k = 0, \ldots, DT_i - 1\}$. Constraints (4.7k)-(4.7l) are then equivalent to the following set of inequalities:

$$\sum_{k=0}^{UT_i-1} u_{ni}^{(k)} \leq x_{ni}, \quad \sum_{k=0}^{DT_i-1} v_{ni}^{(k)} \leq 1 - x_{ni}, \tag{4.9a}$$

$$u_{ni}^{(0)} - u_{ni} = 0, \quad v_{ni}^{(0)} - v_{ni} = 0, \tag{4.9b}$$

$$u_{ni}^{(k)} = u_{a(n),i}^{(k-1)}, \quad \forall k = 1, \ldots, UT_i - 1 \tag{4.9c}$$

$$v_{ni}^{(k)} = v_{a(n),i}^{(k-1)}, \quad \forall k = 1, \ldots, DT_i - 1 \tag{4.9d}$$

As a result, it is sufficient to pass $x_{a(n),i}$, $y_{a(n),i}$, $\{u_{a(n),i}^{(k)}\}_{k=0}^{UT_i-2}$, and $\{v_{a(n),i}^{(k)}\}_{k=0}^{DT_i-2}$ to node n, all of which comes from the parent node $a(n)$.

Moreover, SDDiP requires all state variables to be binary. In MSUC, generator states (x) and commitment decisions (u, v) are already binary, however, the dispatch decision (y) is continuous and requires a binary approximation.

With the above two modeling treatment and the stage-wise independence assumption, the MSUC problem (4.7) can be reformulated as a DP equation ready for SDDiP, and the state space has a dimension of $\sum_{i\in\mathcal{G}}(UT_i + DT_i + \lfloor \log_2(\overline{P}_i/\varepsilon) \rfloor)$.

4.5 SDDiP Enhancements

In this section, we describe several enhancements to the basic SDDiP method described previously.

4.5.1 Level Method for Lagrangian Cut

To obtain the cut coefficients of the Lagrangian cuts, one needs to solve the Lagrangian dual problem (4.5). This is a non-smooth convex optimization problem often solved by a subgradient method (see e.g.,Boyd et al. 2003). We propose to use the Level Method due to Lemaréchal et al. 1995. This method is similar to a cutting plane method that proceeds by considering an approximation or model of the objective function constructed from subgradients evaluated at proceeding iterates. The next iterate is obtained by projecting a minimizer of the model function to an appropriate level set so that its objective value lies in some neighborhood of the objective of the current iterate. In this way the iterates are regularized and the method achieves a theoretical optimal convergence rate. It has also been proven to be very effective in practice. In Section 4.7, we compare the performance of the Level Method with a basic subgradient algorithm for obtaining Lagrangian cut coefficients.

4.5.2 Hybrid Model using "Breakstage"

The quality of a policy obtained by SDDiP depends on the quality of the approximation of the ECTG in each stage. Intuitively, the stages further in the future has less influence on the current stage. Motivated by this, we propose a hybrid modeling approach, which allows us to improve the solution time while not compromising its quality to a large extent. This approximation relies on a prescribed stage t_b, which we will refer to as the *breakstage* hereafter. More specifically, in any decision stage before t_b, we solve the formulation $P_t^i(x_{t-1}^i, \psi_t^{i+1}, \xi_t^j)$, where state variables are binary. From stage t_b onward, we change the state variables back into their original space. It is still valid to add all three types of cuts at every stage, except that the Lagrangian cuts after t_b are not guaranteed to be tight.

In our experiments, we further relax all integrality constraints after t_b to improve solution time. As a result, only Benders' cuts are added for stage problems after t_b. If $t_b = 0$, the method reduces to SDDP applied to the LP relaxation of the original formulation; if $t_b = T + 1$, the fully discretized problem is solved by SDDiP.

The breakstage gives us the flexibility to evaluate the trade-off between solution time and solution quality. Solving the LP relaxation and using a state space of smaller dimension both contribute to the reduction of solution time. In addition, one can always adjust the policy if new information, e.g., a more accurate renewable output forecast, becomes available.

4.5.3 Backward Parallelization

In the backward step of SDDiP, multiple scenario problems are solved, then the cut coefficients returned by each of them are aggregated to produce a cut for its previous stage problem. Since these scenario problems are independent from each other, we implement a simple parallelization scheme using OpenMP to speed up the backward step.

4.6 Experimental Settings

In this section, we discuss our experimental settings. The 14-bus system has 5 generators, 20 transmission lines, and 11 demand buses; the 118-bus system includes 54 generators, 186 transmission lines, and 91 demand buses. Most data about the physical electrical network is from MatPower 6.0. Ramping limit is set to be 80% of the maximum generation capacity or specified otherwise. Minimum up and down times vary from 1 to 10 hours. To avoid infeasibility, slack variables are added to the load balance constraints and penalized with a large cost in the objective function. All penalty costs are assumed to be $5000 per MW.

4.6.1 Stage Problem Size

Deriving strengthened Benders' cuts and Lagrangian cuts require solving MIPs in each stage. Therefore, the size of the stage problem greatly affects the solution time. In the 14-bus system, the numbers of (binary) state variables, integer local variables, and continuous local variables are 127, 10, and 174, respectively. For the 118-bus system, the corresponding numbers are 1086, 108, and 1514.

4.6.2 Scenario Tree Generation

To generate a recombining scenario tree, we start with a given net load in the first stage (12am, $t = 1$). At each following hour (stage), realizations are independently generated according to (4.8). For the 14-bus system, we assume $\xi_t \sim \mathcal{U}(1-\alpha, 1+\alpha)$ for all $t > 1$, and $\alpha \in [0.1, 0.3]$. Six types of scenarios trees are generated, each of them is characterized by net load variation (α) and the number of outcomes at each stage (β). The corresponding tree is denoted by $\mathcal{T}_{14}^{\alpha,\beta}$. In our experiments, we consider $\alpha = 0.1, 0.2, 0.3$ and $\beta = 10, 20$.

For the 118-bus system, we use a truncated normal distribution, which is estimated based on data from California ISO website. We used the hourly net load forecast and the actual net load data across the entire California network in February 2017. The forecast is generated day ahead. For each hour, the distribution of forecast-to-actual ratio is approximated by a normal distribution. Some statistics of these ratios are summarized in Table 4.2.

Table 4.2: Statistics of forecast-to-actual ratio in net load

Hour	mean	std	min	max	Hour	mean	std	min	max
12am	1.01	0.02	0.97	1.04	12pm	1.04	0.07	0.94	1.20
1am	1.01	0.02	0.98	1.05	1pm	1.04	0.07	0.93	1.20
2am	1.02	0.02	0.97	1.05	2pm	1.06	0.07	0.94	1.22
3am	1.02	0.02	0.97	1.05	3pm	1.03	0.07	0.92	1.20
4am	1.02	0.02	0.97	1.05	4pm	1.02	0.06	0.91	1.15
5am	1.01	0.03	0.93	1.04	5pm	1.00	0.04	0.91	1.07
6am	1.00	0.03	0.91	1.06	6pm	1.00	0.03	0.93	1.05
7am	1.00	0.03	0.92	1.07	7pm	1.00	0.02	0.93	1.03
8am	1.03	0.05	0.95	1.16	8pm	0.99	0.02	0.94	1.04
9am	1.05	0.07	0.95	1.23	9pm	0.99	0.02	0.95	1.03
10am	1.05	0.07	0.94	1.23	10pm	0.99	0.02	0.96	1.04
11am	1.05	0.07	0.95	1.20	11pm	1.00	0.02	0.96	1.06

We assume $\xi_t \sim TN(\mu_t, k^2\sigma_t^2)$, where $TN(\mu_t, k^2\sigma_t^2)$ is the normal distribution $\mathcal{N}(\mu_t, k^2\sigma_t^2)$ truncated between $\mu_t \pm 3k\sigma_t$, and μ_t, σ_t are shown as in Table 4.2. The scenario tree, denoted by $\mathcal{T}_{118}^{k,\beta}$, is then characterized by k and the number of outcomes at

each stage (β). In our experiments, we fix $\beta = 20$ and consider k varying from 1.0 to 1.3. Figure 4.1 is an illustration of 50 independent scenarios from scenario tree $\mathcal{T}_{14}^{0.2,20}$(left) and $\mathcal{T}_{118}^{1.3,20}$ (right).

