Multi-stage Stochastic Capacity Expansion: Models and Algorithms

MULTI-STAGE STOCHASTIC CAPACITY EXPANSION: MODELS AND ALGORITHMS

BY

MAJID TAGHAVI, M.Sc., B.Sc.

A THESIS

SUBMITTED TO THE DEGROOTE SCHOOL OF BUSINESS

AND THE SCHOOL OF GRADUATE STUDIES

OF McMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Ph.D. Dissertation (2015) McMaster University
(DeGroote School of Business) Hamilton, Ontario, Canada

TITLE:	Multi-stage Stochastic Capacity Expansion: Models and Algorithms
AUTHOR:	Majid Taghavi DeGroote School of Business McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. Kai Huang
COMMITTEE MEMBERS:	Dr. Kai Huang Dr. Parkash L. Abad Dr. Manish Verma
NUMBER OF PAGES:	xii, 129

تقدیم به همسرِ عزیز و مهربانم فرشته

برای عشق، گذشت، و فداکاریهایش

و تقدیم به دو فرشتهٔ نازنینی که به من هدیه داد، کیانا و النا

و تقدیم به پدر و مادرِ عزیزم

برای رنج دوری که متحمل شدند

و برای همهٔ تشویقها، حمایتها و دعاهای شبانه روزی شان

*To my sweet and lovely wife, **Fereshteh**,*

for her unconditional love, sacrifices, and supports,

*and to the incredible angels she gave me, **Kiana** & **Elena**,*

*and to my loving parents, **Maryam** and **Akbar***

for their affection, unceasing encouragement, and prayers of day and night.

Abstract

In this dissertation, we study several stochastic capacity expansion models in the presence of permanent, spot market, and contract capacity for acquisition. Using a scenario tree approach to handle the data uncertainty of the problems, we develop multi-stage stochastic integer programming formulations for these models. First, we study multi-period single resource stochastic capacity expansion problems, where different sources of capacity are available to the decision maker. We develop efficient algorithms that can solve these models to optimality in polynomial time. Second, we study multi-period stochastic network capacity expansion problems with different sources for capacity. The proposed models are NP-hard multi-stage stochastic integer programs and we develop an efficient, asymptotically convergent approximation algorithm to solve them. Third, we consider some decomposition algorithms to solve the proposed multi-stage stochastic network capacity expansion problem. We propose an enhanced Benders' decomposition algorithm to solve the problem, and a Benders' decomposition-based heuristic algorithm to find tight bounds for it. Finally, we extend the stochastic network capacity expansion model by imposing budget restriction on permanent capacity acquisition cost. We design a Lagrangian relaxation algorithm to solve the model, including heuristic methods to find tight upper bounds for it.

Acknowledgements

I would never have been able to finish my dissertation without the guidance of my advisor and committee members, help from my friends, and support from my family.

I would like to express my deepest gratitude to my advisor, Dr. Kai Huang, for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for this exciting journey of discovery and exploration. I have had his full support in all aspects of my Ph.D. life. He treated me as his younger brother and taught me lessons on teaching, research, and life.

I would also like to express my special thanks of gratitude to my committee members, Dr. Prakash Abad and Dr. Manish Verma. They put a lot of time and attention into reviewing my dissertation at every stage. Their insightful comments and recommendations have helped me to improve the quality of this dissertation. I would like to thank our area chair, Dr. Parkash Abad, for all his support, and for providing valuable teaching opportunities in the DeGroote School of Business. Also, I would like to thank Dr. Mahmut Parlar and Dr. George Steiner who taught me courses in the past five years. Their high quality courses will always be my treasure.

Last but not least, I take this opportunity to express gratitude to all DeGroote School of Business members for their help and support. I have had great pleasure in working and living with them. My special thanks go to my Ph.D. fellows for creating a friendly and fun atmosphere, especially Amirmohsen, Behrouz, Hesam, and Kamran.

Contents

Abstract **iv**

Acknowledgements **v**

1 Introduction **1**

1.1 Introduction 1

1.2 Outline of the Dissertation 6

1.3 Summary of Contributions 8

2 Literature Review **9**

2.1 Single Resource Capacity Expansion Models 9

2.2 Multiple Resource Capacity Expansion Models 11

3 Single Resource Stochastic Capacity Expansion Models **19**

3.1 Introduction 19

3.2 Single Resource Stochastic Capacity Expansion with Two Sources of Capacity 20

3.2.1 The Primal Algorithm 23

3.2.2 The Dual Algorithm 24

3.2.3 Validity of Algorithms 26
3.2.4 Complexity 33
3.2.5 Experimental Results 34
3.3 Single Resource Stochastic Capacity Expansion with Three Sources of Capacity 35
3.3.1 The Primal Algorithm 39
3.3.2 The Dual Algorithm 40
3.3.3 Validity of Algorithms 41
3.3.4 Complexity 52
3.3.5 Experimental Results 53

4 Stochastic Network Capacity Expansion Models **55**
4.1 Introduction 55
4.2 The Stochastic Network Capacity Expansion Model with Two Sources of Capacity 56
4.3 The Approximation Algorithm 60
4.4 The Stochastic Network Capacity Expansion Model with Three Sources of Capacity 69
4.5 Experimental Results 72
4.5.1 Comparing CPLEX and the Approximation Algorithm 73
4.5.2 Approximation Algorithm's Performance for Large-scale Instances 75

5 Decomposition Algorithms for the Stochastic Network Capacity Expansion Problem **77**
5.1 Classical Benders' Decomposition 77

5.2 Improvements to Classical Benders' Decomposition 82
5.2.1 Valid Inequalities 82
5.2.2 LP Relaxation of the Master Problem 83
5.2.3 Pareto-Optimal Cuts 84
5.2.4 Maximum Feasible Subsystem Cut Generation (MFS) 85
5.3 Computational Results for Classical Benders' Algorithms 87
5.4 New Benders' Decomposition-based Algorithm 92
5.5 Experimental Results for the New Benders' Method 99
5.6 Comparison of the Approximation Algorithm and the New Benders' decomposition-based Algorithm 103

6 Stochastic Network Capacity Expansion Problem with Budget Constraint 107
6.1 The Formulation 107
6.2 Lagrangian Relaxation 109
6.3 Experimental Results for the Lagrangian Relaxation 113

7 Conclusions and Future Research Directions 116
7.1 Conclusions 116
7.2 Future Research Directions 118

List of Tables

3.1 Notations of the model $\mathcal{SCE}_{TWO}$ 21

3.2 Performance of the primal and dual algorithms for $\mathcal{SCE}_{TWO}$ 35

3.3 Notations of the model $\mathcal{SCE}_{THREE}$ 37

3.4 Performance of the primal and dual algorithms for $\mathcal{SCE}_{THREE}$ 54

4.1 Notations of the models $\mathcal{SNCE}_{TWO}$ 57

4.2 Additional notations of the model $\mathcal{SNCE}_{THREE}$ 70

4.3 Data for instances of $\mathcal{SNCE}_{TWO}$ 73

4.4 Comparison of CPLEX and the approximation algorithm for $\mathcal{SNCE}_{TWO}$ 74

4.5 Data for large-scale instances of $\mathcal{SNCE}_{TWO}$ 75

4.6 Approximation algorithm results for large-scale instances of $\mathcal{SNCE}_{TWO}$ 76

5.1 Data for instances of $\mathcal{SNCE}_{TWO}$ 88

5.2 Performance of classical Benders' algorithms for $\mathcal{SNCE}_{TWO}$ 89

5.3 Detailed data of classical Benders' algorithms for instances of $\mathcal{SNCE}_{TWO}$ 91

5.4 Data for small and medium-sized instances of $\mathcal{SNCE}_{TWO}$ 100

5.5 Performance of the new Benders' decomposition-based algorithm for small and medium-sized instances of $\mathcal{SNCE}_{TWO}$ 101

5.6 Data for large-scale instances of $\mathcal{SNCE}_{TWO}$ 102

5.7 Performance of the new Benders' decomposition-based algorithm for large-scale instances of $\mathcal{SNCE}_{TWO}$ 103
5.8 Data for instances of $\mathcal{SNCE}_{TWO}$ 104
5.9 Performance comparison for the approximation algorithm and the new Benders' decomposition-based algorithm for $\mathcal{SNCE}_{TWO}$ 105
6.1 Data for instances of $\mathcal{BSNCE}_{TWO}$ 114
6.2 CPLEX and Lagrangian relaxation algorithm comparison for $\mathcal{BSNCE}_{TWO}$ 114

List of Figures

1.1 The scenario tree $\mathcal{T}$ 6

3.1 The Primal Indexing System 23

3.2 The Dual Indexing System 26

List of Algorithms

1 The Primal Algorithm for $\mathcal{RSCE}_{TWO}$ 24

2 The Dual Algorithm for $\mathcal{DRSCE}_{TWO}$ 26

3 The Primal Algorithm for $\mathcal{RSCE}_{THREE}$ 40

4 The Dual Algorithm for $\mathcal{DRSCE}_{THREE}$ 42

5 The Approximation Algorithm for $\mathcal{SNCE}_{TWO}$ 63

6 Classical Benders' Decomposition for $\mathcal{SNCE}_{TWO}$ 81

7 New Benders' Decomposition-based Algorithm for $\mathcal{SNCE}_{TWO}$ 98

8 Initial Upper Bound Construction Algorithm for Lagrangian Relaxation 112

9 Upper Bound Updating Algorithm for Lagrangian Relaxation 113

Chapter 1

Introduction

1.1 Introduction

Capacity expansion is the process of adding facility, equipment, or personnel of similar type, over time, to meet rising demand (Freidenfelds (1981)). Planning for capacity expansion is primarily concerned with decisions on the optimal time of capacity acquisition, the optimal amount of capacity acquisition, and the optimal capacity allocation. These decisions must be made in order to address future demand growth, which is uncertain. On the other hand, planning for capacity expansion is important strategic level decision-making for a wide range of applications. In most cases, capacity expansion involves significant capital investments over a long horizon, and uncertainties in the future demands. This makes the problem very sophisticated. Capacity expansion problems can be found in communication networks, automobile industry, service industry, electricity industry, semiconductor industry, heavy process industry, water distribution industry, etc. Capacity expansion has been widely studied in the literature since the 1960s (Freidenfelds (1981); Luss (1982); Van Mieghem

(2003)).

Most capacity expansion problems are considered over multiple time periods, and as a result, the developed capacity expansion models are multi-period problems. In the literature regarding mathematical programming, *stochastic programming* (Birge and Louveaux (2011); Ruszczyski and Shapiro (2003)) is one of the tools that are used to handle data uncertainties in mathematical problems. Stochastic programming received a lot of attention because of its power and flexibility in modeling complicated decision problems by handling the non-stationary stochastic process of uncertain data, while imposing few modeling restrictions.

Using the stochastic programming approach, the capacity expansion problem can be modeled either as a *two-stage* stochastic programming problem, or as a *multi-stage* stochastic programming problem. In a two-stage approach, the capacity acquisition decision for the whole planning horizon will be made in the first stage, before the realization of the uncertain data, and the capacity allocation decision will be made at the second stage, after the uncertain data is realized. This means that once the capacity acquisition decision has been made, it cannot be changed. Multi-stage approaches can be seen as an extension to the two-stage modeling in that the capacity acquisition decision can be revised in each time period, based on the data that has been realized up to that period. Therefore, multi-stage stochastic programming gives us more modeling power, but it makes the problem computationally more difficult to solve. Some researches have shown that the value of modeling a problem as a multi-stage stochastic programming problem justifies its complexities over modeling the problem as a two-stage stochastic programming problem (Huang and Ahmed (2009)). On the other hand, this computational issue has been resolved to a great extent by improvements in computational power and algorithm development (Lulli and Sen (2004); Shapiro (2006)).

Early models that consider capacity expansion problems are restricted to a single resource. More recently, considering capacity expansion problems for multiple resource models has received a considerable attention in the literature. A major branch of multiple resource capacity expansion problems is the capacity expansion for networks, where the decision maker is interested in the planning of capacity expansion for network resources, e.g., arcs and nodes. The network capacity expansion problem has a wide range of applications in areas such as transportation (Magnanti and Wong (1984); Ahuja et al. (1996); Liu et al. (2007); Marín and Jaramillo (2008)), telecommunications (Balakrishnan et al. (1991); Magnanti et al. (1995); Lee and Kang (2000)), service industry (Berman and Ganz (1994)), water distribution (Hsu et al. (2008)), and manufacturing (Zhang et al. (2004)).

In classical capacity expansion models, there is only one source of capacity available to purchase by the decision maker. This type of capacity has a permanent nature and is assumed to be available to use till the end of the planning horizon. We refer to this type of capacity as *permanent* capacity. More specifically, permanent capacity refers to the capacity purchased and permanently owned by the decision maker.

In practice, there are other sources of capacity, in addition to permanent capacity, that the decision maker is interested in using. In fact, most decision makers prefer to have a mixed purchasing strategy that makes other sources of capacity available to them (Inderfurth and Kelle (2011); Levin et al. (2012); Inderfurth et al. (2013)). Having such strategies enables them to benefit from other sources of capacity, whenever they have a lower price, or when there is not enough permanent capacity to satisfy the demand. In this dissertation, we introduce two commonly used sources of capacity available to decision makers besides permanent capacity: *spot market* capacity and *contract* capacity.

Spot market capacity refers to the capacity that can only be purchased and used in the current period. Unlike permanent capacity, spot market capacity is temporary in nature and will not be available at the end of the current period. The spot market capacity purchase takes place after the realization of uncertain data at the beginning of each period. This enables the decision maker to fully satisfy the demand for each period. While spot market capacity brings a lot of flexibility to the decision maker, it also brings high risk of price uncertainty.

Contract capacity refers to the capacity that is available in the current period, only if a contract has been signed for it in previous periods. The quantity of contract capacity and the length of the contract are assumed to be fixed. Unlike the permanent capacity, the contract capacity will be only available for a specified number of periods, and not to the end of the planning horizon. Signing a contract for the next periods takes place before the realization of the data for next periods. So, considering the price uncertainty risks, the decision maker will decide whether to buy permanent capacity or to sign for contract capacity for future periods.

Similar to the spot market capacity, contract capacity is temporary in nature, in the sense that the contracted capacity is only available for a specified number of periods. Contract capacity is different from the spot market capacity in the sense that it can be used for more than one period. Even when the length of a contract is only one period, spot market capacity and contract capacity are still different in nature. The former is obtained after the uncertainty in the current period is revealed, while the latter is signed before the uncertainty in the next period is revealed.

Capacity delivery lead time is an important issue to address. In modeling capacity expansion problems, some researchers (Huang and Ahmed (2008)) assume that the permanent capacity can be purchased and used in the current period, i.e., there is no lead time. Delivery lead times can be considered with respect to the application of the model and the way the periods have been defined for the model. In this dissertation, we assume that the permanent capacity purchased in the current period, or the contract capacity signed in the current period will be available to use starting from the next period. For the spot market capacity, we assume that it will be purchased and used in the current period. This means that the delivery lead time for permanent and contract capacity is one and for the spot market capacity is zero.

Through all chapters of this dissertation, we deal with uncertainty of parameters in our models. We assume that these uncertain parameters evolve as discrete-time stochastic processes, and we present them as a scenario tree, which is a well-known stochastic programming tool (Kall and Wallace (1994); Birge and Louveaux (2011); Ruszczyski and Shapiro (2003)). A scenario tree assumes that the number of possible states in each period is finite. In a scenario tree $\mathcal{T}$, each node n represents a possible realization of a set of data. We use a scenario tree to represent the realization of stochastic parameters (demand, cost, etc.), as shown in Figure 1.1. Let $\mathcal{T}$ be the scenario tree with T periods and B branches. If N is the total number of nodes in the scenario tree, and if we assume that B is fixed for all nodes in the scenario tree, then we can calculate the size of the scenario tree as $N = \sum_{t=0}^{T-1} B^t = \frac{B^T-1}{B-1}$. For each node n, $a(n)$ is the immediate ancestor node and $\mathcal{C}(n)$ is the set of all immediate descendants. Also, t_n shows the period of node n. $\mathcal{T}(n)$ denotes the subtree with

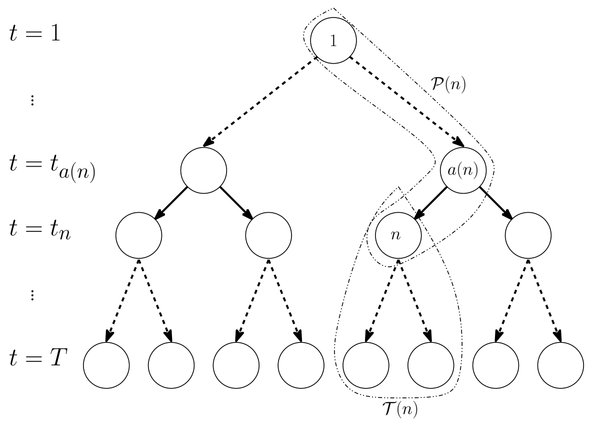

Figure 1.1: The scenario tree $\mathcal{T}$

root node n and $\bar{\mathcal{T}}(n) = \mathcal{T}(n) \setminus \{n\}$. $\mathcal{P}(n)$ denotes the unique path from node n to the root node of the scenario tree and $\bar{\mathcal{P}}(n) = \mathcal{P}(n) \setminus \{n\}$.

Our goal in this dissertation is to study stochastic capacity expansion of both single resource and multiple resource (network) problems in the presence of various sources of capacity for resources[1], and to design efficient exact and approximation algorithms to solve them. For this purpose, a scenario tree approach will be used to handle data uncertainties of the models and multi-stage stochastic programming techniques will be used to model the problems.

1.2 Outline of the Dissertation

In this dissertation, we study stochastic capacity expansion for both single resource and multiple resource problems in the presence of different sources for capacity. In Chapter 2, we review the literature of capacity expansion for single resource and

[1]Note that the use of both source and resource in this dissertation might be confusing. Source refers to the type of capacity that is being purchased (permanent, spot market, and contract). Resource is the entity for which, we consider capacity expansion problem (e.g., node and arc).

multiple resource problems in the context of stochastic programming.

In Chapter 3, we first study multi-period single resource stochastic capacity expansion problems, where two sources of capacity (permanent capacity and spot market capacity) are available for capacity acquisition. Then, we study the extension where three sources of capacity (permanent capacity, spot market capacity and contract capacity) are available to the decision maker. The proposed models are multi-stage stochastic integer programs. We develop efficient algorithms that can solve these models to optimality in polynomial time. We discuss the complexity of algorithms and present the experimental results showing their efficiency when compared to a commercial solver.

In Chapter 4, we study multi-period stochastic network capacity expansion problems with different sources for capacity. The proposed models are NP-hard multi-stage stochastic integer programs that can incorporate multiple sources and different types of capacities in a general network. To solve these models, we develop an efficient approximation algorithm and prove that it is asymptotically convergent to the optimal solution. We demonstrate the efficiency and convergence of the approximation algorithm by presenting the experimental results.

Chapter 5 considers the decomposition algorithms to solve the proposed multi-stage stochastic network capacity expansion problems. We propose a Benders' decomposition algorithm to solve them. Then several techniques will be presented for improved performance. We also develop a Benders' decomposition-based heuristic algorithm that can provide tight bounds for the problems.

In Chapter 6, we extend the stochastic network capacity expansion models introduced in Chapter 4 by imposing budget restriction on permanent capacity acquisition cost. We show that the structural properties that enable the approximation algorithm to solve the problem will be subverted by introducing the budget constraints, and as a result, new algorithms must be designed for this extension. We present a Lagrangian relaxation algorithm to solve the models, including heuristic methods to find upper bounds for it.

A summary of the major contributions of the dissertation, as well as possible future research directions are presented in Chapter 7.

1.3 Summary of Contributions

In this dissertation, we study both single resource and multiple resource capacity expansion problems. For single resource problems, we develop multi-stage stochastic programming formulations involving spot market and contract capacities. Also, we present polynomial-time algorithms to solve them. For multiple resource (network) capacity expansion problems, we develop multi-stage stochastic programming formulations incorporating min-cost network flow model. To solve them, we present an asymptotically convergent approximation algorithm and a Benders' decomposition-based heuristic algorithm. Both these algorithms can outperform CPLEX MIP solver. Finally, we present a multi-stage stochastic programming formulation for the stochastic network capacity expansion problem with budget constraint. A Lagrangian relaxation algorithm is presented to solve this model.

Chapter 2

Literature Review

The capacity expansion problem and its extensions have been extensively studied (Freidenfelds (1981); Luss (1982); Van Mieghem (2003)). We study the literature of capacity expansion in two streams: single resource models and multiple resource models.

2.1 Single Resource Capacity Expansion Models

Early approaches to solve capacity expansion problems are limited to a single resource with deterministic data. Luss (1984) considered a deterministic multi-period capacity expansion problem for a single resource that serves the demand for a range of products. He assumed all cost functions to be concave and found optimal policies for minimizing the total cost using dynamic programming. Rocklin et al. (1984) also considered a single resource capacity expansion where capacity can be both added or dropped. They applied dynamic programming to their model and found an optimal policy for the problem. Aneja and Chaouch (1993) extended the work of Rocklin et al.

(1984) by considering fixed costs for capacity expansions. They also used dynamic programming and found the optimal policy for concave demand distribution functions. Saniee (1995) considered multi-period capacity expansion of a single location in telecommunication systems and modeled it as a time-dependent knapsack problem. He proposed a combination of dynamic programming and shortest path problem to solve the model.

Laguna (1998) extended the work of Saniee (1995) to the case of uncertain demand. He used scenarios to capture the data uncertainty of the problem and applied a robust optimization technique to solve the problem. Riis and Andersen (2004) considered the capacity installation on a single telecommunication facility with stochastic demand. They provided both two-stage and multi-stage formulation for the problem. All these works only considered permanent capacity.

Interestingly, Ahmed et al. (2003) showed that there is a one to one correspondence between stochastic single resource capacity expansion problem and stochastic lot-sizing problem. In fact, stochastic lot-sizing problem can be seen as a sub-structure of the stochastic single resource capacity expansion problem. They developed a branch-and-bound algorithm to solve the model. Huang and Ahmed (2008) studied a multi-stage stochastic version of capacity expansion for a single resource, in which stochastic demand and cost are represented by a scenario tree. They also considered the lot-sizing problem as a sub-structure of single resource capacity expansion problems. They developed polynomial-time algorithms to solve the problem.

There are some works in the literature of capacity expansion that consider spot market and contract capacity. Atamtürk and Hochbaum (2001) studied subcontracting in deterministic multi-period capacity expansion problems. Oren et al. (2005)

used game theory to model capacity expansion of electric power networks with spot market. Examples of applications in cellular manufacturing and flexible manufacturing systems can be found in Lee et al. (1997), Logendran and Puvanunt (1997), and Logendran and Ramakrishna (1997). A related but different research stream is capacity reservation with spot market, as appeared in Erkoc and Wu (2009) and Inderfurth et al. (2013).

The models that we consider in Chapter 3 are different from the above-mentioned models in that we consider three sources of capacity available to the decision maker.

2.2 Multiple Resource Capacity Expansion Models

Multiple resource capacity expansion models can be used for problems involving more that one resource (Ahmed et al. (2003)). Many of these models can be seen as a network of resources. In this section, we primarily focus on network capacity expansion problems.

The study of network capacity expansion problems started with the early work of Fulkerson (1959). He studied arc capacity expansion for a network in order to maximize the flow of the network. He assumed linear capacity expansion costs and budget restriction, and developed a labeling algorithm to solve the model. Doulliez and Rao (1971) considered a network with a common source and multiple sinks. They allowed capacity reduction in their model and assumed that demand is increasing with time for some nodes. They developed a dual network flow algorithm to find the optimal time to satisfy all demand. Christofides and Brooker (1974) considered a network capacity expansion problem with pre-specified arc capacities and budget constraint. The model could find which arcs needed to be added to the network, while maximizing the flow subject to the budget constraint. They developed a tree search algorithm based on dynamic programming. Doulliez and Rao (1975) applied shortest path problem to model a deterministic network capacity expansion problem where arcs are subject to failure and used Dijkstra's algorithm to solve it. Bansal and Jacobsen (1975) developed a Benders' decomposition method to maximize the flow in a network capacity expansion problem with non-linear capacity expansion costs, a fixed budget, and one source and one sink for the network. All these early works are limited to the deterministic setting, where only permanent capacity is available.

In more recent studies with the same limitations, researchers studied deterministic network capacity expansion in the context of multi-commodity flow problem. Leung et al. (1990) studied capacity expansion of distribution centers in a route planning network. Their model considers node capacity expansion only. The original problem is decomposed into subproblems, solved by a Lagrangian relaxation approach. There are other works in the same context which considered arc capacity expansion only. Magnanti et al. (1995) studied telecommunication network capacity installation with two types of facilities and generalized the results to multiple types of facilities. They proposed both Lagrangian relaxation and a cutting plane approach with three set of valid inequalities to solve the model. Bienstock and Günlük (1996) also proposed a mixed-integer programming formulation for a generic telecommunication network capacity installation with the objective of minimizing both capacity installation and flow costs. They studied the polyhedral structure of their model and provided extensions on the valid inequalities proposed in Magnanti et al. (1995). Bienstock et al.

(1998) studied a special case of the generic model presented in Bienstock and Günlük (1996), where they considered a directed graph and tried to minimize the capacity installation cost only. They came up with different sets of valid inequalities and proposed two solution approaches and compared their results. Günlük (1999) considered additional capacity installation for arcs of a capacitated network and used an aggregate multi-commodity flow formulation to model it. He developed a new branching technique and incorporated it into a branch-and-cut algorithm to solve the model. In a more relevant work, Ahuja et al. (1996) addressed arc capacity expansion in a transshipment network with the objective of minimizing the flow cost subject to a budget constraint. Using a parametric network flow problem, they developed an efficient simplex-based algorithm that can solve both linear and integer versions of the problem. Liu et al. (2007) proposed a railroad network model integrating yard location and yard capacity expansion decisions. Their model captured both arc capacity (flow) and node capacity (car handling) and a greedy algorithm was presented to solve their integrated model. In this work, the additional capacity quantity was fixed and the decision was whether to add this quantity or not, while in our model, we decide on the quantity of capacity expansion. Hsu et al. (2008) considered the capacity expansion of water distribution networks using network flow problem. Their model turned out to be a linear model and they tested the result of their model on a real life water distribution system in Taiwan. The models presented in Chapter 4 are different from all above mentioned models in that first, they consider stochastic cost and demand, and second, they allow more than one type of capacity.

With the advent of stochastic programming, researchers have considered stochastic network capacity expansion models. Sen et al. (1994) used a two-stage stochastic programming approach to model private line telecommunication networks. They considered arc capacity expansion and developed a stochastic decomposition algorithm to solve the problem. Berman and Ganz (1994) considered the capacity expansion problem in the service industry. They used a scenario tree and solved a multi-stage stochastic model, which determines the optimal time, location and size of capacity expansion with the objective of maximizing the profit. They also considered a fixed-charge version of the problem and developed an efficient heuristic to solve it. Rajagopalan et al. (1998) studied a multi-stage stochastic capacity planning model with concave expansion costs. They assumed a single product family with non-decreasing deterministic demand when the capacity availability time is uncertain. They designed an efficient dynamic programming algorithm to solve their problem. Chen et al. (2002) developed a multi-stage stochastic programming model to capture technology choice and capacity expansion. They also used a scenario tree approach to deal with the uncertainty of product life cycle and demand. They developed an augmented Lagrangian relaxation approach to solve the model. Christie and David Wu (2002) proposed a multi-stage stochastic programming model of capacity expansion for a semiconductor manufacturer. They assumed demand and capacity to be uncertain and used two different scenario tree approaches. They explored the areas for which each scenario approach is more efficient. Karabuk and Wu (2003) also considered the capacity expansion at a semiconductor manufacturer with uncertain demand and manufacturing capacity. They proposed a planning decomposition approach which could solve the problem to near optimality.

Ahmed et al. (2003) considered a multi-period model for multiple resource capacity expansion problem with integer variables. They developed a multi-stage stochastic integer formulation using a scenario tree approach to handle the uncertain demand, variable and fixed cost of capacity expansion and demand. They proposed a reformulation of their problem using the lot-sizing sub-structure of it and proposed a heuristic method that could find good solutions by perturbing the LP relaxation solutions. Ahmed and Sahinidis (2003) developed a multi-stage stochastic integer model to study a capacity expansion problem for multiple production facilities in the context of a chemical processing network. They assumed costs (both variable and fixed) and demand to be uncertain and used a scenario tree approach to handle the data uncertainty. They developed an approximation scheme using the decomposable structure of their model. The approximation scheme was linear programming based and its solution converges to the optimal solution when the size of the problem increases. Finally, they tested their heuristic method and its asymptotic convergence property on a capacity expansion problem from a chemical processing network context. Tarhan and Grossmann (2008) also addressed the design of chemical process networks with time varying uncertain yields. Their model integrates expanding process capacity and locating production plants. They proposed a duality-based branch-and-bound method to solve their model.

Huang and Ahmed (2009) proposed a multi-stage stochastic programming formulation for a general class of multiple resource capacity expansion problems. They used a scenario tree to realize the uncertain cost and demand. They defined the value of multi-stage stochastic programming (VMS) by comparing two-stage and multi-stage formulation for the general capacity expansion problem. They also provided analytical bound for VMS. Moreover, by exploiting the decomposable structure of their model, they could develop an approximation scheme to solve their model. The approximation scheme consists of two main steps. The first step solves the single resource capacity expansion problem for each facility of the problem, and the second step solves an independent network flow problem for each node in the scenario tree. They proved that their approximation scheme is asymptotically convergent, i.e. the solution by the approximation scheme will converge to the optimal solution, if the number of periods under consideration is large enough. They tested their method on some business problems from a semiconductor tool planning context. Singh et al. (2009) applied the Dantzig-Wolfe decomposition method to solve a multi-stage stochastic mixed integer programming for capacity expansion of production facilities. They used a scenario tree to represent the uncertainty of the problem and incorporated a variable splitting technique into their decomposition method. They tested their solution method on a model from electricity distribution network context. Rasekh and Desrosiers (2010) proposed a model for outsourcing in production planning. Their model helps the decision makers to find the optimal policy between capacity expansion of their own facilities, outsourcing from local suppliers with assured quality, and outsourcing from overseas suppliers with uncertainty in quality. They proposed three algorithms to solve their model, including a column generation, a Benders' decomposition, and another decomposition approach. Pimentel et al. (2013) proposed a multi-stage stochastic mixed-integer programming model for a dynamic network design problem which integrates facility location, capacity expansion, and network design decisions. They considered capacity expansion of facilities (nodes) and used a scenario tree approach to model the uncertainties of demands. They developed a Lagrangian heuristic procedure which can find feasible solutions with reasonably good bounds.

There are other approaches to deal with stochastic network capacity expansion problems other than stochastic programming. Atamtürk and Zhang (2007), Mudchanatongsuk et al. (2007), Ordóñez and Zhao (2007), and Karoonsoontawong (2010) are examples of addressing arc capacity expansion problem with a robust optimization approach. Ordóñez and Zhao (2007) proposed a robust optimization formulation for arc capacity expansion problem with uncertain demand and travel time. They showed that a conic linear counterpart can be solved instead of the original problem and solved it using an interior point method in polynomial time. Mudchanatongsuk et al. (2007) considered the network design of a multi-commodity flow problem with availability for arc capacity expansion. They assumed the transportation cost and demand to be stochastic and the capacity expansion cost to be linear and deterministic. They used a robust optimization approach to deal with the uncertainties of their problem and used column generation technique to solve their model. Another approach in dealing with stochastic capacity expansion is Markov Decision Processes as in Bean et al. (1992), Bhatnagar et al. (1998), and Pratikakis (2010).

Network capacity expansion can be applied to a series of applications. For example, in telecommunication network systems, the number of users usually increase with time, and as a result, the decision makers have to expand the capacity of network in order to be able to satisfy the current users and to absorb new customers. The capacity expansion can be considered for concentrators, cables, number of switches, etc. Balakrishnan et al. (1991) presented an overview of local access telecommunication systems and focused on an expansion planning model for the local access component of public telephone networks based on the new technologies. Chang and Gavish (1993) considered capacity expansion and network design simultaneously, and developed a Lagrangian based heuristic to solve the model. Balakrishnan et al. (1995) also considered local access telecommunication network expansion with piece-wise linear and concave expansion costs. They combined Lagrangian relaxation and dynamic programming to solve the problem. Magnanti et al. (1995) considered telecommunication network capacity installation on a two-type facility problem and generalized the result for multiple type facilities. First type of facilities has a capacity of 1 unit while the second type's capacity is C units. They studied the problem in the context of multi-commodity flow problem and proposed both Lagrangian and cutting plane solution approaches. Bienstock and Günlük (1996) also proposed a mixed-integer formulation for telecommunication network capacity installation and studied its polyhedral structure which led to a cutting plane algorithm for the problem. In another work, Bienstock et al. (1998) presented two variations of minimum capacity installation cost problem in the context of multi-commodity flow problem and came up with different sets of valid inequalities for them. Lee and Kang (2000) considered the capacity expansion of cells in wireless communications. They formulate their model as an integer programming problem and proposed a Tabu search algorithm to solve it. Luna and Mahey (2000) provided a model that could capture both routing and arc capacity expansion in telecommunication systems and found tight bounds for their approximated solution.

Chapter 3

Single Resource Stochastic Capacity Expansion Models

3.1 Introduction

This chapter considers two models of single resource stochastic capacity expansion problem. Traditionally, the models developed for capacity expansion problems assume that there is only one type of capacity available for acquisition. This capacity is purchased and permanently owned by the decision maker and we call it the permanent capacity. However, in real world, there are other types of capacity that the decision maker can use. In this chapter, we introduce two more types of capacity available to decision maker: Spot market capacity which refers to the capacity that can only be purchased and used in the current period, and contract capacity which refers to the capacity that will be available in the current period, only if a contract has been signed for it in a previous period. It is noteworthy that the decision for spot market capacity acquisition will be made after the demand is realized in each period, while the decision for contract capacity acquisition will be made before the demand is realized.

In the following, we start with a single resource stochastic capacity expansion model with permanent and spot market capacity (Section 3.2), and then we consider an extension where contract capacity is available as well (Section 3.3).

3.2 Single Resource Stochastic Capacity Expansion with Two Sources of Capacity

In this section, we consider a multi-period single resource stochastic capacity expansion problem in which permanent capacity and spot market capacity are available to the decision maker simultaneously. We model the problem using multi-stage stochastic programming approach. As mentioned before, we assume that permanent capacity acquisition decision will be made at the beginning of each period, before the realization of the random data in that period; and spot market capacity acquisition decision will be made after the realization of the random data in that period. This implies that the permanent capacity acquisition variable of any scenario tree node will appear in constraints related to its descendant nodes and the spot market acquisition variable of any scenario tree node will only appear in the constraint related to the same node. This model is called $\mathcal{SCE}_{TWO}$ and its notation is presented in Table 3.1.

Without loss of generality, we assume that there is no previously acquired permanent or spot market capacity. We also assume that d_n, c_n^s, and c_n^p are positive values for all $n \in \mathcal{T}$. Moreover, all costs are discounted to their present value. Given these assumptions, the single resource stochastic capacity expansion problem with permanent and spot market capacity can be formulated as follows:

Table 3.1: Notations of the model $\mathcal{SCE}_{TWO}$

Parameters:	
c_n^p	Permanent capacity acquisition cost for node n
c_n^s	Spot market capacity acquisition cost for node n
p_n	Probability of node n
d_n	Demand of node n
Decision variables:	
x_n	Permanent capacity acquisition in node n
z_n	Spot market capacity acquisition in node n

Model $\mathcal{SCE}_{TWO}$:

$$\begin{aligned} \text{Min} \quad & \sum_{n \in \mathcal{T}} p_n \left(c_n^p x_n + c_n^s z_n \right) \\ \text{s.t.} \quad & \sum_{m \in \bar{\mathcal{P}}(n)} x_m + z_n \geq d_n \quad \forall n \in \mathcal{T}, \\ & x_n, z_n \in \mathbb{Z}^+ \qquad\qquad \forall n \in \mathcal{T}. \end{aligned} \tag{3.1}$$

In model $\mathcal{SCE}_{TWO}$, the objective minimizes the expected total acquisition cost of both permanent and spot market capacity, and the first set of constraints force the total available capacity to be no less than the demand for each node. Note that the permanent capacity purchased in the current period will be available to use in the next period. As a result, permanent capacity acquisition variable x_n will appear in $\bar{\mathcal{T}}(n)$. Moreover, model $\mathcal{SCE}_{TWO}$ has the total unimodularity property (Wolsey and Nemhauser (1988)):

Theorem 3.1. *If the demands are integer-valued, the LP relaxation of $\mathcal{SCE}_{TWO}$ will yield integral optimal solutions.*

Proof. Suppose A is the left hand side matrix of the model $\mathcal{SCE}_{TWO}$. Note that A is a 0-1 matrix and each row of it corresponds to a node in the scenario tree. Consider the z_n columns first. There is only a single 1 in each column and all other entries are zero. Now, consider columns of x_n. Entry of row i and column j is 1 only if $i \in \bar{\mathcal{T}}(j)$. Now, we reorder the constraints based on depth-first search method (for each column j, the rows corresponding to $\bar{\mathcal{T}}(j)$ are consecutive, and row $i_1 \in \bar{\mathcal{T}}(j)$ appears before row $i_2 \in \bar{\mathcal{T}}(j)$ if $i_2 \in \mathcal{T}(i_1)$). This new ordering guarantees that in each column, the 1s appear consecutively. Therefore, A is an interval matrix which is totally unimodular. □

Theorem 3.1 shows that if the demands are integer-valued, we can solve a linear relaxation of $\mathcal{SCE}_{TWO}$ instead of the original integer problem. The relaxed version is Model $\mathcal{RSCE}_{TWO}$:

$$\begin{aligned} \text{Min} \quad & \sum_{n \in \mathcal{T}} (c_n^p x_n + c_n^s z_n) \\ \text{s.t.} \quad & \sum_{m \in \bar{\mathcal{P}}(n)} x_m + z_n \geq d_n \quad \forall n \in \mathcal{T}, \\ & x_n, z_n \in \mathbb{R}^+ \qquad\qquad \forall n \in \mathcal{T}. \end{aligned} \tag{3.2}$$

Note that for simplicity of exposition, we have removed the p_n's from the objective function. In the following, all our algorithms are designed for (3.2) which is an LP[1] model.

[1]Linear Programming

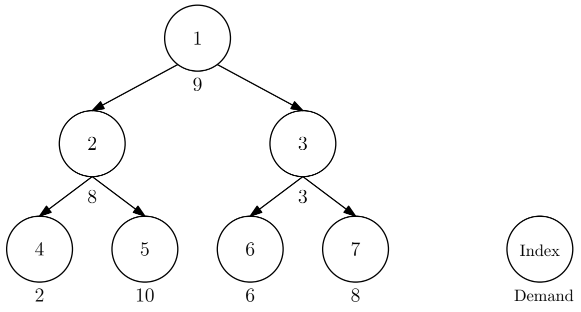

Figure 3.1: The Primal Indexing System

3.2.1 The Primal Algorithm

In this section, we propose a polynomial-time primal algorithm that can solve the stochastic single resource capacity expansion problem $\mathcal{RSCE}_{TWO}$.

The primal algorithm is based on a shifting-up procedure. It compares the acquisition cost of permanent capacity in an ancestor node with the total acquisition cost of permanent and spot market capacities in descendant nodes; and if it is more economical, the capacities of descendant nodes will be shifted to the ancestor node as permanent capacity. For this algorithm, we index the nodes in the scenario tree $\mathcal{T}$ based on the increasing order of their time periods and we call it the *primal indexing system*. Figure 3.1 shows an example of a scenario tree with 3 periods indexed by primal indexing system. Numbers under each node is the demand of the node and numbers inside each node represents the index.

The primal algorithm starts with an initial feasible solution where $x_n^* = 0$ and

$z_n^* = d_n$ for all $n \in \mathcal{T}$. We also need the following definitions:

$$
\begin{aligned}
\mathcal{A}^1(n) &= \big\{m \in \bar{\mathcal{T}}(n) : x_m > 0, \text{ and } x_k = 0, \forall k \in \bar{\mathcal{P}}(m) \setminus \mathcal{P}(n)\big\} \\
\mathcal{A}^2(n) &= \big\{m \in \bar{\mathcal{T}}(n) : z_m > 0, \text{ and } x_k = 0, \forall k \in \bar{\mathcal{P}}(m) \setminus \mathcal{P}(n)\big\} \\
\mathcal{A}(n) &= \mathcal{A}^1(n) \cup \mathcal{A}^2(n) \\
s_n &= \sum_{m \in \mathcal{A}^1(n)} c_m^p + \sum_{m \in \mathcal{A}^2(n)} c_m^s \\
\Delta_n &= \text{Min}\left\{ \underset{m \in \mathcal{A}^1(n)}{\text{Min}} x_m, \underset{m \in \mathcal{A}^2(n)}{\text{Min}} z_m \right\}
\end{aligned}
$$

Given these definitions, the primal algorithm is given in Algorithm 1.