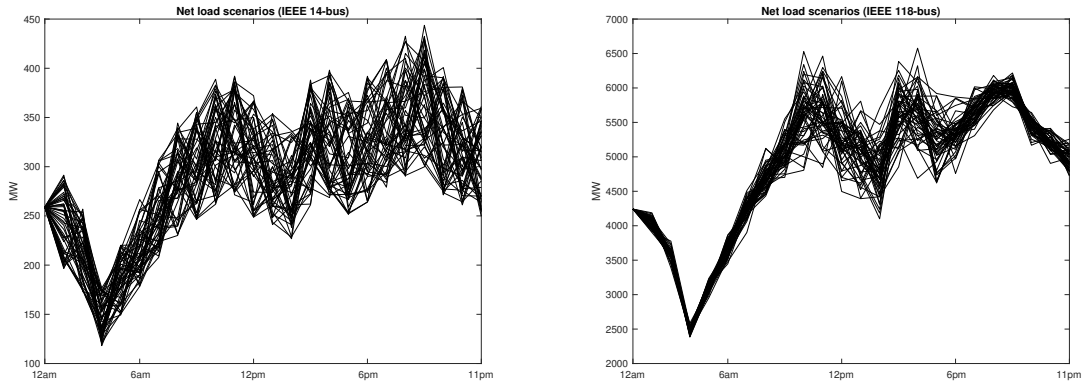


Figure 4.1: 50 scenarios from scenario tree $\mathcal{T}_{14}^{0.2,20}$(left) and $\mathcal{T}_{118}^{1.3,20}$ (right)

4.6.3 Other Implementation Details

In the forward step of the SDDiP algorithm, we generate candidate solutions for five independent sample paths, and in the backward step, we evaluate two of them which result in the highest cost. The Lagrangian dual problem is solved to optimality using a basic subgradient algorithm and the Level Method with an optimality tolerance of 10^{-4} for the 14-bus system and 5×10^{-4} for the 118-bus system, respectively. Other relative MIP tolerance is set to be the same as above for each system. The SDDiP algorithm is implemented in C++ with CPLEX 12.7.0 to solve the MIP and LP subproblems. All experiments are performed on a 16-core machine with Intel Xeon E5-2630 v3 @2.40GHz CPUs and 128GB of main memory. Reported solution times are wall clock times.

4.7 Computational Results

4.7.1 14-bus Results

We generate six different instances for the 14-bus system: $\mathcal{T}_{14}^{\alpha,\beta}$, $\alpha = 0.1, 0.2, 0.3$ and $\beta = 10, 20$. An instance with $\beta = 10$ involves, 10^{23} scenarios, and its extensive form has over 2.5×10^{24} variables, motivating the need for a sampling based decomposition method such as SDDiP. For each instance, the designed experiment consists of two phases: Phase I, Run SDDiP and obtain a policy; Phase II, Evaluate the policy with restored integrality constraints. For each (α, β) pair, we generate two scenario trees, the first one is used in Phase I to obtain policies, and the other is used for evaluation in Phase II.

In Phase I, we solve SDDiP with different breakstages (t_b). As mentioed earlier, when $t_b = 0$, SDDiP reduces to standard SDDP. If $t_b = 1$, nothing changes except that the first-stage problem becomes a MIP. When $t_b > 1$, other types of cuts may be added to the stage problems before $t_b - 1$. In particular, we consider five different cut combinations: Benders' cut only (B), strengthened Benders' cut only (SB), Lagrangian cut obtained by subgradient method (Sub), Lagrangian cut obtained by the Level Method (Level), and strengthened Benders' cut plus Lagrangian cut obtained by the Level Method (SB + Level).

Once t_b and cut families are determined, SDDiP starts. In the first half of iterations, we ignore any integrality constraints and only turn on Benders' cuts to get a rough estimation of the ECTG functions. In the second half, we restore these integrality constraints and add other types of cuts to improve the estimation. The final statistical upper bound is evaluated based on a set of 800 independent forward sample paths. SDDiP terminates after a fixed number of iterations.

In Phase II, we reinstate the integrality constraints in stage problems after t_b. A set of 800 scenarios is sampled independently from the second scenario tree, forward problems are solved with the policy obtained in Phase I, and the cost associated with each scenario is recorded. The performance of the policy is evaluated by comparing the lower bound returned

by SDDiP in Phase I, with the right endpoint of 95%-CI for the sample mean of scenario costs obtained from Phase II. All results in this section are averaged over 3 independent runs.

We discuss our findings with respect to following three aspects:

1. Which cut combination(s) perform the best in SDDiP?
2. What is the effect of different choices of breakstage?
3. What is the speed-up ratio and parallel efficiency from the backward parallelization?

Cut Combinations

To test the power of different families of cuts, we solve each instance with breakstage $t_b = 25$, i.e., the fully discretized problem. In the forward step, we solve MIPs to obtain binary candidate solutions, and in the backward step, different cuts are generated by evaluating these solutions. The power of each cut family is assessed based on the SDDiP gap, solution time, and final evaluation of corresponding policies. The number of iterations in SDDiP is fixed at 150 for instances with $\alpha = 0.1, 0.2$ and 500 for instances with $\alpha = 0.3$.

Figure 4.2 shows the SDDiP results of the six instances with different cut combinations. The figure on the left presents the gap between the lower bound obtained from the last backward step and statistical upper bound returned by the last forward step. The one on the right side contains the solution time of the SDDiP algorithm. The horizontal axis indicates the instances indexed by the (α, β) pair.

Clearly, `SB+Level` and `Level` yield the smallest gap with a reasonable solution time among all. When the net load variation is small, using any type of these cuts is sufficient to solve the problem. When the variation becomes bigger, however, at least one family of tight cuts is in need to close the gap. Strengthened Benders' cut slightly improves the SDDiP gap of only using Benders' cut. Even though Lagrangian cuts and strengthened Benders' cuts are not dominated by each other, there is a significant improvement in SDDiP gap when Lagrangian cuts are used. In addition, it is evident that the Level Method is better

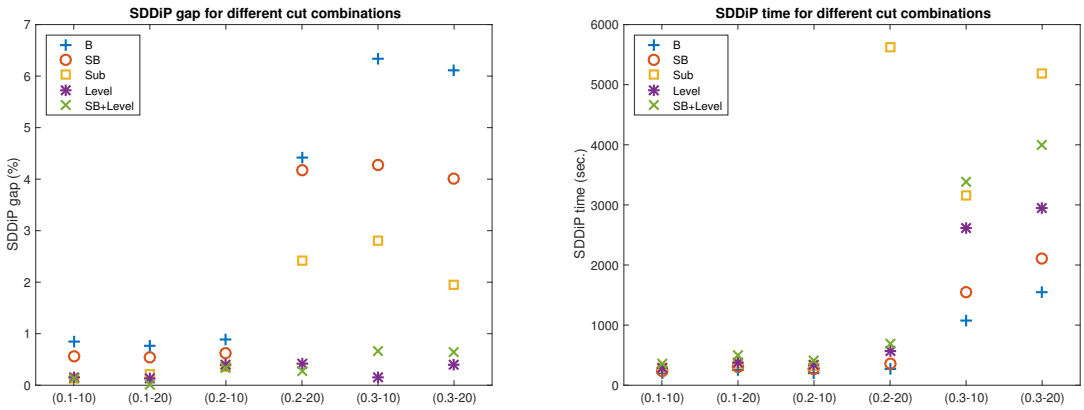


Figure 4.2: SDDiP results with different cut combinations

Horizontal axis indicates instance label (α, β), where α represents the demand variation ($\mathcal{U}(1-\alpha, 1+\alpha)$), β represents the number of branches in the scenario tree. SDDiP gap and time are evaluated upon termination: 150 iterations for Instance 1–4, and 500 for Instance 5 and 6.

than standard subgradient method. It takes less time to reach a much smaller gap, and the solution time is also more stable.