Algorithm 1 The Primal Algorithm for $\mathcal{RSCE}_{TWO}$

```
1: set $x_n^* = 0$ and $z_n^* = d_n, \quad \forall n \in \mathcal{T}$.
2: set $k = \text{Max}\{n \in \mathcal{T} : \bar{\mathcal{T}}(n) \neq \emptyset\}$.
3: while $k \geq 1$
4:       compute $\mathcal{A}^1(k)$, $\mathcal{A}^2(k)$, $s_k$ and $\Delta_k$.
5:       while $c_k^p \leq s_k$
6:             $x_m^* = \begin{cases} x_m^* + \Delta_k & \text{if } m = k, \\ x_m^* - \Delta_k & \text{if } m \in \mathcal{A}^1(k). \end{cases}$
7:             $z_m^* = z_m^* - \Delta_k \qquad \text{if } m \in \mathcal{A}^2(k).$
8:             update $\mathcal{A}^1(k)$, $\mathcal{A}^2(k)$, $s_k$ and $\Delta_k$.
9:       end while
10:      $k = k - 1$
11: end while
12: return $x^* = (x_1^*, \cdots, x_N^*)$ and $z^* = (z_1^*, \cdots, z_N^*)$.
```

3.2.2 The Dual Algorithm

In this section, we consider the dual of the stochastic single resource capacity expansion model (3.2) called $\mathcal{DRSCE}_{TWO}$. Let π_n be the dual variable associated with constraint n in primal problem. Then the dual problem can be formulated as follows:

Model $\mathcal{DRSCE}_{TWO}$:

$$\begin{aligned}
\text{Max} \quad & \sum_{n\in\mathcal{T}} d_n \pi_n \\
\text{s.t.} \quad & \sum_{m\in\bar{\mathcal{T}}(n)} \pi_m \le c_n^p && \forall n \in \mathcal{T}, \\
& \pi_n \le c_n^s && \forall n \in \mathcal{T}, \\
& \pi_n \in \mathbb{R}^+ && \forall n \in \mathcal{T}.
\end{aligned} \tag{3.3}$$

We present a polynomial-time dual algorithm to solve this dual problem. The algorithm has a greedy nature and is based on the following observation:

Lemma 3.1. *A solution $\pi = (\pi_1, \pi_2, \cdots, \pi_N)$ is dual feasible if and only if:*

$$0 \le \pi_n \quad \le \quad \text{Min}\left\{ c_n^s, \underset{m\in\bar{\mathcal{P}}(n)}{\text{Min}} \left\{ c_m^p - \sum_{k\in\bar{\mathcal{T}}(m)\setminus\{n\}} \pi_k \right\}\right\}. \tag{3.4}$$

When π is optimal, the second inequality is tight.

Proof. Non-negativity of π_n and $\pi_n \le c_n^s$ are guaranteed by the constraints in (3.3). Also, note that for each n, all constraints in which π_n appears are corresponding to nodes in $\bar{\mathcal{P}}(n)$. Therefore, for all $m \in \bar{\mathcal{P}}(n)$, we have$\sum_{k\in\bar{\mathcal{T}}(m)} \pi_k \le c_m^p$ or $\pi_n + \sum_{k\in\bar{\mathcal{T}}(m)\setminus\{n\}} \pi_k \le c_m^p$. So, for all $m \in \bar{\mathcal{P}}(n)$, we require that $\pi_n \le c_m^p - \sum_{k\in\bar{\mathcal{T}}(m)\setminus\{n\}} \pi_k$, which implies $\pi_n \le \text{Min}_{m\in\bar{\mathcal{P}}(n)} \left\{ c_m^p - \sum_{k\in\bar{\mathcal{T}}(m)\setminus\{n\}} \pi_k \right\}$. To prove the last claim, suppose that in the optimal solution, the second inequality is not tight. Then we can increase π_n to get a new feasible solution with greater objective value, which is a contradiction. □

To implement the dual algorithm, we use another indexing system called *Dual Indexing system* for nodes in the scenario tree. In dual indexing system, nodes are

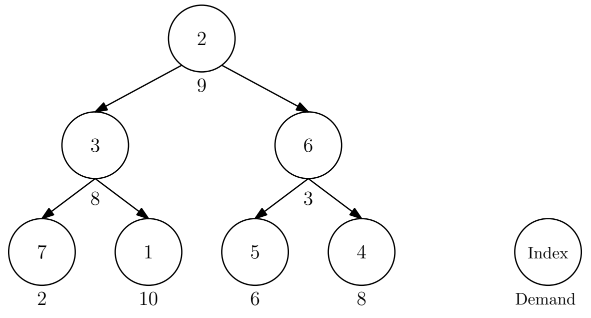

Figure 3.2: The Dual Indexing System

indexed in decreasing order of their demands and we assume that all demands are different, i.e., $d_1 > d_2 > \cdots > d_N > 0$. Figure 3.2 shows an example of a scenario tree indexed by dual indexing system.

The dual algorithm is given in Algorithm 2. In the next section, we prove that the primal and dual algorithms return the optimal solution.

Algorithm 2 The Dual Algorithm for $\mathcal{DRSCE}_{TWO}$

```
1: set $\pi^*_n = 0$ and $c^0_n = c^p_n, \quad \forall n \in \mathcal{T}$.
2: for $k = 1, 2, \cdots, N$
3:     $\pi^*_k = \begin{cases} \underset{m\in\bar{\mathcal{P}}(k)}{\text{Min}} \left\{c^{k-1}_m\right\} & \text{if } \text{Min}_{m\in\bar{\mathcal{P}}(k)} \left\{c^{k-1}_m\right\} \le c^s_k, \\ c^s_k & \text{otherwise.} \end{cases}$
4:     $c^k_n = \begin{cases} c^{k-1}_n - \pi^*_k & \text{if } n \in \bar{\mathcal{P}}(k), \\ c^{k-1}_n & \text{otherwise.} \end{cases}$
5: end for
6: return $\pi^* = (\pi^*_1, \cdots, \pi^*_N)$.
```

3.2.3 Validity of Algorithms

In this section, we verify that the primal and dual algorithms return the primal and dual optimal solutions, respectively. In the following, we assume that d_n, c^s_n, and c^p_n are positive numbers for all $n \in \mathcal{T}$. We first show some properties of the primal solutions produced by Algorithm 1.

Lemma 3.2. *In the primal algorithm, if $x_n^* > 0$, we can find $k \in \bar{\mathcal{T}}(n)$ such that $\sum_{m \in \mathcal{P}(n)} x_m^* = d_k$.*

Proof. The scenario tree has T periods and we use mathematical induction from period T to 1 to prove the statement. Consider the steps in Algorithm 1. The initial solution requires that $x_n^* = 0$ and $z_n^* = d_n$ for all $n \in \mathcal{T}$. The statement is true for the initial solution because $x_n^* = 0$ for all n. For a non-leaf nodes k in period $T-1$, if we shift up capacity from nodes in $\mathcal{A}(k)$ to k as permanent capacity, then $x_k^* = \Delta_k > 0$. Since Δ_k is the minimum demand among all nodes in $\mathcal{A}(k)$, the statement holds. If it is still more economical to shift up capacity from nodes in $\mathcal{A}(k)$ to node k, the next value of x_k^* will be equal to the second smallest demand in $\mathcal{A}(k)$. When this procedure stops for node k, the capacity accumulated in node k is equal to the demand of a node in $\mathcal{A}(k)$, so the statement holds. Now consider any non-leaf nodes k in any period before $t < T-1$. Considering the definition of Δ_k and the structure of initial solution, if Δ_k takes the minimum value among spot market capacities, it will be equal to the demand of a direct descendant node. If Δ_k takes the minimum value among permanent capacities, the values are equal to the demand of a future node due to mathematical induction assumption. Therefore, the statement holds for the whole scenario tree. □

In Lemma 3.3, we show several properties of the dual solutions resulting from Algorithm 2. To facilitate our following proofs, we define m_k for each node k as: $m_k = \text{argmin}_{n \in \bar{\mathcal{P}}(k)} \left\{ c_n^{k-1} \right\}$. When $c_n^{k-1} = c_m^{k-1}$, we let $m_k = m$ if $t_m < t_n$.

Lemma 3.3. *In the dual algorithm, the following results hold:*

(a) If $\pi_k^* = c_{m_k}^{k-1}$*, then* $\sum_{m\in\bar{\mathcal{T}}(m_k)} \pi_m^* = c_{m_k}^p$*, and* $\pi_i^* = 0$*,* $\forall i \in \bar{\mathcal{T}}(m_k)$ *with* $i > k$*.*

(b) If $\pi_k^* = c_{m_k}^{k-1}$*, then* $\sum_{m\in\bar{\mathcal{T}}(i),m<k} \pi_m^* < c_i^p$ *,* $\forall i \in \bar{\mathcal{P}}(m_k)$*.*

(c) If $\pi_k^* = c_k^s$*, then* $\sum_{m\in\bar{\mathcal{T}}(i),m<k} \pi_m^* < c_i^p$ *,* $\forall i \in \bar{\mathcal{P}}(k)$*.*

Proof. (a) Let us define $\mathcal{U}$, $\mathcal{V}$, and $\mathcal{W}$ as follows:

$$\begin{aligned}
\mathcal{U} &= \left\{ i \ : \ i \in \bar{\mathcal{T}}(m_k), \ i < k, \ \pi_i^* = \operatorname*{Min}_{m\in\bar{\mathcal{P}}(i)} \{c_m^{i-1}\} \right\} \\
\mathcal{V} &= \left\{ i \ : \ i \in \bar{\mathcal{T}}(m_k), \ i < k, \ \pi_i^* = c_i^s \right\} \\
\mathcal{W} &= \left\{ i \ : \ i \in \bar{\mathcal{T}}(m_k), \ i > k \right\}
\end{aligned}$$

According to the step 4 of Algorithm 2, when we consider node k, we have $\pi_k^* = c_{m_k}^{k-1} = c_{m_k}^p - \sum_{j\in\bar{\mathcal{T}}(m_k),j<k} \pi_j^* = c_{m_k}^p - \sum_{i\in\mathcal{U}} \pi_i^* - \sum_{i\in\mathcal{V}} \pi_i^*$. Consider the dual constraint: $\sum_{m\in\bar{\mathcal{T}}(m_k)} \pi_m^* = \pi_k^* + \sum_{i\in\mathcal{U}} \pi_i^* + \sum_{i\in\mathcal{V}} \pi_i^* + \sum_{i\in\mathcal{W}} \pi_i^* = c_{m_k}^p + \sum_{i\in\mathcal{W}} \pi_i^* \leq c_{m_k}^p$. This implies that $\sum_{i\in\mathcal{W}} \pi_i^* = 0$ and since $\pi_i^* \geq 0$ for all i, we must have $\pi_i^* = 0$ for each node $i \in \mathcal{W}$. This also shows that $\sum_{m\in\bar{\mathcal{T}}(m_k)} \pi_m^* = c_{m_k}^p$ which completes the proof for (a).

(b) Suppose $i \in \bar{\mathcal{P}}(m_k)$. According to Algorithm 2, when we consider node k, we have $c_i^{k-1} = c_i^p - \sum_{m\in\bar{\mathcal{T}}(i),m<k} \pi_m^*$ or $\sum_{m\in\bar{\mathcal{T}}(i),m<k} \pi_m^* = c_i^p - c_i^{k-1}$. Since $\pi_k^* = c_{m_k}^{k-1}$, based on the definition of m_k, we have $c_i^{k-1} > c_{m_k}^{k-1} \geq 0$. Therefore, $\sum_{m\in\bar{\mathcal{T}}(i),m<k} \pi_m^* < c_i^p$.

(c) According to Algorithm 2, for any node $i \in \bar{\mathcal{P}}(k)$, we can write $c_i^{k-1} = c_i^p - \sum_{m\in\bar{\mathcal{T}}(i),m<k} \pi_m^*$ or $\sum_{m\in\bar{\mathcal{T}}(i),m<k} \pi_m^* = c_i^p - c_i^{k-1}$. If $c_i^{k-1} = 0$, it requires that $\pi_k^* = 0$ which contradicts $\pi_k^* = c_k^s > 0$. So, $\sum_{m\in\bar{\mathcal{T}}(i),m<k} \pi_m^* < c_i^p$. □

Lemma 3.4 shows a property of the dual solution resulting from Algorithm 2. To facilitate our proof, we define sets $\mathcal{U}_i^k$ and $\mathcal{V}_i^k$ for node i at iteration k of the dual algorithm:

$$\mathcal{U}_i^k = \left\{ m_j \;\middle|\; \begin{array}{c} m_j \in \mathcal{T}(i), j < k,\ \pi_j^* = c_{m_j}^{j-1} \\ \text{and there does not exist } l < k \text{ and } l > j \text{ such that} \\ \pi_l^* = c_{m_l}^{l-1},\ m_l \in \mathcal{T}(i),\ \text{and } \mathcal{T}(m_l) \supset \mathcal{T}(m_j) \end{array} \right\}$$

$$\mathcal{V}_i^k = \left\{ j \;\middle|\; \begin{array}{c} j \in \bar{\mathcal{T}}(i),\ j < k,\ \pi_j^* = c_j^s \\ \text{and there does not exist } l < k \text{ and } l > j \text{ such that} \\ \pi_l^* = c_{m_l}^{l-1},\ m_l \in \mathcal{T}(i),\ \text{and } j \in \mathcal{T}(m_l) \end{array} \right\}$$

These sets correspond to all of the capacities that could not be shifted up to node m_k in the primal algorithm.

Lemma 3.4. *In the dual algorithm,*

$$c_{m_k}^{k-1} = c_{m_k}^p - \sum_{m_j \in \mathcal{U}_{m_k}^k} c_{m_j}^p - \sum_{j \in \mathcal{V}_{m_k}^k} c_j^s$$

Proof. Consider node i such that there does not exist $j < k$ and $\pi_j^* = cp_{m_j}^{j-1}$ and $i \in \bar{\mathcal{T}}(m_j)$. Note that for any $j < k$, if $j \in \bar{\mathcal{T}}(i)$, then either $j \in \mathcal{V}_i^k$ or $m_j \in \mathcal{U}_i^k$, or there exists $j < l < k$ such that $\pi_l^* = cp_{m_l}^{l-1}$ and $m_l \in \mathcal{T}(i)$ and $j \in \bar{\mathcal{T}}(m_l)$. Therefore, for node i, we have

$$c_i^{k-1} = c_i^p - \sum_{j \in \bar{\mathcal{T}}(i), j<k} \pi_j^* = c_i^p - \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p - \sum_{j \in \mathcal{V}_i^k} c_j^s$$

The conclusion holds when $i = m_k$. □

Finally, we connect the primal solution and the dual solution in Lemma 3.5 and 3.6, corresponding to the different sources of capacity in the dual solutions:

Lemma 3.5. *In the dual algorithm, if $\pi_k^* = c_{m_k}^{k-1}$, then:*

(a) $z_k^* = 0$, and $x_{m_k}^* > 0$;

(b) $\sum_{m \in \mathcal{P}(m_k)} x_m^* \geq d_k$;

(c) $\forall i \in \bar{\mathcal{P}}(m_k)$, $\sum_{m \in \mathcal{P}(i)} x_m^* < d_k$.

Proof. (a) Since $\pi_k^* = c_{m_k}^{k-1}$, it requires that $c_{m_k}^{k-1} < c_k^s$. On the other hand, from Lemma 3.4, we have $c_{m_k}^{k-1} = c_{m_k}^p - \sum_{m_j \in \mathcal{U}_{m_k}^k} c_{m_j}^p - \sum_{j \in \mathcal{V}_{m_k}^k} c_j^s$ which implies that $c_{m_k}^p < c_k^s + \sum_{m_j \in \mathcal{U}_{m_k}^k} c_{m_j}^p + \sum_{j \in \mathcal{V}_{m_k}^k} c_j^s$. This means that Algorithm 1 will completely shift up the spot market capacity of node k (if any) to m_k, i.e., $z_k^* = 0$.

Next, we need to ensure that the capacity accumulated at node m_k will not be shifted up completely, i.e., $x_{m_k}^* > 0$. Note that for all $i \in \bar{\mathcal{P}}(m_k)$, $c_i^{k-1} = c_i^p - \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p - \sum_{j \in \mathcal{V}_i^k} c_j^s$. Since $c_{m_k}^{k-1} < c_i^{k-1}$, we have $c_{m_k}^p - \sum_{m_j \in \mathcal{U}_{m_k}^k} c_{m_j}^p - \sum_{j \in \mathcal{V}_{m_k}^k} c_j^s < c_i^p - \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p - \sum_{j \in \mathcal{V}_i^k} c_j^s$, which implies that $c_i^p > c_{m_k}^p + \sum_{m_j \in \mathcal{U}_i^k \setminus \mathcal{U}_{m_k}^k} c_{m_j}^p + \sum_{j \in \mathcal{V}_i^k \setminus \mathcal{V}_{m_k}^k} c_j^s$ for all $i \in \bar{\mathcal{P}}(m_k)$. According to Algorithm 1, this means that the capacity in m_k cannot be shifted up completely anymore. So $x_{m_k}^* > 0$.

(b) To the contrary, Suppose that $\sum_{m \in \mathcal{P}(m_k)} x_m^* < d_k$. Since $z_k^* = 0$, we can conclude that there is a node $j \in \bar{\mathcal{P}}(k) \setminus \mathcal{P}(m_k)$ with $x_j^* > 0$. However, in (a), we proved that Algorithm 1 will shift up any capacity of nodes in $\bar{\mathcal{P}}(k) \setminus \mathcal{P}(m_k)$ to node m_k. So, for all $j \in \bar{\mathcal{P}}(k) \setminus \mathcal{P}(m_k)$, x_j^* must be equal to zero, which is a contradiction.

(c) To the contrary, assume that there exists $i \in \bar{\mathcal{P}}(m_k)$ such that $\sum_{m \in \mathcal{P}(i)} x_m^* \geq d_k$. Without loss of generality, assume that i is such a node with the largest t_i. Then in the primal algorithm, we must have $c_i^p \leq \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p + \sum_{j \in \mathcal{V}_i^k} c_j^s$. This implies that node $i = m_k$, which is a contradiction. □

Lemma 3.6. *In the dual algorithm, if $\pi_k^* = c_k^s$, then:*

(a) $z_k^* > 0$;

(b) $\sum_{m \in \bar{\mathcal{P}}(k)} x_m^* + z_k^* = d_k$, and $\sum_{m \in \bar{\mathcal{P}}(k)} x_m^* < d_k$.

Proof. (a) Since $\pi_k^* = c_k^s$, we can conclude that for all $i \in \bar{\mathcal{P}}(k)$, $c_k^s < c_i^{k-1}$. According to Lemma 3.4, $c_i^{k-1} = c_i^p - \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p - \sum_{j \in \mathcal{V}_i^k} c_j^s$. So, $c_i^p > c_k^s + \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p + \sum_{j \in \mathcal{V}_i^k} c_j^s$ which implies that no capacity will be shifted up from node k to nodes $i \in \bar{\mathcal{P}}(k)$ as permanent capacity according to Algorithm 1. Since the initialization of Algorithm 1 enforces all z_n to be positive, we can conclude that $z_k^* > 0$.

(b) To the contrary, assume that $\sum_{m \in \bar{\mathcal{P}}(k)} x_m^* + z_k^* > d_k$. Since $z_k^* > 0$, we can decrease z_k^* to make the constraint tight. This will give us a new feasible solution with less objective value, which contradicts optimality. Then the second half of the claim follows. □

The next theorem will show that the solutions returned by the primal and dual algorithms are optimal.

Theorem 3.2. *The solution $(x^*, z^*) = (x_1^*, x_2^*, \cdots, x_N^*, z_1^*, z_2^*, \cdots, z_N^*)$ returned by the primal algorithm and the solution $\pi^* = (\pi_1^*, \pi_2^*, \cdots, \pi_N^*)$ returned by the dual algorithm are optimal.*

Proof. First, we show that (x^*, z^*) and π^* are both feasible solutions of primal and dual problem, respectively. In the primal algorithm, we start with a feasible solution and in each iteration, we shift up Δ_k capacity which preserve feasibility. The feasibility of π^* is guaranteed by Lemma 3.1.

Next, we need to prove the complementary slackness for primal and dual solutions:

$$\pi_n^* > 0 \implies \sum_{m \in \bar{\mathcal{P}}(n)} x_m^* + z_n^* = d_n \tag{3.5}$$

$$\sum_{m \in \bar{\mathcal{T}}(n)} \pi_m^* < c_n^p \implies x_n^* = 0 \tag{3.6}$$

$$\pi_n^* < c_n^s \implies z_n^* = 0 \tag{3.7}$$

Assume that we are using the dual indexing system. Suppose that $\pi_n^* > 0$. To prove (3.5), two cases are Considered:

Case 1: If $\pi_n^* = c_n^s$, Lemma 3.6(b) guarantees that $\sum_{m \in \bar{\mathcal{P}}(n)} x_m^* + z_n^* = d_n$.