The Phase II evaluation results are summarized in Figure 4.3. `SB+Level` and `Level`

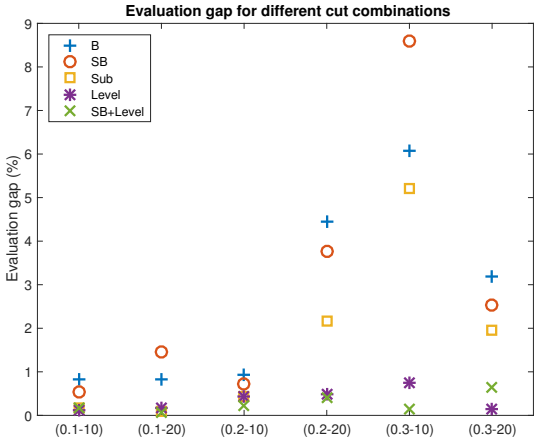


Figure 4.3: Evaluation of policies obtained by different cut combinations

The gap is between LB from SDDiP and statistical UB from policy evaluation on 800 sample paths from the second scenario tree.

produce the most stable policies and yield the tightest statistic upper bound estimation. The policy approximated by Lagrangian cuts using subgradient method is again shown to be inferior to the one with the Level Method. In addition, we can observe a large evaluation

gap for the policy characterized by the strengthened Benders' cut in the instance $(0.3, 10)$. A possible reason is that 10 realizations per stage is not enough to represent the uncertainty with such big variation, the scenario tree used in the evaluation phase has some extreme scenarios that was not assessed in Phase I.

In summary, `SB+Level` or `Level` is the best cut combination for SDDiP, and solving the Lagrangian dual problem using the Level Method is more efficient and stable. Detailed results can be found in Table A1 - A2 in Appendix.

Effect of Breakstage

We next study the hybrid modeling approach proposed in Section 4.5.2. In particular, we choose 6 values for t_b, ranging from 0 to 25. When $t_b = 0$, the standard SDDP algorithm is used to solve the LP relaxation of the original problem. When $t_b > 1$, both strengthened Benders' cuts and Lagrangian cuts (using the Level Method) are used in the backward step for stage problems before t_b. The number of iterations in SDDiP is fixed at 150 for instances with $\alpha = 0.1, 0.2$ and 500 for instances with $\alpha = 0.3$.

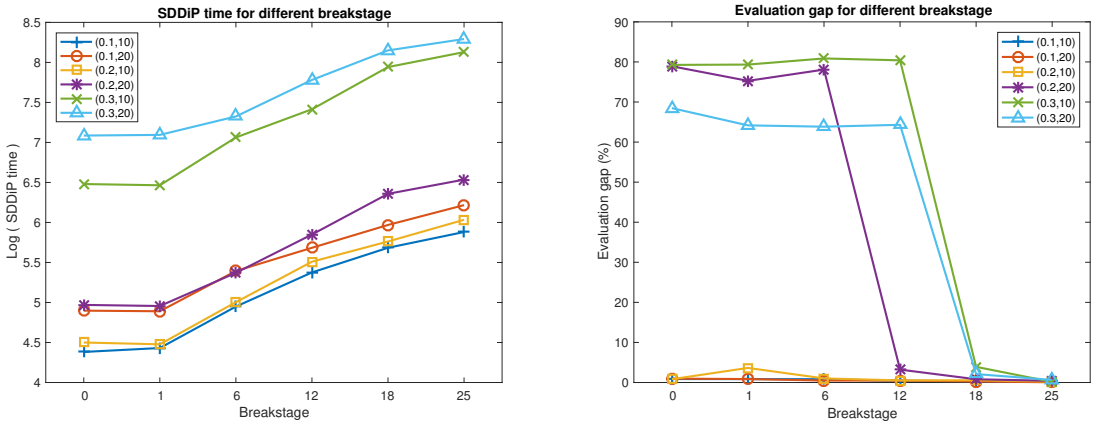


Figure 4.4: Effect of breakstage

SDDiP gap and time are evaluated upon termination: 150 iterations for Inst. 1–4, and 500 for Inst. 5 and 6.

Figure 4.4 summarizes the effect of different breakstage values. The solution time for the SDDiP algorithm increases as the breakstage increases. This is simply because more MIPs are solved as t_b increases. On average, when the breakstage increases from 0 (LP case)

to 25 (fully discretized problem), the solution time increases by a factor of 4. SDDiP gap is not reported here since the algorithm terminates with a gap smaller than 0.6% for all these instances. The right figure in Figure 4.4 summarizes the evaluation results. The evaluation gap tends to decrease as breakstage increases. For instances with smaller net load variation, a policy obtained by solving an approximation model with small breakstage is sufficiently good, i.e., evaluation gap is small. When the uncertainty variation is high (e.g., $\alpha = 0.3$), such a policy results in too much penalty. Therefore, solving an approximation model to optimality does not necessarily imply that SDDiP has produced a good policy, sometimes the effort of recovering the true ECTG function at each stage is necessary. Refer to Table A3 - A4 in Appendix for more detailed computational results.

Backward Parallelization

Let $T(k)$ be the solution time when k threads are used. We define *speed-up ratio* by $\frac{T(1)}{T(k)}$, and *efficiency* by $k\frac{T(1)}{T(k)}$. Figure 4.5 depicts an average speed-up ratio and efficiency graph with respect to the number of threads for a particular instance. We use 32 threads in all of our computation experiments. On average, the maximum speed-up ratio is 4.8 with an efficiency of 15%.

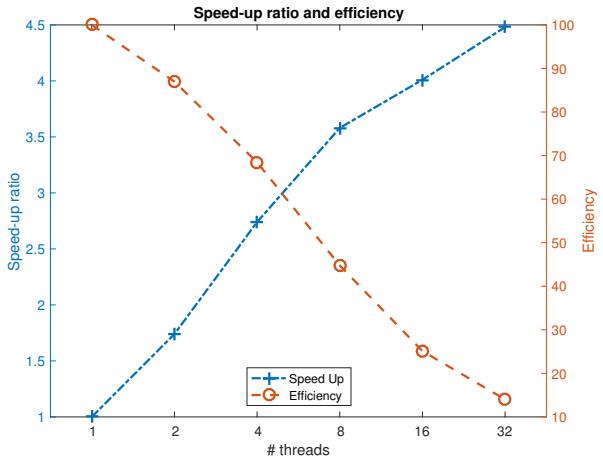


Figure 4.5: Parallelization speed-up ratio & efficiency ($\mathcal{T}_{14}^{0.2,20}$ instance)

4.7.2 118-bus Results

Similar to the 14-bus system, the experiments for the 118-bus system also consists of two phases: SDDiP and policy evaluation. We fix $\beta = 20$ in each scenario tree tested. An instance involves, 2×10^{24} scenarios, and its extensive form has over 5.4×10^{26} variables. Each instance is indexed by a pair (r, k), where r is ramping ratio with respect to the output capacity, and k is the parameter in the truncated normal distribution. We consider twelve instances with $r = 0.9, 0.8, 0.7$ and $k = 1.0, 1.1, 1.2, 1.3$[1]. A smaller r indicates more restricted ramping constraints, while a larger k value suggests a more volatile scenario tree.

We combine SDDiP with sample average approximation (SAA) to evaluate the quality of returned policy. For each pair of (r, k), we generate six scenario trees independently. We solve SDDiP on the first five trees (the algorithm is terminated after 500 iterations), and test each returned policies on the sixth tree. SDDiP results are taken as average over the five runs on the first five trees, and the final evaluation results are the 95%-CIs calculated based on the five policy assessments on the sixth tree.

Table 4.3 contains the SDDiP computation time and evaluation results for the 118-bus system. The results indicate using SDDiP with Benders' cut only is sufficient to produce an accurate and reliable policy for all 12 instances. This could be due to the tight formulation of

Table 4.3: Computational results for 118-bus system

Instance (r, k)	Time (sec.)	Eval. Gap (%)	Instance (r, k)	Time (sec.)	Eval. Gap (%)
(0.9, 1.0)	4389	[0.47, 0.68]	(0.8, 1.2)	4424	[0.51, 0.75]
(0.9, 1.1)	4387	[0.51, 0.59]	(0.8, 1.3)	4455	[0.55, 0.96]
(0.9, 1.2)	4394	[0.50, 0.77]	(0.7, 1.0)	4389	[0.37, 0.63]
(0.9, 1.3)	4405	[0.55, 0.69]	(0.7, 1.1)	4427	[0.58, 0.84]
(0.8, 1.0)	4333	[0.48, 0.63]	(0.7, 1.2)	4455	[0.50, 1.12]
(0.8, 1.1)	4362	[0.48, 0.58]	(0.7, 1.3)	4521	[0.67, 1.28]

[1]We do not consider value bigger than 1.3 because a larger value incurs a net load which exceeds the system's total generation capacity.

a single scenario deterministic UC problem. To verify the tightness of the LP relaxation gap, we independently generate 100 scenarios from the most volatile load distribution ($k = 1.3$), and solve a deterministic 24-hour UC problem and its LP relaxation for each of the scenarios. The ramping limit is set to be 70% of the maximum generation capacity. Indeed, the average LP gap over these 100 instances is only 0.254%. Given that our uncertainty variation is based on real data, such a small LP relaxation gap suggests that the SDDiP with standard Benders' cut is good enough to solve this large-scale MSUC instance.

4.8 Concluding Remarks

In this chapter, we propose a stagewise-decomposition algorithm based on SDDiP with various algorithmic enhancements to solve the MSUC problem. Extensive numerical experiments demonstrate that the proposed algorithm can successfully handle MSUC problems with a huge number of scenarios that were impossible before. It is also verified that Lagrangian cuts are indispensable in achieving exact solution and convergence. Our experiments show that when solving the Lagrangian relaxation of the stage problem, the Level Method performs superior to the standard subgradient method. We also observe that for the 118-bus system, it suffices to use SDDiP with only standard Benders' cuts to obtain a good policy.

There are several interesting future research questions related to MSUC. In this chapter, we decompose the 24-hour MSUC problem on an hourly base. An alternative is to consolidate several consecutive hours into one stage. Such a formulation increases the size of a stage problem but reduces the total number of decision stages. It would be interesting to investigate how such an aggregated model performs compares to the hourly based multistage model. Another direction is to study the MSUC problem under a risk-averse setting, as system reliability is of the utmost importance in practice.