Case 2: If $\pi_n^* = c_{m_n}^{n-1}$, to the contrary, assume that $\sum_{m \in \bar{\mathcal{P}}(n)} x_m^* + z_n^* > d_n$. According to Lemma 3.5(a), $z_k^* = 0$ and $x_{m_n}^* > 0$. Now, according to Lemma 3.2, there is a node $k \in \bar{\mathcal{T}}(m_n)$ such that $\sum_{m \in \mathcal{P}(m_n)} x_m^* = d_k$, which implies that $d_k > d_n$ or $k < n$. Now, consider the optimal dual variable of node k, i.e. π_k^*. If $\pi_k^* = c_k^s$, Lemma 3.6(b) requires that $\sum_{m \in \bar{\mathcal{P}}(k)} x_m^* < d_k$, which contradicts $\sum_{m \in \mathcal{P}(m_n)} x_m^* = d_k$, because $k \in \bar{\mathcal{T}}(m_n)$. If $\pi_k^* = c_{m_k}^{k-1}$, when $m_k \in \bar{\mathcal{T}}(m_n)$, Lemma 3.5(c) requires that $\sum_{m \in \mathcal{P}(m_n)} x_m^* < d_k$ which contradicts $\sum_{m \in \mathcal{P}(m_n)} x_m^* = d_k$; when $m_k \in \mathcal{P}(m_n)$, then since $n > k$ and $n \in \bar{\mathcal{T}}(m_k)$, Lemma 3.3(a) requires that $\pi_n^* = 0$ which contradicts $\pi_n^* > 0$. Case $\sum_{m \in \bar{\mathcal{P}}(n)} x_m^* + z_n^* < d_n$ is impossible, because it violates primal feasibility. Therefore,(3.5) holds.

To prove (3.6), we again use contradiction. Suppose that $\sum_{m \in \bar{\mathcal{T}}(n)} \pi_m^* < c_n^p$ and $x_n^* > 0$ simultaneously. According to Lemma 3.2, we can find a $k \in \bar{\mathcal{T}}(n)$, such that

$\sum_{m\in\mathcal{P}(n)} x_m^* = d_k$. Now, consider the optimal dual value of k, i.e., π_k^*:

Case 1: $\pi_k^* = c_k^s$. Lemma 3.6(b) implies that $\sum_{m\in\bar{\mathcal{P}}(k)} x_m^* < d_k$ which contradicts $\sum_{m\in\mathcal{P}(n)} x_m^* = d_k$, since $\mathcal{P}(n) \subseteq \bar{\mathcal{P}}(k)$.

Case 2: $\pi_k^* = c_{m_k}^{k-1}$. If $m_k \in \bar{\mathcal{T}}(n)$, then $n \in \bar{\mathcal{P}}(m_k)$, and Lemma 3.5(c) requires that $\sum_{m\in\mathcal{P}(n)} x_m^* < d_k$ which contradicts $\sum_{m\in\mathcal{P}(n)} x_m^* = d_k$. If $m_k \in \bar{\mathcal{P}}(n)$, according to Lemma 3.5(b), we have $\sum_{m\in\mathcal{P}(m_k)} x_m^* \geq d_k$. However, since $\sum_{m\in\mathcal{P}(n)} x_m^* = d_k$ and $n \in \bar{\mathcal{T}}(m_k)$, it requires that $x_n^* = 0$ which is a contradiction. So, the only remaining possibility is $m_k = n$. In this case, Lemma 3.3(a) requires that $\sum_{m\in\bar{\mathcal{T}}(m_k)} \pi_m^* = c_{m_k}^p$. Since $m_k = n$, then $\sum_{m\in\bar{\mathcal{T}}(n)} \pi_m^* = c_n^p$ which contradicts $\sum_{m\in\bar{\mathcal{T}}(n)} \pi_m^* < c_n^p$. Therefore, (3.6) holds.

Finally, When $\pi_n^* < c_n^s$, we have a node m_n such that $\pi_n^* = c_{m_n}^{n-1}$. According to Lemma 3.5(a), we have $z_n^* = 0$. Therefore, (3.7) holds. $\square$

3.2.4 Complexity

Suppose the scenario tree is a complete tree with T time periods and B branches for every non-leaf node. Then the number of nodes in the scenario tree is equal to $N = \sum_{t=0}^{T-1} B^t = \frac{B^T-1}{B-1}$, which implies that $T \sim \mathcal{O}(\log N)$.

Theorem 3.3. *The complexity of the primal algorithm is $\mathcal{O}(N^2)$.*

Proof. Initialization step needs at most N operations. Computing each of $\mathcal{A}(k)$, s_k, and Δ_k requires at most N operations. The external **while** (in step 3) will run at most n times. Each iteration of the internal **while** (in step 5) will deplete the capacity for one node and the variable adjustments will be of the complexity of at most $\mathcal{O}(N)$. So the dominant complexity is for two whiles which is $\mathcal{O}(N^2)$. $\square$

Theorem 3.4. *The complexity of the dual algorithm is* $\mathcal{O}(N \log N \log(\log N))$.

Proof. Sorting the demands of N nodes has a complexity of $N \log N$. For each node, updating π_k^* and c_n^k requires at most T minimization and T for new values. Each minimization has a complexity of $T \log T$. Therefore, the total number of operations is at most $N \log N + N(T \log T + T)$. Since $T \sim \mathcal{O}(\log N)$, the number of operations is no more that$N \log N(2 + \log(\log N))$. So, the complexity is $\mathcal{O}(N \log N \log(\log N))$ □

3.2.5 Experimental Results

In this section, we present some experimental results that compare the solution times of the designed primal and dual algorithm with those of CPLEX LP solver.

We create 15 instances of the model $\mathcal{SCE}_{TWO}$ by changing the number of periods in the scenario tree (T) and the number of branches of the scenario tree (B). This results in generating instances with a wide range of scenario tree size (N). The results are shown in Table 3.2. It is important to mention that CPLEX is solving an LP in these tests.

All the codes are written in C++ using IBM/ILOG CPLEX 12.6 Concert Technology. The experiments are conducted on a computer with an Intel Core i3-M330 2.13GHz processor and 4.00 GB of RAM running Ubuntu 14.04 LTS.

Table 3.2 shows that the primal and dual algorithms can perform better than CPLEX LP solver in all instances, especially for large-scale problems. The data presented in Table 3.2 also confirm that the complexity of the dual algorithm is less than the complexity of the primal algorithm.

Table 3.2: Performance of the primal and dual algorithms for $\mathcal{SCE}_{TWO}$

T	B	N	Time (s)			% of CPLEX Time	
			CPLEX	Primal	Dual	Primal	Dual
5	5	781	0.013	0.003	0.001	23.07	7.69
8	3	3,280	0.040	0.010	0.004	25.00	10.00
12	2	4,095	0.043	0.022	0.005	51.16	11.62
5	10	11,111	0.109	0.025	0.011	22.93	10.09
7	5	19,531	0.230	0.044	0.026	19.13	11.30
15	2	32,767	0.669	0.112	0.051	16.74	7.62
8	5	97,656	0.852	0.235	0.133	27.58	15.61
10	4	349,525	6.034	0.934	0.551	15.47	9.13
9	5	488,281	8.826	1.245	0.755	14.10	8.55
13	3	797,161	18.668	2.526	1.397	13.53	7.48
20	2	1,048,575	36.781	4.440	2.161	12.07	5.87
7	10	1,111,111	37.341	4.865	2.505	11.89	5.78
8	8	2,396,745	67.269	5.708	3.844	9.48	5.71
12	4	5,592,405	376.479	16.044	10.225	4.26	2.71
15	3	7,174,453	$*$	23.868	14.109	N/A	N/A

* For the last instance, CPLEX LP solver encounters an out of memory exception.

3.3 Single Resource Stochastic Capacity Expansion with Three Sources of Capacity[*]

In previous section, we studied multi-stage stochastic capacity expansion for a single resource problem with two types of capacity (permanent and spot market) available to purchase. In this section, we add another source of capacity called the contract capacity. In order to have this capacity available in the current period, the decision maker has to sign a contract in previous periods. At the first glance, the three capacity

[*] Published as: Taghavi, M., & Huang, K. (2014). Stochastic Capacity Expansion with Multiple Sources of Capacity. *Operations Research Letters*, 42(4), 263-267

sources model of this section is not very different from the two capacity sources model of the previous section. Thus, one might expect that the algorithm design and complexities will be similar. However, through the rest of this section, it can be seen that the algorithms and their validity proofs are quite different and more sophisticated for the three capacity model of this section. This is because the interactions among three sources of capacity are more complicated than the interactions between two sources of capacity.

Furthermore, it is noteworthy that, although the three capacity model presented in this section assumes that the length of the contract for capacity is one period, this model can capture the case where the decision maker can sign contracts for an arbitrary number of periods. Surprisingly, we will show that the complexity of algorithms that we design will not change, when we implement them for problems allowing multiple period contracts.

This multi-stage stochastic capacity expansion for a single resource problem with three types of capacity (permanent, spot market, and contract) is called $\mathcal{SCE}_{THREE}$ and its notation is presented in Table 3.3.

We assume that we do not have any previously purchased or contracted capacity. We also assume that all costs and demands are positive and all costs are discounted to their present value. With these assumptions, the model $\mathcal{SCE}_{THREE}$ can be formulated as follows:

Table 3.3: Notations of the model $\mathcal{SCE}_{THREE}$

Parameters:	
c_n^p	Unit cost of permanent capacity acquisition for node n
c_n^s	Unit cost of spot market capacity acquisition for node n
c_n^c	Unit cost of contract capacity acquisition for node n
p_n	Probability of node n
d_n	Demand of node n
Decision variables:	
x_n	Permanent capacity acquisition in node n
z_n	Spot market capacity acquisition in node n
y_n	Contract capacity acquisition in node n

Model $\mathcal{SCE}_{THREE}$:

$$\begin{aligned} \text{Min} \quad & \sum_{n \in \mathcal{T}} p_n \left(c_n^p x_n + c_n^c y_n + c_n^s z_n \right) \\ \text{s.t.} \quad & \sum_{m \in \bar{\mathcal{P}}(n)} x_m + y_{a(n)} + z_n \geq d_n \quad \forall n \in \mathcal{T}, \\ & x_n, y_n, z_n \in \mathbb{Z}^+ \qquad\qquad\quad \forall n \in \mathcal{T}, \end{aligned} \tag{3.8}$$

where the objective minimizes the expected total cost of all three types of capacity acquisition over the scenario tree, and the constraint guarantees that for each node in the scenario tree, the total capacity available at node n will satisfy the demand. Note that we purchase the permanent capacity or sign the contract capacity at the beginning of a period, before the realization of the random demand in the same period. So that in model (3.8), we make permanent or contract capacity available in the next period of when they are purchased or signed. In other words, x_n will appear in $\bar{\mathcal{T}}(n)$ and y_n will appear in $\mathcal{C}(n)$. We emphasize that our model can deal with contracts with arbitrary number of periods. For simplicity of exposition, we only use the one-period contract in (3.8). Model $\mathcal{SCE}_{THREE}$ has the following property:

Theorem 3.5. *If the demands are integer-valued, the LP relaxation of* $\mathcal{SCE}_{THREE}$ *will yield integral optimal solutions.*

Proof. Suppose A is the left hand side matrix of $\mathcal{SCE}_{THREE}$. Note that A is a 0-1 matrix and each row in A is corresponding to a node in the scenario tree. Consider the z_n columns. Clearly, there is a single 1 in each column and all other entries are zeros. Consider the y_n columns, entry of row i and column j is 1 only if $i \in \mathcal{C}(j)$. Consider the x_n columns, entry of row i and column j is 1 only if $i \in \bar{\mathcal{T}}(j)$. We can reorder the constraints according to the following procedure: We start from the root node and after each insertion of a node n, we immediately insert all nodes in $\mathcal{C}(n)$. If there is more than one node in $\mathcal{C}(n)$, we will re-start the process from the first inserted node, and when there is no more node to insert, we continue the procedure from the last non-inserted node based on a depth-first method. This new ordering guarantees that in each column, the 1's appear consecutively. Therefore, A is an interval matrix which is totally unimodular. $\square$

Based on Theorem 3.5, if the demands are integer-valued, we can solve the linear relaxation of $\mathcal{SCE}_{THREE}$ which can be modeled as follows:

Model $\mathcal{RSCE}_{THREE}$:

$$\begin{aligned} \text{Min} \quad & \sum_{n\in\mathcal{T}} (c^p_n x_n + c^c_n y_n + c^s_n z_n) \\ \text{s.t.} \quad & \sum_{m\in\bar{\mathcal{P}}(n)} x_m + y_{a(n)} + z_n \geq d_n \quad \forall n \in \mathcal{T}, \\ & x_n, y_n, z_n \in \mathbb{R}^+ \qquad\qquad \forall n \in \mathcal{T}. \end{aligned} \tag{3.9}$$

Note that for simplicity of exposition, we have removed the p_n's from the objective function. In the following, all our algorithms are designed for $\mathcal{RSCE}_{THREE}$.

3.3.1 The Primal Algorithm

In this section, we propose a polynomial-time primal algorithm for $\mathcal{RSCE}_{THREE}$. The primal algorithm will check if it is more economical to shift up capacity to an ancestor node as either permanent capacity or contract capacity by comparing the costs of permanent capacity or contract capacity in an ancestor node with the total cost of permanent, contract, and spot market capacities in descendant nodes.

For this algorithm, we assume that the nodes in the scenario tree $\mathcal{T}$ are indexed by the primal indexing system. The algorithm starts with an initial feasible solution, where $x_n^* = y_n^* = 0$ and $z_n^* = d_n$ for all $n \in \mathcal{T}$. Besides, we need the following definitions:

$$\begin{aligned}
\mathcal{A}^1(n) &= \left\{m \in \bar{\mathcal{T}}(n) : x_m > 0, \text{ and } x_k = 0, \forall k \in \bar{\mathcal{P}}(m) \setminus \mathcal{P}(n)\right\} \\
\mathcal{A}^2(n) &= \left\{m \in \bar{\mathcal{T}}(n) : y_m > 0, \text{ and } x_k = 0, \forall k \in \mathcal{P}(m) \setminus \mathcal{P}(n)\right\} \\
\mathcal{A}^3(n) &= \left\{m \in \bar{\mathcal{T}}(n) : z_m > 0, \text{ and } x_k = y_k = 0, \forall k \in \bar{\mathcal{P}}(m) \setminus \mathcal{P}(n)\right\} \\
\mathcal{A}(n) &= \mathcal{A}^1(n) \cup \mathcal{A}^2(n) \cup \mathcal{A}^3(n) \\
\Delta_n^1 &= \text{Min}\left\{ \underset{m\in\mathcal{A}^1(n)}{\text{Min}}\, x_m,\ \underset{m\in\mathcal{A}^2(n)}{\text{Min}}\, y_m,\ \underset{m\in\mathcal{A}^3(n)}{\text{Min}}\, z_m \right\} \\
\Delta_n^2 &= \underset{\substack{m\in\mathcal{C}(n)\\ z_m>0}}{\text{Min}}\, z_m \\
B_n^1 &= \sum_{m\in\mathcal{A}^1(n)} c_m^p + \sum_{m\in\mathcal{A}^2(n)} c_m^c + \sum_{m\in\mathcal{A}^3(n)} c_m^s - c_n^p \\
B_n^2 &= \sum_{\substack{m\in\mathcal{C}(n)\\ z_m>0}} c_m^s - c_n^c
\end{aligned}$$

Given these notations, the primal algorithm is presented in Algorithm 3.

Algorithm 3 The Primal Algorithm for $\mathcal{RSCE}_{THREE}$

```
 1: set $x^*_n = y^*_n = 0$ and $z^*_n = d_n, \quad \forall n \in \mathcal{T}$.
 2: set $k = \text{Max}\{n \in \mathcal{T} : \bar{\mathcal{T}}(n) \neq \emptyset\}$.
 3: while $k \geq 1$
 4:     compute $\mathcal{A}^1(k), \mathcal{A}^2(k), \mathcal{A}^3(k)$, $\Delta^1_k$, $\Delta^2_k$, $B^1_k$ and $B^2_k$.
 5:     while $B^1_k \geq 0$ or $B^2_k \geq 0$
 6:         if $B^2_k > B^1_k$
 7:             $z^*_m = z^*_m - \Delta^2_k, \quad \forall m \in \mathcal{C}(k)$ and $z^*_m > 0$
 8:             $y^*_k = y^*_k + \Delta^2_k$
 9:         else
10:             $z^*_m = z^*_m - \Delta^1_k, \quad \forall m \in \mathcal{A}^3(k)$
11:             $y^*_m = y^*_m - \Delta^1_k, \quad \forall m \in \mathcal{A}^2(k)$
12:             $x^*_m = \begin{cases} x^*_m + \Delta^1_k & \text{if } m = k, \\ x^*_m - \Delta^1_k & \text{if } m \in \mathcal{A}^1(k). \end{cases}$
13:         end if
14:         update $\mathcal{A}^1(k), \mathcal{A}^2(k), \mathcal{A}^3(k)$, $\Delta^1_k$, $\Delta^2_k$, $B^1_k$ and $B^2_k$.
15:     end while
16:     $k = k - 1$
17: end while
18: return $x^*$, $y^*$, $z^*$.
```

3.3.2 The Dual Algorithm

In this section, we consider the dual of the stochastic single resource capacity expansion model $\mathcal{RSCE}_{THREE}$, which can be formulated as follows:

Model $\mathcal{DRSCE}_{THREE}$:

$$\begin{aligned} \text{Max} \quad & \sum_{n \in \mathcal{T}} d_n \pi_n \\ \text{s.t.} \quad & \sum_{m \in \bar{\mathcal{T}}(n)} \pi_m \leq c^p_n \quad \forall n \in \mathcal{T}, \\ & \sum_{m \in \mathcal{C}(n)} \pi_m \leq c^c_n \quad \forall n \in \mathcal{T}, \\ & \pi_n \leq c^s_n \quad \forall n \in \mathcal{T}, \\ & \pi_n \in \mathbb{R}^+ \quad \forall n \in \mathcal{T}. \end{aligned} \tag{3.10}$$

We present a polynomial-time dual algorithm to solve (3.10). The algorithm has a greedy nature and is based on the following observation.

Lemma 3.7. *A solution $\pi = (\pi_1, \pi_2, \cdots, \pi_N)$ is dual feasible for (3.10), if and only if $\pi_n \geq 0$ and:*

$$\pi_n \leq \operatorname{Min}\left\{ \operatorname*{Min}_{m \in \bar{\mathcal{P}}(n)} \left\{ c^p_m - \sum_{k \in \bar{\mathcal{T}}(m) \setminus \{n\}} \pi_k \right\}, c^c_{a(n)} - \sum_{m \in \mathcal{C}(a(n)) \setminus \{n\}} \pi_m, c^s_n \right\}. \tag{3.11}$$

When π is optimal, the second inequality is tight.

Proof. Non-negativity of π_n and $\pi_n \leq c^s_n$ are guaranteed by forth and third constraints of dual problem (3.10), respectively. Also, note that for each n in first constraint, all constraints in which π_n will appear are corresponding to $\bar{\mathcal{P}}(n)$. Therefore, for all $m \in \bar{\mathcal{P}}(n)$, we have $\sum_{k \in \bar{\mathcal{T}}(m)} \pi_k \leq c^p_m$ or $\pi_n \leq c^p_m - \sum_{k \in \bar{\mathcal{T}}(m) \setminus \{n\}} \pi_k$, which means $\pi_n \leq \operatorname{Min}_{m \in \bar{\mathcal{P}}(n)} \left\{ c^p_m - \sum_{k \in \bar{\mathcal{T}}(m) \setminus \{n\}} \pi_k \right\}$. Now consider the second constraint. For each n, π_n will appear in the constraint for $a(n)$. Therefore, $\sum_{m \in \mathcal{C}(a(n))} \pi_m \leq c^c_{a(n)}$ or $\pi_n \leq c^c_{a(n)} - \sum_{m \in \mathcal{C}(a(n)) \setminus \{n\}} \pi_m$. To prove the last claim, suppose that in the optimal solution, the second inequality is not tight. Then we can always increase π_n to get a new feasible solution with greater objective value, which is a contradiction. □

Given the above observation and using the dual indexing system, the dual algorithm is given in Algorithm 4.