Appendices

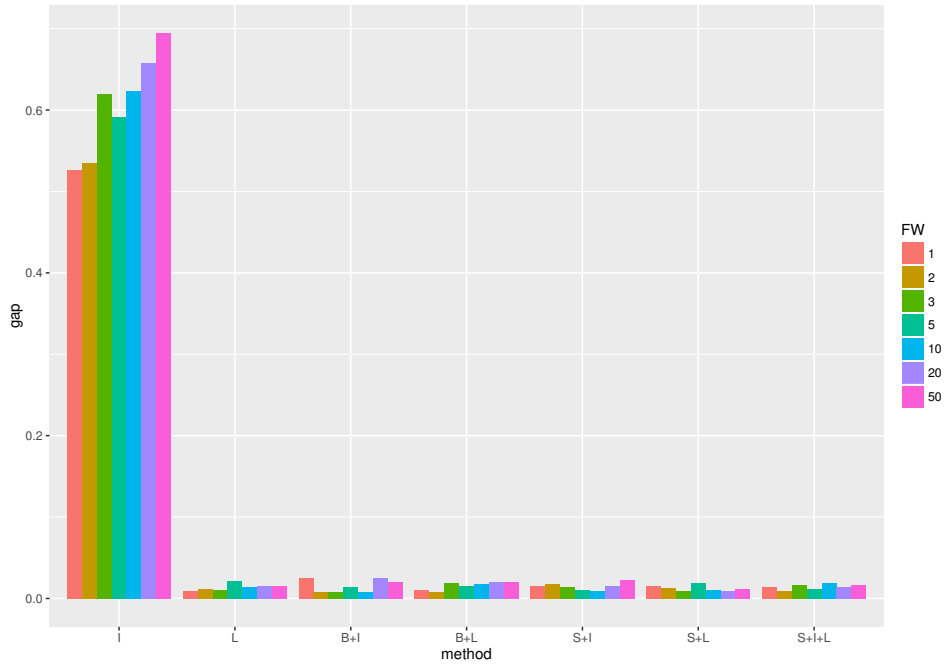


Figure A1: SDDiP gap for 10-year GEP instances

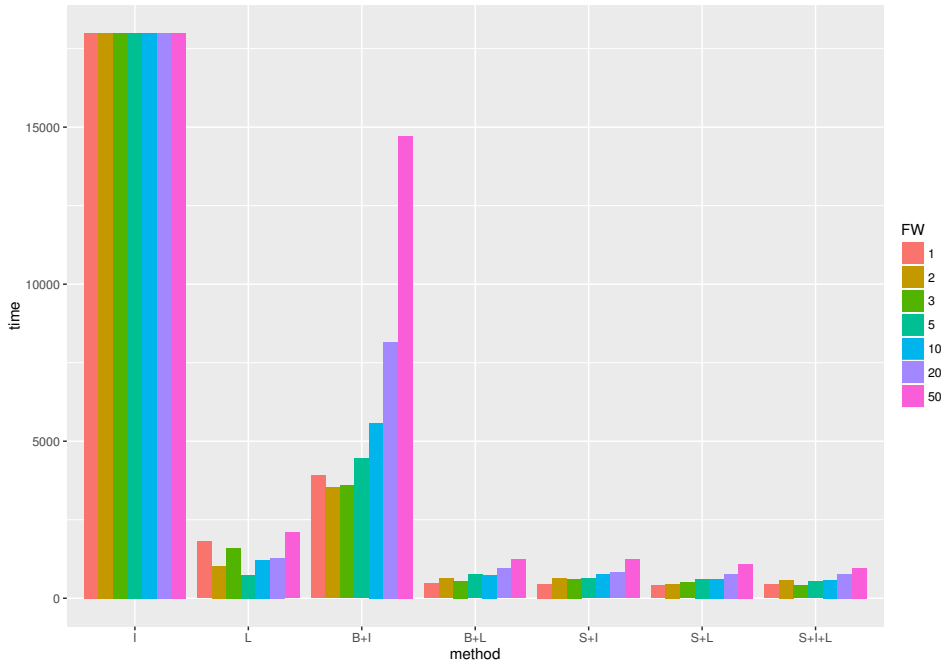


Figure A2: SDDiP computation time for 10-year GEP instances

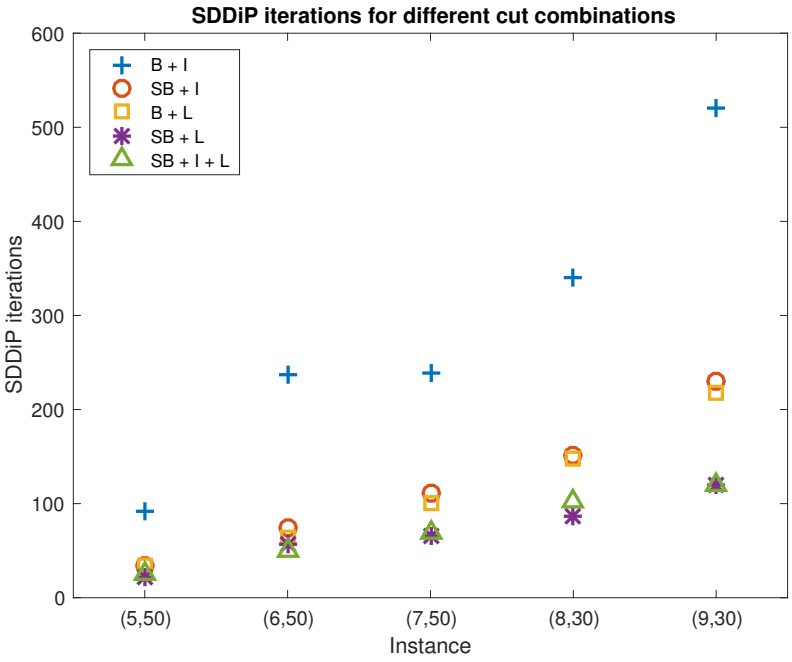


Figure A3: SDDiP iterations for large GEP instances

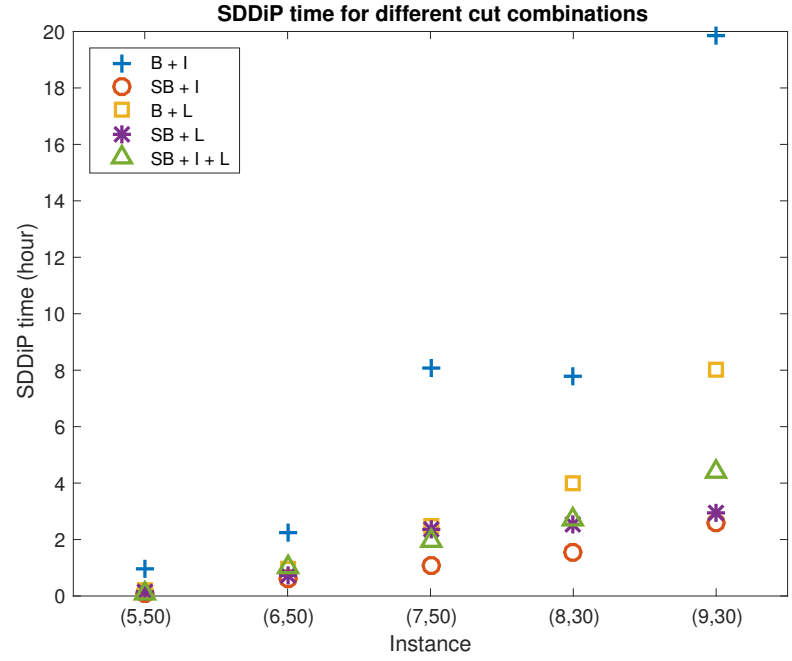


Figure A4: SDDiP computation time for large GEP instances

Table A1: Numerical results of IEEE 14-bus system – cut combinations

Instance		Phase I: SDDiP				Phase II: Evaluation				
(α, β)	Cut	LB (k$)	95%-CI UB (k$)	Gap (%)	Time (sec.)	95%-CI UB (k$)	Gap (%)	Min (k$)	Max (k$)	Std (k$)
(0.1, 10)	B	88.66	[89.25 , 89.41]	0.85	200	[89.24 , 89.40]	0.83	85.85	92.93	1.17
	SB	88.96	[89.31 , 89.47]	0.57	243	[89.29 , 89.45]	0.55	85.71	93.16	1.19
	Sub	89.30	[89.25 , 89.42]	0.13	296	[89.29 , 89.46]	0.17	85.50	92.88	1.21
	Level	89.30	[89.28 , 89.45]	0.16	282	[89.24 , 89.41]	0.12	85.42	93.10	1.20
	SB + Level	89.34	[89.28 , 89.45]	0.13	358	[89.32 , 89.48]	0.16	85.91	92.89	1.20
(0.1, 20)	B	88.55	[89.05 , 89.23]	0.76	248	[89.13 , 89.29]	0.83	85.33	92.93	1.17
	SB	88.83	[89.14 , 89.31]	0.54	314	[89.02 , 90.14]	1.46	85.62	314.79	8.08
	Sub	89.18	[89.21 , 89.37]	0.22	360	[89.10 , 89.27]	0.10	85.71	93.14	1.20
	Level	89.14	[89.08 , 89.26]	0.13	372	[89.12 , 89.29]	0.17	85.53	92.39	1.22
	SB + Level	89.21	[89.04 , 89.21]	0.00	501	[89.10 , 89.26]	0.06	85.33	92.47	1.17
(0.2, 10)	B	89.05	[89.52 , 89.85]	0.89	206	[89.55 , 89.88]	0.92	82.04	98.78	2.44
	SB	89.18	[89.41 , 89.74]	0.62	263	[89.51 , 89.83]	0.73	82.46	97.48	2.34
	Sub	89.61	[89.60 , 89.93]	0.36	322	[89.65 , 89.99]	0.42	82.97	97.11	2.46
	Level	89.57	[89.58 , 89.93]	0.40	340	[89.63 , 89.96]	0.43	82.62	96.17	2.42
	SB + Level	89.59	[89.54 , 89.89]	0.33	417	[89.45 , 89.78]	0.22	81.27	97.46	2.43

Table A2: Numerical results of IEEE 14-bus system – cut combinations (cont'd)

Instance		Phase I: SDDiP				Phase II: Evaluation				
(α, β)	Cut	LB (k$)	95%-CI UB (k$)	Gap (%)	Time (sec.)	95%-CI UB (k$)	Gap (%)	Min (k$)	Max (k$)	Std (k$)
(0.2, 20)	B	89.80	[92.86 , 93.94]	4.41	268	[93.03 , 93.99]	4.45	83.61	187.07	6.89
	SB	89.96	[92.88 , 93.89]	4.18	350	[92.43 , 93.48]	3.76	81.39	185.01	7.54
	Sub	90.63	[92.26 , 92.87]	2.42	5619	[92.03 , 92.62]	2.16	83.61	105.80	4.29
	Level	92.35	[92.18 , 92.73]	0.41	573	[92.24 , 92.81]	0.49	78.86	106.22	4.05
	SB + Level	92.42	[92.12 , 92.68]	0.27	689	[92.26 , 92.80]	0.41	82.82	106.82	3.91
(0.3, 10)	B	87.17	[91.30 , 93.06]	6.34	1083	[91.05 , 92.80]	6.07	78.40	382.15	12.68
	SB	87.49	[90.41 , 91.39]	4.27	1543	[89.93 , 95.71]	8.59	76.09	1222.05	41.73
	Sub	90.57	[90.58 , 93.22]	2.84	3158	[90.93 , 95.54]	5.20	77.94	900.15	33.21
	Level	90.65	[90.04 , 90.79]	0.15	2614	[90.56 , 91.33]	0.75	75.35	110.28	5.58
	SB + Level	90.70	[90.55 , 91.31]	0.66	3390	[90.05 , 90.83]	0.14	76.68	110.32	5.60
(0.3, 20)	B	92.06	[93.50 , 98.06]	6.12	1540	[93.96 , 95.09]	3.19	81.10	201.46	8.16
	SB	92.23	[94.38 , 96.07]	4.00	2109	[93.72 , 94.61]	2.52	78.22	203.20	6.42
	Sub	92.95	[93.77 , 94.79]	1.95	5182	[93.91 , 94.80]	1.95	79.48	200.80	6.39
	Level	93.94	[93.64 , 94.31]	0.39	2955	[93.45 , 94.07]	0.14	81.34	115.17	4.52
	SB + Level	93.95	[93.86 , 94.56]	0.64	3992	[93.87 , 94.56]	0.64	80.75	116.15	4.97

Table A3: Numerical results of IEEE 14-bus system – effect of breakstage

Instance		Phase I: SDDiP				Phase II: Evaluation				
(α, β)	t_b	LB (k$)	95%-CI UB (k$)	Gap (%)	Time (sec.)	95%-CI UB (k$)	Gap (%)	Min (k$)	Max (k$)	Std (k$)
(0.1, 10)	-1	88.66	[88.56 , 88.73]	0.08	80	[89.27 , 89.43]	0.87	86.00	93.33	1.17
	0	88.67	[88.61 , 88.78]	0.12	84	[89.27 , 89.44]	0.85	85.37	93.67	1.20
	6	88.91	[89.04 , 89.21]	0.34	141	[89.46 , 89.64]	0.82	85.45	94.17	1.30
	12	89.07	[89.17 , 89.33]	0.30	216	[89.28 , 89.44]	0.42	86.17	92.87	1.20
	18	89.10	[89.03 , 89.20]	0.11	294	[89.23 , 89.39]	0.33	86.14	93.65	1.19
	23	89.34	[89.28 , 89.45]	0.13	358	[89.32 , 89.48]	0.16	85.91	92.89	1.20
(0.1, 20)	-1	88.55	[88.41 , 88.58]	0.03	134	[89.25 , 89.42]	0.97	85.18	92.98	1.22
	0	88.57	[88.55 , 88.72]	0.17	133	[89.17 , 89.34]	0.86	85.81	93.64	1.23
	6	88.85	[88.87 , 89.03]	0.21	220	[89.11 , 89.28]	0.49	85.26	92.77	1.18
	12	89.00	[88.97 , 89.13]	0.15	294	[89.24 , 89.42]	0.48	85.35	93.63	1.33
	18	89.02	[89.01 , 89.16]	0.16	391	[89.09 , 89.26]	0.26	85.32	92.78	1.19
	23	89.21	[89.04 , 89.21]	0.00	501	[89.10 , 89.26]	0.06	85.33	92.47	1.17
(0.2, 10)	-1	89.06	[88.91 , 89.26]	0.23	90	[89.47 , 89.81]	0.83	80.69	97.96	2.43
	0	89.07	[88.98 , 89.31]	0.27	88	[88.69 , 92.44]	3.64	82.71	851.31	27.04
	6	89.15	[89.00 , 89.32]	0.19	149	[89.56 , 90.07]	1.02	83.01	169.30	3.66
	12	89.32	[89.26 , 89.59]	0.31	247	[89.43 , 89.78]	0.52	83.07	96.98	2.55
	18	89.32	[89.08 , 89.40]	0.09	318	[89.47 , 89.82]	0.56	82.43	98.47	2.53
	23	89.59	[89.54 , 89.89]	0.33	417	[89.45 , 89.78]	0.22	81.27	97.46	2.43

Table A4: Numerical results of IEEE 14-bus system – effect of breakstage (cont'd)

Instance		Phase I: SDDiP				Phase II: Evaluation				
(α, β)	t_b	LB (k$)	95%-CI UB (k$)	Gap (%)	Time (sec.)	95%-CI UB (k$)	Gap (%)	Min (k$)	Max (k$)	Std (k$)
(0.2, 20)	-1	89.81	[89.81 , 90.15]	0.38	144	[354.53 , 425.95]	78.92	83.01	2560.00	515.36
	0	89.83	[89.72 , 90.06]	0.26	142	[298.23 , 363.42]	75.28	81.67	2920.00	470.34
	6	89.88	[89.94 , 90.27]	0.43	215	[338.67 , 410.28]	78.09	84.57	2410.00	516.69
	12	90.08	[89.96 , 90.28]	0.22	347	[92.44 , 93.08]	3.22	83.37	109.11	4.62
	18	91.98	[91.97 , 92.72]	0.80	577	[92.20 , 92.73]	0.81	84.46	105.12	3.85
	23	92.42	[92.12 , 92.68]	0.27	689	[92.26 , 92.80]	0.41	82.82	106.82	3.91
(0.3, 10)	-1	87.16	[86.94 , 87.49]	0.37	652	[356.42 , 420.32]	79.26	77.14	2520.23	461.05
	0	87.20	[87.18 , 87.74]	0.62	642	[354.78 , 422.00]	79.34	76.59	2413.33	484.99
	6	87.52	[87.40 , 87.97]	0.51	1167	[390.43 , 457.58]	80.87	75.53	2299.30	484.49
	12	87.72	[87.34 , 87.89]	0.19	1660	[379.85 , 447.66]	80.40	76.07	2401.98	489.28
	18	90.08	[90.23 , 90.87]	0.87	2812	[90.47 , 93.74]	3.91	77.09	556.37	23.60
	23	90.70	[90.55 , 91.31]	0.66	3390	[90.05 , 90.83]	0.14	76.68	110.32	5.60
(0.3, 20)	-1	92.05	[91.65 , 92.20]	0.16	1195	[231.09 , 291.93]	68.47	82.99	2532.54	438.99
	0	92.08	[92.10 , 92.62]	0.57	1205	[202.24 , 256.99]	64.17	83.43	2579.31	394.98
	6	92.18	[92.11 , 92.66]	0.52	1517	[202.96 , 254.90]	63.84	79.09	2271.58	374.76
	12	92.43	[92.07 , 92.60]	0.18	2394	[203.26 , 258.95]	64.31	79.13	2358.78	401.87
	18	93.47	[93.19 , 93.97]	0.53	3463	[93.72 , 95.48]	2.10	81.01	426.44	12.68
	23	93.95	[93.86 , 94.56]	0.64	3992	[93.87 , 94.56]	0.64	80.75	116.15	4.97

REFERENCES

Abgottspon, H. et al. “Risk-averse medium-term hydro optimization considering provision of spinning reserves”. In: *Probabilistic Methods Applied to Power Systems (PMAPS), 2014 International Conference on*. IEEE. 2014, pp. 1–6.

Ahmed, S. “Two-Stage Stochastic Integer Programming: A Brief Introduction”. In: *Wiley Encyclopedia of Operations Research and Management Science*. Ed. by J. J. Cochran et al. John Wiley & Sons, Inc., 2010.

Ahmed, S. and N. V. Sahinidis. “An approximation scheme for stochastic integer programs arising in capacity expansion”. In: *Operations Research* 51.3 (2003), pp. 461–471.

Ahmed, S., A. J. King, and G. Parija. “A multi-stage stochastic integer programming approach for capacity expansion under uncertainty”. In: *Journal of Global Optimization* 26.1 (2003), pp. 3–24.

Akbari, T., A. Rahimikian, and A. Kazemi. “A multi-stage stochastic transmission expansion planning method”. In: *Energy Conversion and Management* 52.8 (2011), pp. 2844–2853.

Alonso-Ayuso, A., 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”. In: *European Journal of Operational Research* 151.3 (2003), pp. 503–519.

Angulo, G., S. Ahmed, and S. S. Dey. “Improving the integer L-shaped method”. In: *INFORMS Journal on Computing* (2016). To appear.

Archibald, T. et al. “Nested Benders decomposition and dynamic programming for reservoir optimisation”. In: *Journal of the Operational Research Society* 50.5 (1999), pp. 468–479.

Bacaud, L. et al. “Bundle methods in stochastic optimal power management: A disaggregated approach using preconditioners”. In: *Computational Optimization and Applications* 20.3 (2001), pp. 227–244.

Baringo, L. and A. J. Conejo. “Risk-constrained multi-stage wind power investment”. In: *Power Systems, IEEE Transactions on* 28.1 (2013), pp. 401–411.

Bean, J. C., J. L. Higle, and R. L. Smith. “Capacity expansion under stochastic demands”. In: *Operations Research* 40.3-supplement-2 (1992), S210–S216.

Benders, J. F. “Partitioning procedures for solving mixed-variables programming problems”. In: *Numerische mathematik* 4.1 (1962), pp. 238–252.

Berman, O., Z. Ganz, and J. M. Wagner. “A stochastic optimization model for planning capacity expansion in a service industry under uncertain demand”. In: *Naval Research Logistics (NRL)* 41.4 (1994), pp. 545–564.

Bertsekas, D. P. *Nonlinear programming*. Athena scientific Belmont, 1999.

Bienstock, D. and J. F. Shapiro. “Optimizing resource acquisition decisions by stochastic programming”. In: *Management Science* 34.2 (1988), pp. 215–229.

Birge, J. R. “Decomposition and partitioning methods for multistage stochastic linear programs”. In: *Operations Research* 33.5 (1985), pp. 989–1007.

Birge, J. R. and F. Louveaux. *Introduction to Stochastic programming*. 2nd ed. Springer, 2011.

Bloom, J. A. “Solving an electricity generating capacity expansion planning problem by generalized Benders’ decomposition”. In: *Operations Research* 31.1 (1983), pp. 84–100.

Bloom, J. A., M. Caramanis, and L. Charny. “Long-range generation planning using generalized benders’ decomposition: Implementation and experience”. In: *Operations Research* 32.2 (1984), pp. 290–313.