3.3.3 Validity of Algorithms

In this section, we verify that the primal and dual algorithms for $\mathcal{RSCE}_{THREE}$ return the primal and dual optimal solutions, respectively. In the following, we assume that

Algorithm 4 The Dual Algorithm for $\mathcal{DRSCE}_{THREE}$

```
1:  set $\pi^*_n = 0, \quad \forall n \in \mathcal{T}$.
2:  set $cp^0_n = c^p_n,\ cc^0_n = c^c_n, \quad \forall n \in \mathcal{T}$.
3:  for $k = 1, 2, \cdots, N$
4:      If $\text{Min}_{m\in\bar{\mathcal{P}}(k)} \left\{cp^{k-1}_m\right\} > c^s_k$ and $cc^{k-1}_{a(k)} > c^s_k$
5:          $\pi^*_k = c^s_k$
6:      else if $\text{Min}_{m\in\bar{\mathcal{P}}(k)} \left\{cp^{k-1}_m\right\} > cc^{k-1}_{a(k)}$
7:          $\pi^*_k = cc^{k-1}_{a(k)}$
8:      else
9:          $\pi^*_k = \text{Min}_{m\in\bar{\mathcal{P}}(k)} \left\{cp^{k-1}_m\right\}$
10:     end if
11:     $cp^k_n = \begin{cases} cp^{k-1}_n - \pi^*_k & \text{if } n \in \bar{\mathcal{P}}(k), \\ cp^{k-1}_n & \text{otherwise .} \end{cases}$
12:     $cc^k_n = \begin{cases} cc^{k-1}_n - \pi^*_k & \text{if } n = a(k) \text{ and } \pi^*_k \neq \text{Min}_{m\in\bar{\mathcal{P}}(k)} \left\{cp^{k-1}_m\right\}, \\ cc^{k-1}_n & \text{otherwise .} \end{cases}$
13: end for
14: return $\pi^* = (\pi^*_1, \cdots, \pi^*_N)$.
```

d_n, c^s_n, and c^p_n are positive numbers for all $n \in \mathcal{T}$, and $d_1 > d_2 > \cdots > d_N$. We also define m_k for each node k as: $m_k = \text{argmin}_{n\in\bar{\mathcal{P}}(k)} \left\{cp^{k-1}_n\right\}$. When $cp^{k-1}_n = cp^{k-1}_m$, we let $m_k = m$ if $t_m < t_n$.

We first show some properties of the primal solutions produced by Algorithm 3.

Lemma 3.8. *In the primal algorithm:*

(a) If $x^*_n > 0$, we can find $k \in \bar{\mathcal{T}}(n)$ such that $\sum_{m\in\mathcal{P}(n)} x^*_m = d_k$;

(b) If $y^*_n > 0$, we can find $k \in \mathcal{C}(n)$ such that $\sum_{m\in\mathcal{P}(n)} x^*_m + y^*_n = d_k$.

Proof. Note that the scenario tree has T periods. We can use mathematical induction from period T to 1 to prove (a) and (b) simultaneously. Consider the steps in Algorithm 3. The initial solution requires that $x^*_n = y^*_n = 0$ and $z^*_n = d_n$ for all $n \in \mathcal{T}$. Clearly, the statements hold for the initial solution. For a non-leaf node k in period

$T-1$, if we shift up capacity from nodes in $\mathcal{A}(k)$ to k as permanent capacity, then $x_k^* = \Delta_k^1 > 0$. If we shift up capacity from nodes in $\mathcal{A}(k)$ to k as contract capacity, then $y_k^* = \Delta_k^2 > 0$. Note that $\mathcal{A}(k) = \mathcal{C}(k)$. In either case, x_k^* or y_k^* will be the minimum demand among all nodes in $\mathcal{C}(k)$, and the statements hold. If it is still more economical to shift up capacity from the rest of nodes in $\mathcal{C}(k)$ to node k, the next value of x_k^* or y_k^* will be equal to the second smallest demand in $\mathcal{C}(k)$. When this procedure stops for node k, the statements hold.

Now consider any non-leaf node k in any period $t < T-1$. Consider two cases. Firstly, $B_k^1 \geq B_k^2$ and $B_k^1 \geq 0$. According to the definition, Δ_k^1 takes the minimum value among permanent capacities or contract capacities or spot market capacities of certain descendants of k, the statements hold according to mathematical induction assumption. Secondly, $B_k^2 > B_k^1$ and $B_k^2 \geq 0$. Then according to the definition, Δ_k^2 takes the minimum value among spot market capacities of direct descendants (sons) of k. It will be equal to the demand of a future node due to mathematical induction assumption. Therefore the statements hold for the whole scenario tree. $\square$

In Lemma 3.9 and Proposition 1, we show some properties of the dual solutions produced by Algorithm 4. To facilitate our proofs, we define sets $\mathcal{U}_i^k$, $\mathcal{V}_i^k$, and $\mathcal{W}_i^k$ for node i at iteration k in the dual algorithm:

$$\mathcal{U}_i^k = \left\{ m_j \;\middle|\; \begin{array}{c} m_j \in \mathcal{T}(i), j < k,\ \pi_j^* = cp_{m_j}^{j-1} \\ \text{and there does not exist } l < k \text{ and } l > j \text{ such that} \\ \pi_l^* = cp_{m_l}^{l-1},\ m_l \in \mathcal{T}(i),\ \text{and } \mathcal{T}(m_l) \supset \mathcal{T}(m_j) \end{array} \right\},$$

$$\mathcal{V}_i^k = \left\{ a(j) \;\middle|\; \begin{array}{c} a(j) \in \mathcal{T}(i),\ j < k,\ \pi_j^* = cc_{a(j)}^{j-1} \\ \text{and there does not exist } l < k \text{ and } l > j \text{ such that} \\ \pi_l^* = cp_{m_l}^{l-1},\ m_l \in \mathcal{T}(i),\ \text{and } a(j) \in \mathcal{T}(m_l) \end{array} \right\},$$

$$\mathcal{W}_i^k = \left\{ j \;\middle|\; \begin{array}{c} j \in \bar{\mathcal{T}}(i),\ j < k,\ \pi_j^* = c_j^s \\ \text{and there does not exist } l < k \text{ and } l > j \text{ such that} \\ \pi_l^* = cp_{m_l}^{l-1},\ m_l \in \mathcal{T}(i),\ \text{and } j \in \mathcal{T}(m_l), \text{or} \\ \pi_l^* = cc_{a(l)}^{l-1},\ a(l) \in \mathcal{T}(i),\ \text{and } j \in \mathcal{C}(a(l)) \end{array} \right\}.$$

These three sets correspond to all the capacities that could not be shifted up to node m_k in the primal algorithm.

Lemma 3.9. *In the dual algorithm, the following results hold:*

(a) If $\pi_k^* = cp_{m_k}^{k-1}$, then $\sum_{m \in \bar{\mathcal{T}}(m_k)} \pi_m^* = c_{m_k}^p$, and $\pi_i^* = 0$, $\forall i \in \bar{\mathcal{T}}(m_k)$ with $i > k$.

(b) If $\pi_k^* = cp_{m_k}^{k-1}$, then $\sum_{m \in \bar{\mathcal{T}}(i), m<k} \pi_m^* < c_i^p$ for all $i \in \bar{\mathcal{P}}(m_k)$.

(c) If $\pi_k^* = cc_{a(k)}^{k-1}$, then $\sum_{m \in \mathcal{C}(a(k))} \pi_m^* = c_{a(k)}^c$, and $\pi_i^* = 0$, $\forall i \in \mathcal{C}(a(k))$ with $i > k$.

(d) If $\pi_k^* = cc_{a(k)}^{k-1}$, then $\sum_{m \in \bar{\mathcal{T}}(i), m<k} \pi_m^* < c_i^p$, $\forall i \in \mathcal{P}(a(k))$.

(e) If $\pi_k^* = c_k^s$, then $\sum_{m \in \bar{\mathcal{T}}(i), m<k} \pi_m^* < c_i^p$, $\forall i \in \bar{\mathcal{P}}(k)$, and $\sum_{m \in \mathcal{C}(a(k))} \pi_m^* < c_{a(k)}^c$.

Proof. (a) Let us define $\mathcal{U}$, $\mathcal{V}$, $\mathcal{W}$, and $\mathcal{X}$ as follows:

$$\begin{aligned} \mathcal{U} &= \left\{ i \ :\ i \in \bar{\mathcal{T}}(m_k),\ i < k,\ \pi_i^* = cp_{m_i}^{i-1} \right\} \\ \mathcal{V} &= \left\{ i \ :\ i \in \bar{\mathcal{T}}(m_k),\ i < k,\ \pi_i^* = cc_{a(i)}^{i-1} \right\} \\ \mathcal{W} &= \left\{ i \ :\ i \in \bar{\mathcal{T}}(m_k),\ i < k,\ \pi_i^* = c_i^s \right\} \\ \mathcal{X} &= \left\{ i \ :\ i \in \bar{\mathcal{T}}(m_k),\ i > k \right\} \end{aligned}$$

According to the step 11 of Algorithm 4, when we consider node k, we have $\pi_k^* = cp_{m_k}^{k-1} = c_{m_k}^p - \sum_{j\in\bar{\mathcal{T}}(m_k),j<k}\pi_j^* = c_{m_k}^p - \sum_{i\in\mathcal{U}}\pi_i^* - \sum_{i\in\mathcal{V}}\pi_i^* - \sum_{i\in\mathcal{W}}\pi_i^*$. Consider the dual constraint: $\sum_{m\in\bar{\mathcal{T}}(m_k)}\pi_m^* = \pi_k^* + \sum_{i\in\mathcal{U}}\pi_i^* + \sum_{i\in\mathcal{V}}\pi_i^* + \sum_{i\in\mathcal{W}}\pi_i^* + \sum_{i\in\mathcal{X}}\pi_i^* = c_{m_k}^p + \sum_{i\in\mathcal{X}}\pi_i^* \le c_{m_k}^p$. This implies that $\sum_{i\in\mathcal{X}}\pi_i^* = 0$ and since $\pi_i^* \ge 0$ for all i, we must have $\pi_i^* = 0$ for each node $i \in \mathcal{X}$. This also shows that $\sum_{m\in\bar{\mathcal{T}}(m_k)}\pi_m^* = c_{m_k}^p$ which completes the proof of (a).

(b) Suppose $i \in \bar{\mathcal{P}}(m_k)$. According to Algorithm 4 , when we consider node k, we have $cp_i^{k-1} = c_i^p - \sum_{m\in\bar{\mathcal{T}}(i),m<k}\pi_m^*$, i.e., $\sum_{m\in\bar{\mathcal{T}}(i),m<k}\pi_m^* = c_i^p - cp_i^{k-1}$. Since $\pi_k^* = cp_{m_k}^{k-1}$, based on the definition of m_k, we have $cp_i^{k-1} > cp_{m_k}^{k-1} \ge 0$. Therefore $\sum_{m\in\bar{\mathcal{T}}(i),m<k}\pi_m^* < c_i^p$.

(c) Define $\mathcal{U}$, $\mathcal{V}$, $\mathcal{W}$, and $\mathcal{X}$ similarly as in (a), except that we replace $\bar{\mathcal{T}}(m_k)$ with $\mathcal{C}(a(k))$. Note that for node k, we have $\pi_k^* = cc_{a(k)}^{k-1} = c_{a(k)}^c - \sum_{i\in\mathcal{U}}\pi_i^* - \sum_{i\in\mathcal{V}}\pi_i^* - \sum_{i\in\mathcal{W}}\pi_i^*$. Consider the dual constraint: $\sum_{m\in\mathcal{C}(a(k))}\pi_m^* = \pi_k^* + \sum_{i\in\mathcal{U}}\pi_i^* + \sum_{i\in\mathcal{V}}\pi_i^* + \sum_{i\in\mathcal{W}}\pi_i^* + \sum_{i\in\mathcal{X}}\pi_i^* \le c_{a(k)}^c$. This implies that $\sum_{i\in\mathcal{X}}\pi_i^* = 0$, i.e., $\pi_i^* = 0$ for each node $i \in \mathcal{X}$. This also shows that $\sum_{m\in\mathcal{C}(a(k))}\pi_m^* = c_{a(k)}^c$ which completes the proof of (c).

(d) Suppose $i \in \mathcal{P}(a(k))$. According to Algorithm 4 , when we consider node k, we have $cp_i^{k-1} = c_i^p - \sum_{m\in\bar{\mathcal{T}}(i),m<k}\pi_m^*$, i.e., $\sum_{m\in\bar{\mathcal{T}}(i),m<k}\pi_m^* = c_i^p - cp_i^{k-1}$. Since $\pi_k^* = cc_{a(k)}^{k-1}$, we have $cp_i^{k-1} > cc_{a(k)}^{k-1} \ge 0$. Therefore $\sum_{m\in\bar{\mathcal{T}}(i),m<k}\pi_m^* < c_i^p$.

(e) According to Algorithm 4, for any node $i \in \bar{\mathcal{P}}(k)$, we have $cp_i^{k-1} = c_i^p - \sum_{m\in\bar{\mathcal{T}}(i),m<k}\pi_m^*$, i.e., $\sum_{m\in\bar{\mathcal{T}}(i),m<k}\pi_m^* = c_i^p - cp_i^{k-1}$. If $cp_i^{k-1} = 0$, it requires that $\pi_k^* = 0$ which is a contradiction of $\pi_k^* = c_k^s > 0$. So $\sum_{m\in\bar{\mathcal{T}}(i),m<k}\pi_m^* < c_i^p$. For node $a(k)$, we have $cc_{a(k)}^{k-1} = c_{a(k)}^c - \sum_{m\in\mathcal{C}(a(k)),m<k}\pi_m^*$. Note that $cc_{a(k)}^{k-1} > c_k^s \ge 0$, so that $\sum_{m\in\mathcal{C}(a(k)),m<k}\pi_m^* < c_{a(k)}^c$. □

Proposition 1. *In the dual algorithm, the following results hold:*

(a) $cp^{k-1}_{m_k} = c^p_{m_k} - \sum_{m_j \in \mathcal{U}^k_{m_k}} c^p_{m_j} - \sum_{a(j)\in\mathcal{V}^k_{m_k}} c^c_{a(j)} - \sum_{j\in\mathcal{W}^k_{m_k}} c^s_j$.

(b) If there exists $j \in \mathcal{C}(a(k))$ *such that* $j < k$ *and* $\pi^*_j = cc^{j-1}_{a(j)}$*, then* $cc^{k-1}_{a(k)} = 0$*; otherwise* $cc^{k-1}_{a(k)} = c^c_{a(k)} - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi^*_j = c^s_j} c^s_j$.

Proof. (a) Consider node i such that there does not exist $j < k$ and $\pi^*_j = cp^{j-1}_{m_j}$ and $i \in \bar{\mathcal{T}}(m_j)$. Note that for any $j < k$, if $j \in \bar{\mathcal{T}}(i)$, then either $j \in \mathcal{W}^k_i$, or $a(j) \in \mathcal{V}^k_i$, or $m_j \in \mathcal{U}^k_i$, or there exists $j < l < k$ such that $\pi^*_l = cp^{l-1}_{m_l}$ and $m_l \in \mathcal{T}(i)$ and $j \in \bar{\mathcal{T}}(m_l)$, or there exists $j < l < k$ such that $\pi^*_l = cc^{l-1}_{a(l)}$ and $a(l) \in \mathcal{T}(i)$ and $j \in \mathcal{C}(a(l))$. Therefore, for node i, we have

$$cp^{k-1}_i = c^p_i - \sum_{j\in\bar{\mathcal{T}}(i), j<k} \pi^*_j = c^p_i - \sum_{m_j\in\mathcal{U}^k_i} c^p_{m_j} - \sum_{a(j)\in\mathcal{V}^k_i} c^c_{a(j)} - \sum_{j\in\mathcal{W}^k_i} c^s_j$$

The conclusion holds when $i = m_k$.

(b) Note that $cc^{k-1}_i = c^c_i - \sum_{j\in\mathcal{C}(i), j<k} \pi^*_j$ and Algorithm 4 will update $cc^k_{a(k)}$ only if $\pi^*_k = c^s_k$ or $\pi^*_k = cc^{k-1}_{a(k)}$. If there exists a node $j \in \mathcal{C}(a(k))$ such that $j < k$ and $\pi^*_j = cc^{j-1}_{a(j)}$, then $cc^j_{a(k)} = cc^{j-1}_{a(j)} - \pi^*_j = 0$. This means that for all nodes $k > j$, $cc^{k-1}_{a(k)} = 0$. If for all $j \in \mathcal{C}(a(k))$ with $j < k$, we have $\pi^*_j = c^s_j$, then $cc^{k-1}_{a(k)} = c^c_{a(k)} - \sum_{j\in\mathcal{C}(a(k)), j<k} \pi^*_j = c^c_{a(k)} - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi^*_j=c^s_j} c^s_j$. Therefore the conclusion follows. □

We link the primal solutions and dual solutions in Lemma 3.10, 3.11 and 3.12, corresponding to the different sources of capacity in the dual solutions.

Lemma 3.10. *In the dual algorithm, if* $\pi^*_k = cp^{k-1}_{m_k}$*, then:*

(a) $y^*_{a(k)} = z^*_k = 0$ *and* $x^*_{m_k} > 0$ *;*

(b) $\sum_{m\in\mathcal{P}(m_k)} x^*_m \geq d_k$ *;*

(c) For all $i \in \bar{\mathcal{P}}(m_k)$, $\sum_{m\in\mathcal{P}(i)} x^*_m < d_k$.

Proof. (a) Since $\pi^*_k = cp^{k-1}_{m_k}$, it is required that $cp^{k-1}_{m_k} \leq cc^{k-1}_{a(k)}$. Also, Proposition 1(a) shows that $cp^{k-1}_{m_k} = c^p_{m_k} - \sum_{m_j\in\mathcal{U}^k_{m_k}} c^p_{m_j} - \sum_{a(j)\in\mathcal{V}^k_{m_k}} c^c_{a(j)} - \sum_{j\in\mathcal{W}^k_{m_k}} c^s_j$. If in Proposition 1(b), we have $cc^{k-1}_{a(k)} = c^c_{a(k)} - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi^*_j=c^s_j} c^s_j$, then it holds that $c^p_{m_k} - \sum_{m_j\in\mathcal{U}^k_{m_k}} c^p_{m_j} - \sum_{a(j)\in\mathcal{V}^k_{m_k}} c^c_{a(j)} - \sum_{j\in\mathcal{W}^k_{m_k}} c^s_j \leq c^c_{a(k)} - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi^*_j=c^s_j} c^s_j$, which implies that $c^p_{m_k} \leq c^c_{a(k)} + \sum_{m_j\in\mathcal{U}^k_{m_k}} c^p_{m_j} + \sum_{a(j)\in\mathcal{V}^k_{m_k}} c^c_{a(j)} + \sum_{j\in\mathcal{W}^k_{m_k}} c^s_j - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi^*_j=c^s_j} c^s_j$. This means that in Algorithm 3, the contract capacity of node $a(k)$ (if any) will be totally shifted up to node m_k, i.e., $y^*_{a(k)} = 0$. In Proposition 1(b), if we have $cc^{k-1}_{a(k)} = 0$, then it requires that $\pi^*_k = cp^{k-1}_{m_k} = 0 = cc^{k-1}_{a(k)}$, which also implies the contract capacity of node $a(k)$ will be totally shifted up to node m_k. On the other hand, $\pi^*_k = cp^{k-1}_{m_k}$ also requires that $cp^{k-1}_{m_k} \leq c^s_k$ which means that $c^p_{m_k} \leq c^s_k + \sum_{m_j\in\mathcal{U}^k_{m_k}} c^p_{m_j} + \sum_{a(j)\in\mathcal{V}^k_{m_k}} c^c_{a(j)} + \sum_{j\in\mathcal{W}^k_{m_k}} c^s_j$. This means that Algorithm 3 will completely shift up the spot market capacity of node k (if any) to m_k, i.e., $z^*_k = 0$.

Next, we need to ensure that the capacity accumulated at node m_k will not be shifted up completely, i.e., $x^*_{m_k} > 0$. Note that for all $i \in \bar{\mathcal{P}}(m_k)$, $cp^{k-1}_i = c^p_i - \sum_{m_j\in\mathcal{U}^k_i} c^p_{m_j} - \sum_{a(j)\in\mathcal{V}^k_i} c^c_{a(j)} - \sum_{j\in\mathcal{W}^k_i} c^s_j$. Since $cp^{k-1}_{m_k} < cp^{k-1}_i$, we have $c^p_{m_k} - \sum_{m_j\in\mathcal{U}^k_{m_k}} c^p_{m_j} - \sum_{a(j)\in\mathcal{V}^k_{m_k}} c^c_{a(j)} - \sum_{j\in\mathcal{W}^k_{m_k}} c^s_j < c^p_i - \sum_{m_j\in\mathcal{U}^k_i} c^p_{m_j} - \sum_{a(j)\in\mathcal{V}^k_i} c^c_{a(j)} - \sum_{j\in\mathcal{W}^k_i} c^s_j$, which implies that $c^p_i > c^p_{m_k} + \sum_{m_j\in\mathcal{U}^k_i\setminus\mathcal{U}^k_{m_k}} c^p_{m_j} + \sum_{a(j)\in\mathcal{V}^k_i\setminus\mathcal{V}^k_{m_k}} c^c_{a(j)} + \sum_{j\in\mathcal{W}^k_i\setminus\mathcal{W}^k_{m_k}} c^s_j$ for all $i \in \bar{\mathcal{P}}(m_k)$. According to Algorithm 3, this means that the capacity in m_k cannot be shifted up completely anymore. So $x^*_{m_k} > 0$.

(b) Suppose that $\sum_{m\in\mathcal{P}(m_k)} x^*_m < d_k$. Since $y^*_{a(k)} = z^*_k = 0$, there is a node in $j \in \bar{\mathcal{P}}(k) \setminus \mathcal{P}(m_k)$ with $x^*_j > 0$. However, in (a), we have proved that Algorithm 3 will shift up any capacity of nodes in $\bar{\mathcal{P}}(k)\setminus\mathcal{P}(m_k)$ to node m_k. So, for all $j \in \bar{\mathcal{P}}(k)\setminus\mathcal{P}(m_k)$, x^*_j must be equal to zero, which is a contradiction.

(c) Assume that there exists $i \in \bar{\mathcal{P}}(m_k)$ such that $\sum_{m\in\mathcal{P}(i)} x_m^* \geq d_k$. Without loss of generality, assume that i is such a node with the largest t_i. Then in the primal algorithm, we must have $c_i^p \leq \sum_{m_j\in\mathcal{U}_i^k} c_{m_j}^p + \sum_{a(j)\in\mathcal{V}_i^k} c_{a(j)}^c + \sum_{j\in\mathcal{W}_i^k} c_j^s$. This implies that node $i = m_k$, which is a contradiction. Therefore, the conclusion holds. □

Lemma 3.11. *In the dual algorithm, if* $\pi_k^* = cc_{a(k)}^{k-1}$*, then:*

(a) $y_{a(k)}^* > 0$ *and* $z_k^* = 0$ *;*

(b) $\sum_{m\in\bar{\mathcal{P}}(k)} x_m^* < d_k$ *;*

(c) $\sum_{m\in\bar{\mathcal{P}}(k)} x_m^* + y_{a(k)}^* \geq d_k$.

Proof. (a) Since $\pi_k^* = cc_{a(k)}^{k-1}$, $cc_{a(k)}^{k-1} \leq c_k^s$. If in Proposition 1(b), $cc_{a(k)}^{k-1} = c_{a(k)}^c - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi_j^*=c_j^s} c_j^s$, then we have $c_{a(k)}^c - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi_j^*=c_j^s} c_j^s < c_k^s$, i.e., $c_{a(k)}^c < c_k^s + \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi_j^*=c_j^s} c_j^s$. This means that the spot market capacity of node k will be totally shifted up to node $a(k)$ as contract capacity, i.e., $z_k^* = 0$. In Proposition 1(b), if there is a node $j \in \mathcal{C}(a(k))$ such that $j < k$ and $\pi_j^* = cc_{a(j)}^{j-1}$, then $cc_{a(k)}^{k-1} = 0$. Note that $cc_{a(j)}^{j-1} \leq c_j^s$. This implies that the spot market capacity of nodes in $\mathcal{C}(a(j))$ that is no less than d_j will be shifted up to node $a(j) = a(k)$. Since $d_j > d_k$, $z_k^* = 0$.

On the other hand, since $\pi_k^* = cc_{a(k)}^{k-1}$, $cc_{a(k)}^{k-1} < cp_{m_k}^{k-1}$, which implies that for all node $i \in \bar{\mathcal{P}}(k)$, we have $cc_{a(k)}^{k-1} < cp_i^{k-1}$. According to Proposition 1(a), this can be written as $cc_{a(k)}^{k-1} < c_i^p - \sum_{m_j\in\mathcal{U}_i^k} c_{m_j}^p - \sum_{a(j)\in\mathcal{V}_i^k} c_{a(j)}^c - \sum_{j\in\mathcal{W}_i^k} c_j^s$. In Proposition 1(b), if $cc_{a(k)}^{k-1} = c_{a(k)}^c - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi_j^*=c_j^s} c_j^s$, then $c_{a(k)}^c - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi_j^*=c_j^s} c_j^s < c_i^p - \sum_{m_j\in\mathcal{U}_i^k} c_{m_j}^p - \sum_{a(j)\in\mathcal{V}_i^k} c_{a(j)}^c - \sum_{j\in\mathcal{W}_i^k} c_j^s$, i.e., $c_i^p > c_{a(k)}^c + \sum_{m_j\in\mathcal{U}_i^k} c_{m_j}^p + \sum_{a(j)\in\mathcal{V}_i^k} c_{a(j)}^c + \sum_{j\in\mathcal{W}_i^k} c_j^s - \sum_{j\in\mathcal{C}(a(k)),\ j<k,\ \pi_j^*=c_j^s} c_j^s$. This implies that the contract capacity accumulated in node $a(k)$ will not be shifted up anymore, i.e., $y_{a(k)}^* > 0$. In Proposition 1(b), if $cc_{a(k)}^{k-1} = 0$, then $c_i^p > \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p + \sum_{a(j) \in \mathcal{V}_i^k} c_{a(j)}^c + \sum_{j \in \mathcal{W}_i^k} c_j^s$ which again implies that no contract capacity will be shifted up from node $a(k)$, i.e., $y_{a(k)}^* > 0$.

(b) Suppose $\sum_{m \in \bar{\mathcal{P}}(k)} x_m^* \geq d_k$. According to (a), $y_{a(k)}^* > 0$ and we can decrease $y_{a(k)}^*$ to make the constraint tight. This will give us a new feasible solution with decreasing objective value, which is a contradiction of optimality.

(c) The claim is an immediate result of (a) and primal feasibility. □

Lemma 3.12. *In the dual algorithm, if $\pi_k^* = c_k^s$, then:*

(a) $z_k^* > 0$;

(b) $\sum_{m \in \bar{\mathcal{P}}(k)} x_m^* + y_{a(k)}^* + z_k^* = d_k$, and $\sum_{m \in \bar{\mathcal{P}}(k)} x_m^* + y_{a(k)}^* < d_k$.

Proof. (a) Since $\pi_k^* = c_k^s$, we can conclude that $c_k^s < cc_{a(k)}^{k-1}$, and for all $i \in \bar{\mathcal{P}}(k)$, $c_k^s < cp_i^{k-1}$. Now, according to Proposition 1(a), $cp_i^{k-1} = c_i^p - \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p - \sum_{a(j) \in \mathcal{V}_i^k} c_{a(j)}^c - \sum_{j \in \mathcal{W}_i^k} c_j^s$. So, $c_k^s < c_i^p - \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p - \sum_{a(j) \in \mathcal{V}_i^k} c_{a(j)}^c - \sum_{j \in \mathcal{W}_i^k} c_j^s$, i.e., $c_i^p > c_k^s + \sum_{m_j \in \mathcal{U}_i^k} c_{m_j}^p + \sum_{a(j) \in \mathcal{V}_i^k} c_{a(j)}^c + \sum_{j \in \mathcal{W}_i^k} c_j^s$ which implies that no capacity will be shifted up from node k to nodes $i \in \bar{\mathcal{P}}(k)$ as permanent capacity according to Algorithm 3. In Proposition 1(b), if $cc_{a(k)}^{k-1} = c_{a(k)}^c - \sum_{j \in \mathcal{C}(a(k)),\ j<k,\ \pi_j^* = c_j^s} c_j^s$, then $c_k^s < c_{a(k)}^c - \sum_{j \in \mathcal{C}(a(k)),\ j<k,\ \pi_j^* = c_j^s} c_j^s$ or $c_{a(k)}^c > c_k^s + \sum_{j \in \mathcal{C}(a(k)),\ j<k,\ \pi_j^* = c_j^s} c_j^s$, which implies that no capacity will be shifted up from node k to $a(k)$ as contract capacity. Since the initialization of Algorithm 3 enforces all z_n to be positive, we can conclude that $z_k^* > 0$. Also, $cc_{a(k)}^{k-1}$ cannot be zero in Proposition 1(b), because then $c_k^s < cc_{a(k)}^{k-1} = 0$ will be a contradiction.

(b) Suppose $\sum_{m \in \bar{\mathcal{P}}(k)} x_m^* + y_{a(k)}^* + z_k^* > d_k$. Since $z_k^* > 0$, we can decrease z_k^* to make the constraint tight. This will give us a new feasible solution with less objective value, which is a contradiction of optimality. Then the second half of the claim follows. □

Theorem 3.6. *The solution (x^*, y^*, z^*) returned by the primal algorithm and the solution π^* returned by the dual algorithm are optimal.*

Proof. Firstly, (x^*, y^*, z^*) and π^* are feasible solutions of primal and dual problems respectively. Indeed, in the primal algorithm, we start with a feasible solution and in each iteration, we preserve feasibility. Therefore, (x^*, y^*, z^*) is feasible. The feasibility of π^* is guaranteed by Lemma 3.7.

Next, we only need to prove the complementary slackness for primal and dual solutions:

$$\pi_n^* > 0 \implies \sum_{m \in \bar{\mathcal{P}}(n)} x_m^* + y_{a(n)}^* + z_n^* = d_n \tag{3.12}$$

$$\sum_{m \in \bar{\mathcal{T}}(n)} \pi_m^* < c_n^p \implies x_n^* = 0 \tag{3.13}$$

$$\sum_{m \in \mathcal{C}(n)} \pi_m^* < c_n^c \implies y_n^* = 0 \tag{3.14}$$

$$\pi_n^* < c_n^s \implies z_n^* = 0 \tag{3.15}$$

Assume we are using the dual indexing system. Suppose that $\pi_n^* > 0$. To prove (3.12), three cases are considered:

Case 1: If $\pi_n^* = c_n^s$, Lemma 3.12(b) guarantees that $\sum_{m \in \bar{\mathcal{P}}(n)} x_m^* + y_{a(n)}^* + z_n^* = d_n$.

Case 2: If $\pi_n^* = cc_{a(n)}^{n-1} > 0$, then we assume that $\sum_{m \in \bar{\mathcal{P}}(n)} x_m^* + y_{a(n)}^* + z_n^* > d_n$. According to Lemma 3.11(a), $z_n^* = 0$. So, $\sum_{m \in \bar{\mathcal{P}}(n)} x_m^* + y_{a(n)}^* > d_n$. Since $y_{a(n)}^* > 0$, according to Lemma 3.8(b), there is a node $k \in \mathcal{C}(a(n))$ such that $\sum_{m \in \mathcal{P}(a(n))} x_m^* + y_{a(n)}^* = d_k$, or (since $\mathcal{P}(a(n)) = \bar{\mathcal{P}}(n)$), $\sum_{m \in \bar{\mathcal{P}}(n)} x_m^* + y_{a(n)}^* = d_k$, which implies that $d_k > d_n$ or $k < n$. Now consider the dual variable of node k, i.e., π_k^*. If $\pi_k^* = c_k^s$, according to Lemma 3.12(b), $\sum_{m \in \bar{\mathcal{P}}(k)} x_m^* + y_{a(k)} < d_k$, which is a contradiction with

$\sum_{m\in\bar{\mathcal{P}}(n)} x^*_m + y^*_{a(n)} = d_k$, since $\bar{\mathcal{P}}(n) = \bar{\mathcal{P}}(k)$. If $\pi^*_k = cc^{k-1}_{a(k)}$, then since $k < n$, Lemma 3.9(c) requires that $\pi^*_n = 0$ which is a contradiction. Finally, if $\pi^*_k = cp^{k-1}_{m_k}$, then $m_k \in \mathcal{P}(a(n))$, and according to Algorithm 4, $\pi^*_n = 0$, which is a contradiction.

Case 3: If $\pi^*_n = cp^{n-1}_{m_n} > 0$, then we assume that $\sum_{m\in\bar{\mathcal{P}}(n)} x^*_m + y^*_{a(n)} + z^*_n > d_n$. According to Lemma 3.10(a), $y^*_{a(n)} = z^*_n = 0$, and $x^*_{m_n} > 0$. Then according to Lemma 3.8(a), there is a node $k \in \bar{\mathcal{T}}(m_n)$ such that $\sum_{m\in\mathcal{P}(m_n)} x^*_m = d_k$, which implies that $d_k > d_n$ or $k < n$. Now consider the dual variable of node k, i.e., π^*_k. If $\pi^*_k = c^s_k$, according to Lemma 3.12(b), $\sum_{m\in\bar{\mathcal{P}}(k)} x^*_m + y_{a(k)} < d_k$ or $\sum_{m\in\bar{\mathcal{P}}(k)} x^*_m < d_k$, which is a contradiction with $\sum_{m\in\mathcal{P}(m_n)} x^*_m = d_k$, since $m_n \in \bar{\mathcal{P}}(k)$. If $\pi^*_k = cc^{k-1}_{a(k)}$, Lemma 3.11(b) requires that $\sum_{m\in\bar{\mathcal{P}}(k)} x^*_m < d_k$, which is a contradiction with $\sum_{m\in\mathcal{P}(m_n)} x^*_m = d_k$. Finally, if $\pi^*_k = cp^{k-1}_{m_k}$, when $m_k \in \bar{\mathcal{T}}(m_n)$, Lemma 3.10(c) requires that $\sum_{m\in\mathcal{P}(m_n)} x^*_m < d_k$ which is a contradiction with $\sum_{m\in\mathcal{P}(m_n)} x^*_m = d_k$; when $m_k \in \mathcal{P}(m_n)$, since $k < n$ and $n \in \bar{\mathcal{T}}(m_k)$, Lemma 3.9(a) implies that $\pi^*_n = 0$ which is a contradiction. Therefore, (3.12) holds.

To prove (3.13), we also use contradiction. Assume that $\sum_{m\in\bar{\mathcal{T}}(n)} \pi^*_m < c^p_n$ and $x^*_n > 0$ simultaneously. According to Lemma 3.8(a), we can find a $k \in \bar{\mathcal{T}}(n)$, such that $\sum_{m\in\mathcal{P}(n)} x^*_m = d_k$. Now, consider the optimal dual value of k, i.e., π^*_k:

Case 1: $\pi^*_k = c^s_k$. According to Lemma 3.12(b), $\sum_{m\in\bar{\mathcal{P}}(k)} x^*_m + y^*_{a(k)} < d_k$ which is a contradiction with $\sum_{m\in\mathcal{P}(n)} x^*_m = d_k$, since $k \in \bar{\mathcal{T}}(n)$.

Case 2: $\pi^*_k = cc^{k-1}_{a(k)}$. According to Lemma 3.11(b), $\sum_{m\in\bar{\mathcal{P}}(k)} x^*_m < d_k$ and since $a(k) \in \mathcal{T}(n)$, it is a contradiction with $\sum_{m\in\mathcal{P}(n)} x^*_m = d_k$.

Case 3: $\pi^*_k = cp^{k-1}_{m_k}$. If $m_k \in \bar{\mathcal{T}}(n)$, Lemma 3.10(c) implies that $\sum_{m\in\mathcal{P}(i)} x^*_m < d_k$, $\forall i \in \bar{\mathcal{P}}(m_k)$. Since $n \in \bar{\mathcal{P}}(m_k)$, we must have $\sum_{m\in\mathcal{P}(n)} x^*_m < d_k$, which is a contradiction with $\sum_{m\in\mathcal{P}(n)} x^*_m = d_k$. If $m_k \in \bar{\mathcal{P}}(n)$, Lemma 3.10(b) shows that

$\sum_{m\in\mathcal{P}(m_k)} x^*_m \geq d_k$. Since $n \in \bar{\mathcal{T}}(m_k)$ and $\sum_{m\in\mathcal{P}(n)} x^*_m = d_k$, we can conclude that $x^*_n = 0$ which is a contradiction. The only possibility left is $m_k = n$. In this case, Lemma 3.9(a) requires that $\sum_{m\in\bar{\mathcal{T}}(n)} \pi^*_m = c^p_n$ which is a contradiction with $\sum_{m\in\bar{\mathcal{T}}(n)} \pi^*_m < c^p_n$. Therefore, (3.13) holds.

To prove (3.14), we again use contradiction. Assume that $\sum_{m\in\mathcal{C}(n)} \pi^*_m < c^c_n$ and $y^*_n > 0$ simultaneously. According to Lemma 3.8(b), we can find a node $k \in \mathcal{C}(n)$ such that $\sum_{m\in\mathcal{P}(n)} x^*_m + y^*_{a(k)} = d_k$ (note that $y^*_{a(k)} = y^*_n$). Now consider the dual variable of node k, i.e., π^*_k:

Case 1: $\pi^*_k = c^s_k$. According to Lemma 3.12(b), $\sum_{m\in\bar{\mathcal{P}}(k)} x^*_m + y^*_{a(k)} < d_k$ which is a contradiction with $\sum_{m\in\mathcal{P}(n)} x^*_m + y^*_{a(k)} = d_k$, since $\bar{\mathcal{P}}(k) = \mathcal{P}(n)$.

Case 2: $\pi^*_k = cc^{k-1}_{a(k)}$. According to Lemma 3.9(c), $\sum_{m\in\mathcal{C}(a(k))} \pi^*_m = c^c_{a(k)}$. Since $a(k) = n$, it is a contradiction with $\sum_{m\in\mathcal{C}(n)} \pi_m < c^c_n$.

Case 3: $\pi^*_k = cp^{k-1}_{m_k}$. In this case, $m_k \in \mathcal{P}(n)$. Then, Lemma 3.10(b) implies that $\sum_{m\in\mathcal{P}(n)} x^*_m \geq d_k$. This means that $y^*_{a(k)} = y^*_n = 0$, which is a contradiction. Therefore, (3.14) holds.

Finally, when $\pi^*_n < c^s_n$, we have either $\pi^*_n = cp^{n-1}_{m_n}$ or $\pi^*_n = cc^{n-1}_{a(n)}$. Lemma 3.10(a) and Lemma 3.11(a) show that $z^*_n = 0$, respectively. Therefore, (3.15) holds. □

3.3.4 Complexity

Suppose the scenario tree is a complete tree with T time levels and B branches for every non-leaf node. Then the number of nodes in the scenario tree is $N = \sum_{t=0}^{T-1} B^t = \frac{B^T-1}{B-1}$, which implies that $T \sim \mathcal{O}(\log N)$.

Theorem 3.7. *The complexity of the primal algorithm is $\mathcal{O}(N^2)$.*

Proof. Initialization step needs at most N operations. Computing $\mathcal{A}(k)$, Δ^1_k, Δ^2_k,

B_k^1, and B_k^2 requires at most N operations. The loop in step 3 will run at most N times. Each iteration of the loop in step 5 will deplete the capacity for one node and the subsequent adjustments will be of complexity $\mathcal{O}(N)$. So the complexity of Algorithm 3 is $\mathcal{O}(N^2)$. □

Theorem 3.8. *The complexity of the dual algorithm is $\mathcal{O}(N \log N)$.*

Proof. Sorting the demands of N nodes has a complexity of $N \log N$. For each iteration, calculating π_k^* involves minimization of at most $T+1$ values; updating cp_n^k and cc_n^k requires at most T operations. Therefore, the total number of operations is at most $N \log N + 2NT + N$. Since $T \sim \mathcal{O}(\log N)$, the complexity of Algorithm 4 is $\mathcal{O}(N \log N)$. □

Surprisingly, if we assume the contracts can take arbitrary length, the complexity of the primal algorithm will keep the same, $\mathcal{O}(N^2)$. This is due to the fact that the size of $\mathcal{A}(k)$ is at most N no matter what are the lengths of the contracts. Similarly, if we assume the contracts can take arbitrary length, the complexity of the dual algorithm will keep the same, $\mathcal{O}(N \log N)$. This is due to the fact that there are at most $\mathcal{O}(N)$ operations in the loop of Algorithm 4.

3.3.5 Experimental Results

In this section, we compare the solution times of the primal and dual algorithm designed to solve $\mathcal{SCE}_{THREE}$ with those of CPLEX LP solver. We created 15 instances of $\mathcal{SCE}_{THREE}$ with the same sizes created for $\mathcal{SCE}_{TWO}$ in Section 3.2.5. We present the results in Table 3.2.

All the codes are written in C++ calling CPLEX 12.6. The experiments are conducted on a computer with an Intel Core i3-M330 2.13GHz processor and 4.00 GB of RAM running Ubuntu 14.04 LTS.