Boer, S. V. de, R. Freling, and N. Piersma. “Mathematical programming for network revenue management revisited”. In: *European Journal of Operational Research* 137.1 (2002), pp. 72–92.

Boyd, S., L. Xiao, and A. Mutapcic. “Subgradient methods”. In: *lecture notes of EE392o, Stanford University, Autumn Quarter* 2004 (2003).

Bruno, S. et al. “Risk neutral and risk averse approaches to multistage renewable investment planning under uncertainty”. In: *European Journal of Operational Research* 250.3 (2016), pp. 979–989.

Carøe, C. C. and R. Schultz. “Dual decomposition in stochastic integer programming”. In: *Operations Research Letters* 24.1 (1999), pp. 37–45.

Carpentier, P. et al. “Stochastic optimization of unit commitment: a new decomposition framework”. In: *IEEE Transactions on Power Systems* 11.2 (1996), pp. 1067–1073.

Cerisola, S. et al. “Stochastic power generation unit commitment in electricity markets: A novel formulation and a comparison of solution methods”. In: *Operations Research* 57.1 (2009), pp. 32–46.

Cerisola, S. et al. “Stochastic power generation unit commitment in electricity markets: A novel formulation and a comparison of solution methods”. In: *Operations Research* 57.1 (2009), pp. 32–46.

Cerisola, S., J. M. Latorre, and A. Ramos. “Stochastic dual dynamic programming applied to nonconvex hydrothermal models”. In: *European Journal of Operational Research* 218.3 (2012), pp. 687–697.

Chen, L. and T. Homem-de Mello. “Re-solving stochastic programming models for airline revenue management”. In: *Annals of Operations Research* 177.1 (2010), pp. 91–114.

Chen, Z.-L. and W. B. Powell. “Convergent cutting-plane and partial-sampling algorithm for multistage stochastic linear programs with recourse”. In: *Journal of Optimization Theory and Applications* 102.3 (1999), pp. 497–524.

Chen, Z.-L., S. Li, and D. Tirupati. “A scenario-based stochastic programming approach for technology and capacity planning”. In: *Computers & Operations Research* 29.7 (2002), pp. 781–806.

Dantzig, G. B. and G. Infanger. “Multi-stage stochastic linear programs for portfolio optimization”. In: *Annals of Operations Research* 45.1 (1993), pp. 59–76.

Davis, M. H. A. et al. “Optimal capacity expansion under uncertainty”. In: *Advances in Applied Probability* (1987), pp. 156–176.

Dempster, M. et al. “Planning logistics operations in the oil industry”. In: *Journal of the Operational Research Society* (2000), pp. 1271–1288.

Dupačová, J., N. Gröwe-Kuska, and W. Römisch. “Scenario reduction in stochastic programming”. In: *Mathematical programming* 95.3 (2003), pp. 493–511.

Dupačová, J., M. Bertocchi, and V. Moriggia. “Testing the structure of multistage stochastic programs”. In: *Computational Management Science* 6.2 (2009), pp. 161–185.

Erlenkotter, D. “Optimal plant size with time-phased imports”. In: *Investments for Capacity Expansion: Size, Location, and Time-Phasing* 5 (1967), p. 157.

Flach, B., L. Barroso, and M. Pereira. “Long-term optimal allocation of hydro generation for a price-maker company in a competitive market: latest developments and a stochastic dual dynamic programming approach”. In: *IET generation, transmission & distribution* 4.2 (2010), pp. 299–314.

Fleten, S.-E. and T. K. Kristoffersen. “Short-term hydropower production planning by stochastic programming”. In: *Computers & Operations Research* 35.8 (2008), pp. 2656–2671.

Freidenfelds, J. “Capacity expansion when demand is a birth-death random process”. In: *Operations Research* 28.3-part-ii (1980), pp. 712–721.

Gassmann, H. I. “MSLiP: A computer code for the multistage stochastic linear programming problem”. In: *Mathematical Programming* 47.1-3 (1990), pp. 407–423.

Giglio, R. J. “Stochastic capacity models”. In: *Management Science* 17.3 (1970), pp. 174–184.

Girardeau, P, V Leclere, and A. Philpott. “On the convergence of decomposition methods for multistage stochastic convex programs”. In: *Mathematics of Operations Research* 40.1 (2014), pp. 130–145.

Gjelsvik, A., M. M. Belsnes, and A. Haugstad. “An algorithm for stochastic medium-term hydrothermal scheduling under spot price uncertainty”. In: *Proceedings of 13th Power Systems Computation Conference*. 1999.

Glover, F. “Improved linear integer programming formulations of nonlinear integer problems”. In: *Management Science* 22.4 (1975), pp. 455–460.

Grinold, R. C. “Infinite horizon stochastic programs”. In: *SIAM journal on control and optimization* 24.6 (1986), pp. 1246–1260.

Gröwe-Kuska, N., H. Heitsch, and W. Römisch. “Scenario reduction and scenario tree construction for power management problems”. In: *Power tech conference proceedings, 2003 IEEE Bologna*. Vol. 3. IEEE. 2003, 7–pp.

Heitsch, H. and W. Römisch. “Scenario reduction algorithms in stochastic programming”. In: *Computational optimization and applications* 24.2-3 (2003), pp. 187–206.

Heitsch, H., W. Römisch, and C. Strugarek. “Stability of multistage stochastic programs”. In: *SIAM Journal on Optimization* 17.2 (2006), pp. 511–525.

Helseth, A. et al. “Co-optimizing sales of energy and capacity in a hydropower scheduling model”. In: *PowerTech, 2015 IEEE Eindhoven*. IEEE. 2015, pp. 1–6.

Hoffman, A. J. “On approximate solutions of systems of linear inequalities.” In: *Journal of Research of the National Bureau of Standards* 49.4 (1952), pp. 263–265.

Høyland, K. and S. W. Wallace. “Generating scenario trees for multistage decision problems”. In: *Management Science* 47.2 (2001), pp. 295–307.

Huang, K. and S. Ahmed. “The value of multistage stochastic programming in capacity planning under uncertainty”. In: *Operations Research* 57.4 (2009), pp. 893–904.

Jacobs, J. et al. “SOCRATES: A system for scheduling hydroelectric generation under uncertainty”. In: *Annals of Operations Research* 59.1 (1995), pp. 99–133.

Jiang, R., Y. Guan, and J.-P. Watson. “Cutting planes for the multistage stochastic unit commitment problem”. In: *Mathematical Programming* 157.1 (2016), pp. 121–151.

Jin, S. et al. “Modeling and solving a large-scale generation expansion planning problem under uncertainty”. In: *Energy Systems* 2.3-4 (2011), pp. 209–242.

Kiwiel, K. C. “An aggregate subgradient method for nonsmooth convex minimization”. In: *Mathematical Programming* 27.3 (1983), pp. 320–341.

Kuhn, D. *Generalized bounds for convex multistage stochastic programs*. Vol. 548. Springer Science & Business Media, 2006.

Laporte, G. and F. V. Louveaux. “The integer L-shaped method for stochastic integer programs with complete recourse”. In: *Operations research letters* 13.3 (1993), pp. 133–142.

Lemaréchal, C., A. Nemirovskii, and Y. Nesterov. “New variants of bundle methods”. In: *Mathematical programming* 69.1 (1995), pp. 111–147.

Li, T., M. Shahidehpour, and Z. Li. “Risk-constrained bidding strategy with stochastic unit commitment”. In: *IEEE Transactions on Power Systems* 22.1 (2007), pp. 449–458.

Löhndorf, N., D. Wozabal, and S. Minner. “Optimizing trading decisions for hydro storage systems using approximate dual dynamic programming”. In: *Operations Research* 61.4 (2013), pp. 810–823.

Lulli, G. and S. Sen. “A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems”. In: *Management Science* 50.6 (2004), pp. 786–796.

Manne, A. S. “Capacity expansion and probabilistic growth”. In: *Econometrica: Journal of the Econometric Society* (1961), pp. 632–649.