Table 3.4: Performance of the primal and dual algorithms for $\mathcal{SCE}_{THREE}$

T	B	N	Time (s)			% of CPLEX Time	
			CPLEX	Primal	Dual	Primal	Dual
5	5	781	0.015	0.003	0.001	19.56	7.55
8	3	$3,280$	0.048	0.011	0.005	22.20	9.59
12	2	$4,095$	0.058	0.024	0.006	41.97	9.51
5	10	$11,111$	0.108	0.034	0.011	31.44	10.08
7	5	$19,531$	0.255	0.047	0.035	18.26	13.76
15	2	$32,767$	0.657	0.147	0.054	22.42	8.16
8	5	$97,656$	0.938	0.309	0.143	32.92	15.25
10	4	$349,525$	6.502	1.217	0.576	18.72	8.85
9	5	$488,281$	7.345	1.454	0.786	19.80	10.69
13	3	$797,161$	22.721	2.771	1.858	12.19	8.17
20	2	$1,048,575$	50.323	6.050	2.287	12.02	4.54
7	10	$1,111,111$	52.387	6.232	2.376	11.89	4.53
8	8	$2,396,745$	72.733	7.241	4.069	9.95	5.59
12	4	$5,592,405$	391.762	21.490	12.925	5.48	3.29
15	3	$7,174,453$	$*$	25.133	16.329	N/A	N/A

* For the last instance, CPLEX LP solver encounters an out of memory exception.

Table 3.2 shows that both primal and dual algorithms can outperform CPLEX LP solver in all instances, especially for large-scale ones. Table 3.2 also confirms that the complexity of the dual algorithm is less than the complexity of the primal algorithm.

Chapter 4

Stochastic Network Capacity Expansion Models

4.1 Introduction

In this chapter, we consider multiple resource stochastic capacity expansion models in the context of network flow problems. All of the models that we considered in Chapter 3 were involved with only one single resource, whereas in this chapter, we consider network models that involve multiple resources. More specifically, we consider stochastic network capacity expansion models in the context of a general min-cost network flow problem.

Stochastic network capacity expansion problem can be modeled as a multi-stage stochastic integer problem. A scenario tree can be used to handle data uncertainty of the model. In this problem, the objective is to minimize the expected total capacity acquisition cost over the whole scenario tree. We start with a model that allows two sources of capacity (permanent and spot market) in Section 4.2 and then, we extend the result to the model that allows three sources of capacity (permanent, spot market, and contract) in Section 4.4

4.2 The Stochastic Network Capacity Expansion Model with Two Sources of Capacity

In this section, we consider a multi-period, multiple resource stochastic capacity expansion model in the context of min-cost network flow problem. This model can be used in a wide range of network applications, where the decision maker has the choice to obtain the capacities from different sources, and is interested in expanding the capacities of both nodes and arcs. In this section, we assume permanent and spot market capacities are available.

Consider an arbitrary network $(\mathcal{N}, \mathcal{A})$ with $|\mathcal{N}|$ nodes and $|\mathcal{A}|$ arcs. We will consider capacity expansion of $|\mathcal{N}| + |\mathcal{A}|$ resources over T periods. We assume that there are three types of nodes in the network: *origin*, *destination*, and *transient*. If i is an origin node, then its demand is negative; if i is a destination node, its demand is positive; otherwise, for transient nodes, the demand is zero.

In the model notations presented in Table 4.1, we use u and v to distinguish parameters and decision variables between arcs and nodes, respectively. In addition, note that index n refers to a node in the scenario tree $\mathcal{T}$. Furthermore, $\delta^+(i)$ is the set of arcs in $\mathcal{A}$ emanating from node i, and $\delta^-(i)$ is the set of arcs in $\mathcal{A}$ entering node i. Also, we assume that the demand in the network is balanced. This assumption requires that the summation of demand for all origin and destination nodes in the network is equal to zero, i.e., for all $n \in \mathcal{T}$, $\sum_{i:d_{ni}<0} d_{ni} + \sum_{i:d_{ni}>0} d_{ni} = 0$.

Table 4.1: Notations of the models $\mathcal{SNCE}_{TWO}$

Parameters:	
c^{up}_{nij}	Permanent capacity acquisition cost for arc $(i,j) \in \mathcal{A}$ at node $n \in \mathcal{T}$
c^{vp}_{ni}	Permanent capacity acquisition cost for node $i \in \mathcal{N}$ at node $n \in \mathcal{T}$
c^{us}_{nij}	Spot market capacity acquisition cost for arc $(i,j) \in \mathcal{A}$ at node $n \in \mathcal{T}$
c^{vs}_{ni}	Spot market capacity acquisition cost for node $i \in \mathcal{N}$ at node $n \in \mathcal{T}$
c^{u}_{ij}	Flow cost on arc $(i,j) \in \mathcal{A}$
C^{u}_{ij}	Unit capacity of arc $(i,j) \in \mathcal{A}$
C^{v}_{i}	Unit capacity of node $i \in \mathcal{N}$
d_{ni}	Random demand for node $i \in \mathcal{N}$ at node $n \in \mathcal{T}$
Decision variables:	
x_{nij}	Flow on arc $(i,j) \in \mathcal{A}$ at node $n \in \mathcal{T}$
u^{p}_{nij}	Permanent capacity acquisition for arc $(i,j) \in \mathcal{A}$ at node $n \in \mathcal{T}$
v^{p}_{ni}	Permanent capacity acquisition for node $i \in \mathcal{N}$ at node $n \in \mathcal{T}$
u^{s}_{nij}	Spot market capacity acquisition for arc $(i,j) \in \mathcal{A}$ at node $n \in \mathcal{T}$
v^{s}_{ni}	Spot market capacity acquisition for node $i \in \mathcal{N}$ at node $n \in \mathcal{T}$

The stochastic network capacity expansion problem with permanent and spot market capacity can be formulated as a mixed-integer programming (MIP) problem in (4.1)-(4.7). In this formulation, the objective function (4.1) minimizes the expected total flow costs and capacity acquisition costs related to nodes and arcs. Constraint (4.2) ensures that all demands are satisfied for all nodes in the scenario tree. Constraint (4.3) guarantees that the flow on each arc will not exceed the total permanent and spot market capacity acquired for that arc. Constraint (4.4) ensures that the outgoing flow of each node cannot exceed the total permanent and spot market capacity acquired for that node. Non-negativity and integrality constraints are enforced through (4.5) and (4.7).

Model $\mathcal{SNCE}_{TWO}$:

$$\text{Min} \sum_{n\in\mathcal{T}} p_n \left[\sum_{(i,j)\in\mathcal{A}} (c^u_{ij}x_{nij} + c^{up}_{nij}u^p_{nij} + c^{us}_{nij}u^s_{nij}) + \sum_{i\in\mathcal{N}} (c^{vp}_{ni}v^p_{ni} + c^{vs}_{ni}v^s_{ni}) \right] \tag{4.1}$$

$$\text{s.t.} \sum_{(j,i)\in\delta^-_i} x_{nji} - \sum_{(i,j)\in\delta^+_i} x_{nij} = d_{ni} \qquad \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \tag{4.2}$$

$$x_{nij} \le C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \right) \qquad \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \tag{4.3}$$

$$\sum_{(i,j)\in\delta^+_i} x_{nij} \le C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \right) \qquad \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \tag{4.4}$$

$$x_{nij} \in \mathbb{R}^+ \qquad \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \tag{4.5}$$

$$u^p_{nij},\ u^s_{nij} \in \mathbb{Z}^+ \qquad \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \tag{4.6}$$

$$v^p_{ni},\ v^s_{ni} \in \mathbb{Z}^+ \qquad \forall i \in \mathcal{N}, \forall n \in \mathcal{T}. \tag{4.7}$$

Theorem 4.1. *The stochastic program $\mathcal{SNCE}_{TWO}$ and its deterministic counterpart are NP-hard.*

Proof. We show that any instance of the NP-hard integer knapsack problem (Garey and Johnson (1979)) with $|\mathcal{N}|+|\mathcal{A}|$ items can be polynomially transformed to a single period instance of the deterministic network capacity expansion problem, which is just a single node scenario instance of the stochastic model $\mathcal{SNCE}_{TWO}$. Suppose the scenario tree has one node, so we can drop the index n in the model. Also, let u^p_{nij} and v^p_{ni} be zero. Now, from (4.2), we have $\sum_{(j,i)\in\delta^-_i} x_{ji} - d_i = \sum_{(i,j)\in\delta^+_i} x_{ij}$. If we insert this in (4.4), we will have $\sum_{(j,i)\in\delta^-_i} x_{ji} - d_i \le C^v_i v^s_i$, or $C^v_i v^s_i - \sum_{(j,i)\in\delta^-_i} x_{ji} \ge -d_i$. Also, according to (4.3), we have $x_{ij} \le C^u_{ij}u^s_{ij}$ or $\sum_{(i,j)\in\delta^-_i} x_{ij} \le \sum_{(i,j)\in\delta^-_i} C^u_{ij}u^s_{ij}$, or $-\sum_{(j,i)\in\delta^-_i} x_{ji} \le \sum_{(i,j)\in\delta^-_i} C^u_{ij}u^s_{ij}$. Therefore, $C^v_i v^s_i + \sum_{(i,j)\in\delta^-_i} C^u_{ij}u^s_{ij} \ge -d_i$. Now assume that in the network, there is only one origin (node 1) and all other nodes are destinations. So, for all nodes i other than node 1, we have $v_i^s = \lceil \frac{d_i}{C_i^v} \rceil$, where $\lceil x \rceil$ is the closest integer value greater than or equal to x; and the rest of the problem can be stated as follows:

$$\begin{aligned} \text{Min} \quad & \sum_{(1,j)\in\mathcal{A}} c_{1j}^{us} u_{1j}^s + c_1^{vs} v_1^s \\ \text{s.t.} \quad & C_1^v v_1^s + \sum_{(1,j)\in\delta_1^-} C_{1j}^u u_{1j}^s \geq -d_1 \\ & u_{1j}^s, v_1^s \in \mathbb{Z}^+, \qquad \forall j \in \mathcal{N}, \end{aligned}$$

which can be seen as the integer knapsack problem presented by Garey and Johnson (1979). Note that minimization knapsack problem can be transfered into an equivalent maximization knapsack problem in polynomial time (Han and Makino (2010)). □

Theorem 4.1 shows that the problem $\mathcal{SNCE}_{TWO}$ is intractable for large-scale instances. Therefore, we will develop a decomposition-based approximation algorithm in Section 4.3. This approach will decompose the original problem into subproblems of single resources, corresponding to the nodes and arcs of the network. The approximation algorithm returns near-optimal solutions, and can be proved to be asymptotically convergent to the optimal solution.

4.3 The Approximation Algorithm

In this section we develop an approximation algorithm for the problem $\mathcal{SNCE}_{TWO}$ which returns near optimal solutions, and can be proved to be asymptotically convergent.

The problem $\mathcal{SNCE}_{TWO}$ can be decomposed into smaller subproblems, which is the key to our approximation scheme. Ahmed and Sahinidis (2003) and Huang and Ahmed (2009) also explore decomposition structures to solve multi-stage stochastic capacity expansion problems. However, our method is different in that it includes multiple sources of capacity, which makes the analysis relevant to costs of multiple sources and different from previous works.

For simplicity of exposition, let $\mathcal{X} = (\mathbf{x}_n)_{n\in\mathcal{T}}$ where $\mathbf{x}_n = (x_{nij})_{(i,j)\in\mathcal{A}}$ and $\mathcal{Y} = (\mathbf{y}_n)_{n\in\mathcal{T}} = (\mathbf{u}^{\mathbf{p}}_n, \mathbf{v}^{\mathbf{p}}_n, \mathbf{u}^{\mathbf{s}}_n, \mathbf{v}^{\mathbf{s}}_n)_{n\in\mathcal{T}}$ where $\mathbf{u}^{\mathbf{p}}_n = (u^p_{nij})_{(i,j)\in\mathcal{A}}$, $\mathbf{v}^{\mathbf{p}}_n = (v^p_{ni})_{i\in\mathcal{N}}$, $\mathbf{u}^{\mathbf{s}}_n = (u^s_{nij})_{(i,j)\in\mathcal{A}}$, and $\mathbf{v}^{\mathbf{s}}_n = (v^s_{ni})_{i\in\mathcal{N}}$.

We start with an illustration of the decomposition structure. Note that the problem $\mathcal{SNCE}_{TWO}$ is equivalent to:

$$\begin{aligned}
\text{Min} \quad & \sum_{n\in\mathcal{T}} p_n \sum_{(i,j)\in\mathcal{A}} c^u_{ij} x_{nij} + \sum_{i\in\mathcal{N}} Q^1_i(\mathcal{X}) + \sum_{(i,j)\in\mathcal{A}} Q^2_{(i,j)}(\mathcal{X}) && \\
\text{s.t.} \quad & \sum_{(j,i)\in\delta^-_i} x_{nji} - \sum_{(i,j)\in\delta^+_i} x_{nij} = d_{ni} && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \qquad (4.8)\\
& x_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T},
\end{aligned}$$

where

$$\begin{aligned} Q_i^1(\mathcal{X}) = \text{Min} \quad & \sum_{n\in\mathcal{T}} p_n \left(c_{ni}^{vp} v_{ni}^p + c_{ni}^{vs} v_{ni}^s\right) \\ \text{s.t.} \quad & \sum_{m\in\bar{\mathcal{P}}(n)} v_{mi}^p + v_{ni}^s \geq \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}}{C_i^v} \qquad \forall n \in \mathcal{T}, \\ & v_{ni}^p, v_{ni}^s \in \mathbb{Z}^+ \qquad \forall n \in \mathcal{T}, \end{aligned} \tag{4.9}$$

and

$$\begin{aligned} Q_{(i,j)}^2(\mathcal{X}) = \text{Min} \quad & \sum_{n\in\mathcal{T}} p_n \left(c_{nij}^{up} u_{nij}^p + c_{nij}^{us} u_{nij}^s\right) \\ \text{s.t.} \quad & \sum_{m\in\bar{\mathcal{P}}(n)} u_{mij}^p + u_{nij}^s \geq \frac{x_{nij}}{C_{ij}^u} \qquad \forall n \in \mathcal{T}, \\ & u_{nij}^p, u_{nij}^s \in \mathbb{Z}^+ \qquad \forall n \in \mathcal{T}. \end{aligned} \tag{4.10}$$

The problem (4.8) involves the capacity allocation decisions, while (4.9) involves the capacity acquisition for nodes of the network, and (4.10) involves the capacity acquisition for arcs of the network. Note that for a fixed capacity allocation decision $\mathcal{X} = (\mathbf{x}_n)_{n\in\mathcal{T}}$, (4.9) and (4.10) can be seen as the single resource capacity expansion subproblems $\mathcal{SCE}_{TWO}$ introduced in Section 3.2. More specifically, if we let

$$\delta_n^1 = \left[\frac{\sum_{(i,j)\in\delta_i^+} x_{nij}}{C_i^v}\right]_i \qquad \text{and} \qquad \delta_n^2 = \left[\frac{x_{nij}}{C_{ij}^u}\right]_{(i,j)},$$

we can see (4.9) and (4.10) as single resource subproblems $\mathcal{SCE}_{TWO}$:

$$\begin{aligned} \text{Min} \quad & \sum_{n\in\mathcal{T}} p_n \left(c^{vp}_{ni} v^p_{ni} + c^{vs}_{ni} v^s_{ni}\right) & & \\ \text{s.t.} \quad & \sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \geq \delta^1_n & \forall n \in \mathcal{T}, & \\ & v^p_{ni}, v^s_{ni} \in \mathbb{Z}^+ & \forall n \in \mathcal{T}. & \end{aligned} \tag{4.11}$$

and

$$\begin{aligned} \text{Min} \quad & \sum_{n\in\mathcal{T}} p_n \left(c^{up}_{nij} u^p_{nij} + c^{us}_{nij} u^s_{nij}\right) & & \\ \text{s.t.} \quad & \sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \geq \delta^2_n & \forall n \in \mathcal{T}, & \\ & u^p_{nij}, u^s_{nij} \in \mathbb{Z}^+ & \forall n \in \mathcal{T}. & \end{aligned} \tag{4.12}$$

Both (4.11) and (4.12) can be solved using the polynomial-time primal and dual algorithms developed in Section 3.2.1 and Section 3.2.2.

Having identified the sub-structure of the problem $\mathcal{SNCE}_{TWO}$, we propose the approximation algorithm outlined in Algorithm 5. In Step 1, the LP relaxation of $\mathcal{SNCE}_{TWO}$ is solved to optimality. This problem is a multi-stage stochastic linear program. Step 2 requires the solution of $|\mathcal{N}| + |\mathcal{A}|$ single resource stochastic capacity expansion subproblems, which are multi-stage stochastic integer programs. It is noteworthy that if we replace the right hand side in constraints of (4.9) by $\left\lceil \frac{\sum_{(i,j)\in\delta^+_i} x_{nij}}{C^v_i} \right\rceil$, and the right hand side in constraints of (4.10) by $\left\lceil \frac{x_{nij}}{C^u_{ij}} \right\rceil$, then the programs can be solved using the primal and dual algorithms developed in Section 3.2.1 and Section 3.2.2. Finally, Step 3 requires the solution of N (for all nodes in the tree) independent linear capacity allocation problems, which are min-cost flow problems.

Algorithm 5 The Approximation Algorithm for $\mathcal{SNCE}_{TWO}$

1: Solve the LP relaxation of $\mathcal{SNCE}_{TWO}$ and Let $(\mathcal{X}^{LP}, \mathcal{Y}^{LP}) := (\mathbf{x}_n^{LP}, \mathbf{y}_n^{LP})_{n\in\mathcal{T}}$. If $\mathbf{y}_n^{LP}$ is integral for all n, stop and return $(\mathcal{X}^{LP}, \mathcal{Y}^{LP})$.

2: Fix the capacity allocation decisions $\mathcal{X}^{LP}$. For each resource, solve the independent single resource capacity expansion problem, i.e., for node $i \in \mathcal{N}$, solve (4.11) with $\delta_n^1 = \left\lceil \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}^{LP}}{C_i^v} \right\rceil$, and for arc $(i,j) \in \mathcal{A}$, solve (4.12) with $\delta_n^2 = \left\lceil \frac{x_{nij}^{LP}}{C_{ij}^u} \right\rceil$. Let $\mathcal{Y}^H = (\mathbf{y}_n^H)_{n\in\mathcal{T}} = (\mathbf{u}_n^{\mathbf{p}H}, \mathbf{v}_n^{\mathbf{p}H}, \mathbf{u}_n^{\mathbf{s}H}, \mathbf{v}_n^{\mathbf{s}H})_{n\in\mathcal{T}}$ denote the corresponding optimal solution.

3: Fix the capacity allocation decisions $\mathcal{Y}^H$. For each $n \in \mathcal{T}$, solve the independent network flow problem (4.13) and let $\mathbf{x}_n^H$ denote the corresponding optimal solution, and $\mathcal{X}^H = (\mathbf{x}_n^H)_{n\in\mathcal{T}}$.

$$\begin{aligned}
\text{Min} \quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}^u x_{nij} && \\
\text{s.t.} \quad & \sum_{(j,i)\in\delta_i^-} x_{nji} - \sum_{(i,j)\in\delta_i^+} x_{nij} = d_{ni} && \forall i \in \mathcal{N}, \\
& x_{nij} \le C_{ij}^u \left(\sum_{m\in\bar{\mathcal{P}}(n)} u_{mij}^{pH} + u_{nij}^{sH} \right) && \forall (i,j) \in \mathcal{A}, \qquad (4.13)\\
& \sum_{(i,j)\in\delta_i^+} x_{nij} \le C_i^v \left(\sum_{m\in\bar{\mathcal{P}}(n)} v_{mi}^{pH} + v_{ni}^{sH} \right) && \forall i \in \mathcal{N}, \\
& x_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}.
\end{aligned}$$

4: Return $(\mathcal{X}^H, \mathcal{Y}^H) := (\mathbf{x}_n^H, \mathbf{y}_n^H)_{n\in\mathcal{T}}$.