Meibom, P. et al. “Stochastic optimization model to study the operational impacts of high wind penetrations in Ireland”. In: *Power Systems, IEEE Transactions on* 26.3 (2011), pp. 1367–1379.

Mokrian, P. and M. Stephen. “A stochastic programming framework for the valuation of electricity storage”. In: *26th USAEE/IAEE North American Conference*. 2006, pp. 24–27.

Möller, A., W. Römisch, and K. Weber. “Airline network revenue management by multistage stochastic programming”. In: *Computational Management Science* 5.355–377 (2008).

Morales-España, G., J. M. Latorre, and A. Ramos. “Tight and compact MILP formulation for the thermal unit commitment problem”. In: *IEEE Transactions on Power Systems* 28.4 (2013), pp. 4897–4908.

Nemhauser, G. and L. Wolsey. *Integer and Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley, 1999.

Newham, N. and A Wood. “Transmission investment planning using SDDP”. In: *Power Engineering Conference, 2007. AUPEC 2007. Australasian Universities*. IEEE. 2007, pp. 1–5.

Nielsen, S. S. and S. A. Zenios. “A stochastic programming model for funding Single Premium Deferred Annuities”. In: *Mathematical Programming* 75.2 (1996), pp. 177–200.

Nowak, M. P. and W. Römisch. “Stochastic Lagrangian relaxation applied to power scheduling in a hydro-thermal system under uncertainty”. In: *Annals of Operations Research* 100.1-4 (2000), pp. 251–272.

Ostrowski, J., M. F. Anjos, and A. Vannelli. “Tight mixed integer linear programming formulations for the unit commitment problem”. In: *IEEE Transactions on Power Systems* 27.1 (2012), pp. 39–46.

Pan, K. and Y. Guan. “Strong Formulations for Multistage Stochastic Self-Scheduling Unit Commitment”. In: *Operations Research* 64.6 (2016), pp. 1482–1498.

Pennanen, T. “Epi-convergent discretizations of multistage stochastic programs via integration quadratures”. In: *Mathematical Programming* 116.1-2 (2009), pp. 461–479.

Pereira, M. V. and L. M. Pinto. “Multi-stage stochastic optimization applied to energy planning”. In: *Mathematical programming* 52.1-3 (1991), pp. 359–375.

Pflug, G. C. “Scenario tree generation for multiperiod financial optimization by optimal discretization”. In: *Mathematical programming* 89.2 (2001), pp. 251–271.

Philpott, A. B. and Z Guan. “On the convergence of stochastic dual dynamic programming and related methods”. In: *Operations Research Letters* 36.4 (2008), pp. 450–455.

Philpott, A. B. and V. L. de Matos. “Dynamic sampling algorithms for multi-stage stochastic programs with risk aversion”. In: *European Journal of Operational Research* 218.2 (2012), pp. 470–483.

Philpott, A., F. Wahid, and B. Frédéric. “MIDAS: A Mixed Integer Dynamic Approximation Scheme”. In: *Optimization-online* (2016).

Rajan, D. and S. Takriti. “Minimum up/down polytopes of the unit commitment problem with start-up costs”. In: *IBM Res. Rep* (2005).

Rebennack, S. “Combining sampling-based and scenario-based nested Benders decomposition methods: application to stochastic dual dynamic programming”. In: *Mathematical Programming* (2013), pp. 1–47.

Römisch, W. and R. Schultz. “Multistage Stochastic Integer Programs: An Introduction”. In: *Online Optimization of Large Scale Systems*. Ed. by M. Grötschel, S. O. Krumke, and J. Rambau. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 581–600.

Rotting, T. and A Gjelsvik. “Stochastic dual dynamic programming for seasonal scheduling in the Norwegian power system”. In: *IEEE Transactions on Power Systems* 7.1 (1992), pp. 273–279.

Ruszczynski, A. P. and A. Shapiro. *Stochastic programming*. Vol. 10. Elsevier Amsterdam, 2003.

Ryan, S. M., J. McCalley, and D. L. Woodruff. *Long term resource planning for electric power systems under uncertainty*. Tech. rep. Technical report, Iowa State University, 2011.

Schrijver, A. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

Schulze, T. and K. McKinnon. “The value of stochastic programming in day-ahead and intra-day generation unit commitment”. In: *Energy* 101 (2016), pp. 592–605.

Schulze, T., A. Grothey, and K. McKinnon. “A stabilised scenario decomposition algorithm applied to stochastic unit commitment problems”. In: *Rapport technique, The University of Edinburgh, School of Mathematics* (2015).

Sen, S., L. Yu, and T. Genc. “A stochastic programming approach to power portfolio optimization”. In: *Operations Research* 54.1 (2006), pp. 55–72.

Shapiro, A. “Inference of statistical bounds for multistage stochastic programming problems”. In: *Mathematical Methods of Operations Research* 58.1 (2003), pp. 57–68.

— “On a time consistency concept in risk averse multistage stochastic programming”. In: *Operations Research Letters* 37.3 (2009), pp. 143–147.

— “Analysis of stochastic dual dynamic programming method”. In: *European Journal of Operational Research* 209.1 (2011), pp. 63–72.

Shapiro, A. “Minimax and risk averse multistage stochastic programming”. In: *European Journal of Operational Research* 219.3 (2012), pp. 719–726.

Shapiro, A. et al. “Risk neutral and risk averse stochastic dual dynamic programming method”. In: *European journal of operational research* 224.2 (2013), pp. 375–391.

Shiina, T. and J. R. Birge. “Stochastic unit commitment problem”. In: *International Transactions in Operational Research* 11.1 (2004), pp. 19–32.

Singh, K. J., A. B. Philpott, and R. K. Wood. “Dantzig-Wolfe decomposition for solving multistage stochastic capacity-planning problems”. In: *Operations Research* 57.5 (2009), pp. 1271–1286.

Sturt, A. and G. Strbac. “Efficient stochastic scheduling for simulation of wind-integrated power systems”. In: *IEEE transactions on Power Systems* 27.1 (2012), pp. 323–334.

Tahanan, M. et al. “Large-scale Unit Commitment under uncertainty”. In: *4OR* 13.2 (2015), pp. 115–171.

Takriti, S., J. R. Birge, and E. Long. “A stochastic model for the unit commitment problem”. In: *Power Systems, IEEE Transactions on* 11.3 (1996), pp. 1497–1508.

Takriti, S., B. Krasenbrink, and L. S.-Y. Wu. “Incorporating fuel constraints and electricity spot prices into the stochastic unit commitment problem”. In: *Operations Research* 48.2 (2000), pp. 268–280.

Thomé, F. et al. “Non-Convexities Representation on Hydrothermal Operation Planning Using SDDP”. In: *URL: www. psr-inc. com, submitted* (2013).

Tseng, C.-L. *On power system generation unit commitment problems*. University of California, Berkeley, 1996.

Tuohy, A. et al. “Unit commitment for systems with significant wind penetration”. In: *IEEE Transactions on Power Systems* 24.2 (2009), pp. 592–601.

Uçkun, C., A. Botterud, and J. R. Birge. “An improved stochastic unit commitment formulation to accommodate wind uncertainty”. In: *IEEE Transactions on Power Systems* 31.4 (2016), pp. 2507–2517.

Wallace, S. W. and S.-E. Fleten. “Stochastic programming models in energy”. In: *Handbooks in Operations Research and Management Science* 10 (2003), pp. 637–677.

Wang, J. et al. “Stochastic unit commitment with sub-hourly dispatch constraints”. In: *Applied energy* 105 (2013), pp. 418–422.

Wu, L., M. Shahidehpour, and T. Li. “Stochastic security-constrained unit commitment”. In: *IEEE Transactions on Power Systems* 22.2 (2007), pp. 800–811.

Zheng, Q. P. et al. “A decomposition approach to the two-stage stochastic unit commitment problem”. In: *Annals of Operations Research* 210.1 (2013), pp. 387–410.

Zheng, Q. P., J. Wang, and A. L. Liu. “Stochastic optimization for unit commitment – A review”. In: *IEEE Transactions on Power Systems* 30.4 (2015), pp. 1913–1924.