We can prove that the approximation algorithm has an asymptotic convergence property. First, we present an upper bound for the gap between the optimal solution and the solution returned by the approximation algorithm:

Theorem 4.2. *For a solution $(\mathcal{X}, \mathcal{Y})$, let $f(\mathcal{X}, \mathcal{Y})$ be the objective function. If $(\mathcal{X}^*, \mathcal{Y}^*)$ is the optimal solution of $\mathcal{SNCE}_{TWO}$ and $(\mathcal{X}^H, \mathcal{Y}^H)$ is the solution returned* *by the approximation algorithm, then,*

$$f(\mathcal{X}^H, \mathcal{Y}^H) - f(\mathcal{X}^*, \mathcal{Y}^*) \leq \sum_{(i,j)\in\mathcal{A}} \left(c_{1ij}^{up} + c_{1ij}^{us}\right) + \sum_{i\in\mathcal{N}} \left(c_{1i}^{vp} + c_{1i}^{vs}\right).$$

Proof. We know that $f(\mathcal{X}^{LP}, \mathcal{Y}^{LP}) \leq f(\mathcal{X}^*, \mathcal{Y}^*)$. Therefore,

$$\begin{aligned} f(\mathcal{X}^H, \mathcal{Y}^H) - f(\mathcal{X}^*, \mathcal{Y}^*) &\leq f(\mathcal{X}^H, \mathcal{Y}^H) - f(\mathcal{X}^{LP}, \mathcal{Y}^{LP}) \\ &= f(\mathcal{X}^H, \mathcal{Y}^H) - f(\mathcal{X}^{LP}, \mathcal{Y}^H) \\ &\quad + f(\mathcal{X}^{LP}, \mathcal{Y}^H) - f(\mathcal{X}^{LP}, \mathcal{Y}^{LP}) \\ &\leq f(\mathcal{X}^{LP}, \mathcal{Y}^H) - f(\mathcal{X}^{LP}, \mathcal{Y}^{LP}), \end{aligned}$$

where the last inequality holds since $\mathcal{X}^H$ is an optimal capacity allocation corresponding to $\mathcal{Y}^H$, while $\mathcal{X}^{LP}$ is just a feasible solution. Therefore, $f(\mathcal{X}^H, \mathcal{Y}^H) \leq f(\mathcal{X}^{LP}, \mathcal{Y}^H)$. Now we have:

$$\begin{aligned} f(\mathcal{X}^{LP}, \mathcal{Y}^H) - f(\mathcal{X}^{LP}, \mathcal{Y}^{LP}) &= \\ &\sum_{(i,j)\in\mathcal{A}} \sum_{n\in\mathcal{T}} p_n \left[c_{nij}^{up}\left(u_{nij}^{pH} - u_{nij}^{pLP}\right) + c_{nij}^{us}\left(u_{nij}^{sH} - u_{nij}^{sLP}\right)\right] \\ &+ \sum_{i\in\mathcal{N}} \sum_{n\in\mathcal{T}} p_n \left[c_{ni}^{vp}\left(v_{ni}^{pH} - v_{ni}^{pLP}\right) + c_{ni}^{vs}\left(v_{ni}^{sH} - v_{ni}^{sLP}\right)\right]. \end{aligned} \tag{4.14}$$

Note that we can analyze the subproblems for nodes and arcs separately. For any node $i \in \mathcal{N}$,

$$\begin{aligned}
&\sum_{n\in\mathcal{T}} p_n\left(c^{vp}_{ni}v^{pH}_{ni} + c^{vs}_{ni}v^{sH}_{ni}\right) \\
&\quad = \quad \text{Min} \quad \sum_{n\in\mathcal{T}} p_n\left(c^{vp}_{ni}v^{p}_{ni} + c^{vs}_{ni}v^{s}_{ni}\right) \\
&\qquad\quad \text{s.t.} \quad \sum_{m\in\bar{\mathcal{P}}(n)} v^{p}_{mi} + v^{s}_{ni} \geq \left\lceil \frac{\sum_{(i,j)\in\delta_i^+} x^{LP}_{nij}}{C^v_i} \right\rceil \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad\quad v^{p}_{ni}, v^{s}_{ni} \in \mathbb{R}^+ \qquad \forall n\in\mathcal{T} \\
&\quad = \quad \text{Max} \quad \sum_{n\in\mathcal{T}} \left\lceil \frac{\sum_{(i,j)\in\delta_i^+} x^{LP}_{nij}}{C^v_i} \right\rceil \pi_{in} \\
&\qquad\quad \text{s.t.} \quad \textstyle\sum_{m\in\bar{\mathcal{T}}(n)} \pi_{im} \leq p_n c^{vp}_{ni} \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad\quad \pi_{in} \leq p_n c^{vs}_{ni} \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad\quad \pi_{in} \in \mathbb{R}^+ \qquad \forall n\in\mathcal{T},
\end{aligned} \tag{4.15}$$

and

$$\begin{aligned}
&\sum_{n\in\mathcal{T}} p_n\left(c^{vp}_{ni}v^{pLP}_{ni} + c^{vs}_{ni}v^{sLP}_{ni}\right) \\
&\quad = \quad \text{Min} \quad \sum_{n\in\mathcal{T}} p_n\left(c^{vp}_{ni}v^{p}_{ni} + c^{vs}_{ni}v^{s}_{ni}\right) \\
&\qquad\quad \text{s.t.} \quad \sum_{m\in\bar{\mathcal{P}}(n)} v^{p}_{mi} + v^{s}_{ni} \geq \frac{\sum_{(i,j)\in\delta_i^+} x^{LP}_{nij}}{C^v_i} \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad\quad v^{p}_{ni}, v^{s}_{ni} \in \mathbb{R}^+ \qquad \forall n\in\mathcal{T} \\
&\quad = \quad \text{Max} \quad \sum_{n\in\mathcal{T}} \frac{\sum_{(i,j)\in\delta_i^+} x^{LP}_{nij}}{C^v_i} \pi_{in} \\
&\qquad\quad \text{s.t.} \quad \textstyle\sum_{m\in\bar{\mathcal{T}}(n)} \pi_{im} \leq p_n c^{vp}_{ni} \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad\quad \pi_{in} \leq p_n c^{vs}_{ni} \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad\quad \pi_{in} \in \mathbb{R}^+ \qquad \forall n\in\mathcal{T}.
\end{aligned} \tag{4.16}$$

Therefore,

$$\begin{aligned}
&\sum_{n\in\mathcal{T}} p_n \left[c_{ni}^{vp}(v_{ni}^{pH} - v_{ni}^{pLP}) + c_{ni}^{vs}(v_{ni}^{sH} - v_{ni}^{sLP})\right] \\
&\leq \quad \text{Max} \quad \sum_{n\in\mathcal{T}} \left(\left\lceil \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}^{LP}}{C_i^v} \right\rceil - \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}^{LP}}{C_i^v} \right) \pi_{in} \\
&\qquad \text{s.t.} \quad \sum_{m\in\bar{\mathcal{T}}(n)} \pi_{im} \leq p_n c_{ni}^{vp} \qquad \forall n\in\mathcal{T}, \\
&\qquad\qquad \pi_{in} \leq p_n c_{ni}^{vs} \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad v_{ni}^p, v_{ni}^s \in \mathbb{R}^+ \qquad \forall n\in\mathcal{T} \\
&\leq \quad \text{Max} \quad \sum_{n\in\mathcal{T}} \pi_{in} \\
&\qquad \text{s.t.} \quad \sum_{m\in\bar{\mathcal{T}}(n)} \pi_{im} \leq p_n c_{ni}^{vp} \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad \pi_{in} \leq p_n c_{ni}^{vs} \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad v_{ni}^p, v_{ni}^s \in \mathbb{R}^+ \qquad \forall n\in\mathcal{T} \\
&= \quad \text{Min} \quad \sum_{n\in\mathcal{T}} p_n \left(c_{ni}^{vp} v_{ni}^p + c_{ni}^{vs} v_{ni}^s\right) \\
&\qquad \text{s.t.} \quad \sum_{m\in\bar{\mathcal{P}}(n)} v_{mi}^p + v_{ni}^s \geq 1 \qquad \forall n\in\mathcal{T} \\
&\qquad\qquad v_{ni}^p, v_{ni}^s \in \mathbb{R}^+ \qquad \forall n\in\mathcal{T} \\
&\leq \quad c_{1i}^{vp} + c_{1i}^{vs},
\end{aligned} \tag{4.17}$$

where in (4.17), the first inequality comes from (4.15) and (4.16), the second inequality holds because $\left\lceil \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}^{LP}}{C_i^v} \right\rceil - \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}^{LP}}{C_i^v} \leq 1$, and the equality comes from duality. Finally, consider the case when the demand is 1 in all nodes of the scenario tree. Note that a feasible solution is to buy 1 unit of permanent capacity at the root node (which will satisfy the demands for all nodes except for the root node) and 1 unit of spot market capacity to satisfy the demand of the root node. This feasible solution will provide an upper bound $c_{1i}^{vp} + c_{1i}^{vs}$ for the optimal objective value. Therefore, the last inequality holds. We can repeat this reasoning for each arc and show that for all arc $(i, j) \in \mathcal{A}$:

$$\sum_{n \in \mathcal{T}} p_n \left[c_{nij}^{up}(u_{nij}^{pH} - u_{nij}^{pLP}) + c_{nij}^{us}(u_{nij}^{sH} - s_{nij}^{sLP}) \right] \leq c_{1ij}^{up} + c_{1ij}^{us} \tag{4.18}$$

The result will follow if we incorporate (4.18) and (4.17) into (4.14). $\square$

Theorem 4.2 shows that the optimality gap of the solution returned by Algorithm 5 is bounded by the sum of the permanent and spot market capacity acquisition costs of arcs and nodes of the network for the root node in the scenario tree, and is independent of other model parameters such as the number of time periods, the number of branches in the scenario tree, etc. This property enables us to get the following asymptotic convergence result:

Theorem 4.3. *Assume that:*

(i) There exists $\epsilon_1 > 0$ such that for all $n \in \mathcal{T}$, there is at least one node in the network whose demand is at least ϵ_1;

(ii) There exists $\epsilon_2 > 0$ such that for all $n \in \mathcal{T}$, and for any arc $(i', j') \in A$ with $\sum_{(j',i') \in \delta_{i'}^-} x_{nj'i'} - \sum_{(i',j') \in \delta_{i'}^+} x_{ni'j'} < 0$, the total capacity allocation cost $c_{i'j'}^u + c_{i'}^v \geq \epsilon_2$;

then

$$\lim_{T \to \infty} \frac{f(\mathcal{X}^H, \mathcal{Y}^H) - f(\mathcal{X}^*, \mathcal{Y}^*)}{f(\mathcal{X}^*, \mathcal{Y}^*)} = 0$$

for $\mathcal{SNCE}_{TWO}$.

Proof. According to Theorem 4.2, we have:

$$\frac{f(\mathcal{X}^H, \mathcal{Y}^H) - f(\mathcal{X}^*, \mathcal{Y}^*)}{f(\mathcal{X}^*, \mathcal{Y}^*)} \leq \frac{\sum_{(i,j)\in\mathcal{A}} \left(c_{1ij}^{up} + c_{1ij}^{us}\right) + \sum_{i\in\mathcal{N}} \left(c_{1i}^{vp} + c_{1i}^{vs}\right)}{f(\mathcal{X}^*, \mathcal{Y}^*)}.$$

Therefore, we only need to show that $\lim_{T\to\infty} f(\mathcal{X}^*, \mathcal{Y}^*) = \infty$. For a fixed T, consider the dual of the linear relaxation of $\mathcal{SNCE}_{TWO}$ with dual variables π_{ni}^d, π_{nij}^u, and π_{ni}^v for (4.2), (4.3), and (4.4), respectively:

$$\begin{aligned}
\text{Max} \quad & \sum_{n\in\mathcal{T}} \pi_{ni}^d d_{ni} & \\
\text{s.t.} \quad & -\pi_{ni}^d + \pi_{nj}^d - \pi_{nij}^u - \pi_{ni}^v \leq p_n c_{ij}^u & \forall (i,j)\in\mathcal{A}, \forall n\in\mathcal{T}, \\
& C_{ij}^u \sum_{m\in\bar{\mathcal{T}}(n)} \pi_{nij}^u \leq p_n c_{nij}^{up} & \forall (i,j)\in\mathcal{A}, \forall n\in\mathcal{T}, \\
& C_i^v \sum_{m\in\bar{\mathcal{T}}(n)} \pi_{ni}^v \leq p_n c_{ni}^{vp} & \forall i\in\mathcal{N}, \forall n\in\mathcal{T}, \\
& C_{ij}^u \pi_{nij}^u \leq p_n c_{nij}^{us} & \forall (i,j)\in\mathcal{A}, \forall n\in\mathcal{T}, \\
& C_i^v \pi_{ni}^v \leq p_n c_{ni}^{vs} & \forall i\in\mathcal{N}, \forall n\in\mathcal{T}, \\
& \pi_{ni}^d,\ \pi_{ni}^v \in \mathbb{R}^+ & \forall i\in\mathcal{N}, \forall n\in\mathcal{T}, \\
& \pi_{nij}^u \in \mathbb{R}^+ & \forall (i,j)\in\mathcal{A}, \forall n\in\mathcal{T}.
\end{aligned}$$

Let the vector form of the dual variables be π_n^d, π_n^u, and π_n^v. For any dual feasible solution $(\pi_n^d, \pi_n^u, \pi_n^v)$, let $g(\pi_n^d, \pi_n^u, \pi_n^v) = \sum_{n\in\mathcal{T}} \pi_{ni}^d d_{ni}$ denote the objective value of the above dual problem. Clearly, weak duality ensures that $f(\mathcal{X}^{LP}, \mathcal{Y}^{LP}) \geq g(\pi_n^d, \pi_n^u, \pi_n^v)$. Now, we try to find a dual feasible solution $(\bar{\pi}_n^d, \bar{\pi}_n^u, \bar{\pi}_n^v)$ such that $g(\bar{\pi}_n^d, \bar{\pi}_n^u, \bar{\pi}_n^v) \geq T\epsilon$ for some $\epsilon > 0$. We set $\bar{\pi}_n^u = \bar{\pi}_n^v = 0$, for all $n \in \mathcal{T}$. Assumption (i) requires that there is at least one node $i' \in \mathcal{N}$ such that $-d_{ni'} \geq \epsilon_1$. So, constraint (4.2) ensures that there is at least one node i' such that $\sum_{(j',i')\in\delta^-_{i'}} x_{nj'i'} - \sum_{(i',j')\in\delta^+_{i'}} x_{ni'j'} < 0$; and we can find at least one emanating arc (i',j') of node i'. Now, if we let $\bar{\pi}^d_{ni'} = -p_n(c^u_{i'j'} + c^v_{i'})$ and $\bar{\pi}^d_{ni} = 0$ for all $i \in \mathcal{N} \setminus \{j'\}$, then $(\bar{\pi}^d_n, \bar{\pi}^u_n, \bar{\pi}^v_n)$ is a dual feasible solution. Also, assumption (ii) requires that $-\bar{\pi}^d_{ni'} \geq p_n\epsilon_2$. Therefore, $g(\bar{\pi}^d_n, \bar{\pi}^u_n, \bar{\pi}^v_n) = \sum_{n\in\mathcal{T}} \bar{\pi}^d_{ni} d_{ni} = \sum_{n\in\mathcal{T}} \bar{\pi}^d_{ni'} d_{ni'} \geq \sum_{n\in\mathcal{T}} p_n\epsilon_1\epsilon_2$. Since the summation of probabilities of each stage in the scenario tree is 1 and we have T stages, $\sum_{n\in\mathcal{T}} p_n = T$, which means that $g(\bar{\pi}^d_n, \bar{\pi}^u_n, \bar{\pi}^v_n) \geq T\epsilon_1\epsilon_2$. Using weak duality, we can conclude that $f(\mathcal{X}^{LP}, \mathcal{Y}^{LP}) \geq T\epsilon_1\epsilon_2$. Therefore, $\lim_{T\to\infty} f(\mathcal{X}^{LP}, \mathcal{Y}^{LP}) = \infty$. □

We showed that the approximation scheme will converge to the optimal solution, if we consider a sufficiently large number of periods.

4.4 The Stochastic Network Capacity Expansion Model with Three Sources of Capacity

We can extend all the results in Section 4.2 and 4.3 to the situation where three sources of capacity, i.e., permanent, spot market, and contract capacities interact with each other. We assume that the contract capacity will be available in the next period right after signing the contract. Here, we consider the case that the contract length is one period, but our model can capture the case where the contract can be signed for an arbitrary number of periods. In addition to notations introduced in Table 4.1, we need the ones presented in Table 4.2.

The stochastic network capacity expansion model with three sources of capacity can be formulated as follows:

Table 4.2: Additional notations of the model $\mathcal{SNCE}_{THREE}$

Parameters:	
c^{uc}_{nij}	Contract capacity acquisition cost for arc $(i,j) \in \mathcal{A}$ at node $n \in \mathcal{T}$
c^{vc}_{ni}	Contract capacity acquisition cost for node $i \in \mathcal{N}$ at node $n \in \mathcal{T}$
Decision variables:	
u^{c}_{nij}	Contract capacity acquisition for arc $(i,j) \in \mathcal{A}$ at node $n \in \mathcal{T}$
v^{c}_{ni}	Contract capacity acquisition of node $i \in \mathcal{N}$ at node $n \in \mathcal{T}$

Model $\mathcal{SNCE}_{THREE}$:

$$\begin{aligned}
\text{Min } & \sum_{n\in\mathcal{T}} p_n \Bigg[\sum_{(i,j)\in\mathcal{A}} (c^u_{ij}x_{nij} + c^{up}_{nij}u^p_{nij} + c^{us}_{nij}u^s_{nij} + c^{uc}_{nij}u^c_{nij}) \\
& \qquad\qquad + \sum_{i\in\mathcal{N}} (c^{vp}_{ni}v^p_{ni} + c^{vs}_{ni}v^s_{ni} + c^{vc}_{ni}v^c_{ni}) \Bigg] \\
\text{s.t. } & \sum_{(j,i)\in\delta_i^-} x_{nji} - \sum_{(i,j)\in\delta_i^+} x_{nij} = d_{ni} && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& x_{nij} \le C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} + u^c_{a(n)ij} \right) && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \\
& \sum_{(i,j)\in\delta_i^+} x_{nij} \le C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} + v^c_{a(n)i} \right) && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& x_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \\
& u^p_{nij},\ u^s_{nij},\ u^c_{nij} \in \mathbb{Z}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \\
& v^p_{ni},\ v^s_{ni},\ v^c_{ni} \in \mathbb{Z}^+ && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}.
\end{aligned}$$

Similar to $\mathcal{SNCE}_{TWO}$, it can be proved that $\mathcal{SNCE}_{THREE}$ is an NP-hard problem.

The approximation algorithm presented in Section 4.3 can be used to solve this problem, since $\mathcal{SNCE}_{THREE}$ has the same decomposition structure as $\mathcal{SNCE}_{TWO}$, i.e., it can be decomposed into single resource capacity expansion problems corresponding to nodes and arcs. Each subproblem of $\mathcal{SNCE}_{THREE}$, which includes all three kinds of capacity, can be formulated as:

Model $\mathcal{SCE}_{THREE}$:

$$\begin{aligned} \text{Min} \quad & \sum_{n \in \mathcal{T}} p_n \left(c_n^p x_n + c_n^s z_n + c_n^c y_n \right) & & \\ \text{s.t.} \quad & \sum_{m \in \bar{\mathcal{P}}(n)} x_m + z_n + y_{a(n)} \geq d_n & \forall n \in \mathcal{T} & \\ & x_n, z_n, y_n \in \mathbb{Z}^+ & \forall n \in \mathcal{T}. & \end{aligned} \tag{4.19}$$

Therefore, the approximation algorithm can also be applied to $\mathcal{SNCE}_{THREE}$, provided that in step 2, $\mathcal{SCE}_{THREE}$ can be solved efficiently for all nodes and arcs. This can be done by applying polynomial-time algorithms that we designed in Section 3.3.1 and Section 3.3.2.

For $\mathcal{SNCE}_{THREE}$, we define $\mathcal{Y} = (\mathbf{y}_n)_{n \in \mathcal{T}} = (\mathbf{u}_n^{\mathbf{p}}, \mathbf{v}_n^{\mathbf{p}}, \mathbf{u}_n^{\mathbf{s}}, \mathbf{v}_n^{\mathbf{s}}, \mathbf{u}_n^{\mathbf{c}}, \mathbf{v}_n^{\mathbf{c}})_{n \in \mathcal{T}}$ where $\mathbf{u}_n^{\mathbf{p}} = (u_{nij}^p)_{(i,j) \in \mathcal{A}}$, $\mathbf{v}_n^{\mathbf{p}} = (v_{ni}^p)_{i \in \mathcal{N}}$, $\mathbf{u}_n^{\mathbf{s}} = (u_{nij}^s)_{(i,j) \in \mathcal{A}}$, $\mathbf{v}_n^{\mathbf{s}} = (v_{ni}^s)_{i \in \mathcal{N}}$, $\mathbf{u}_n^{\mathbf{c}} = (u_{nij}^c)_{(i,j) \in \mathcal{A}}$, and $\mathbf{v}_n^{\mathbf{c}} = (v_{ni}^c)_{i \in \mathcal{N}}$. Then the following results hold:

Theorem 4.4. *For $\mathcal{SNCE}_{THREE}$, given the same assumptions as in Theorem 4.3, we have*

$$\lim_{T \to \infty} \frac{f(\mathcal{X}^H, \mathcal{Y}^H) - f(\mathcal{X}^*, \mathcal{Y}^*)}{f(\mathcal{X}^*, \mathcal{Y}^*)} = 0$$

The proof of Theorem 4.4 is analogous to that of Theorem 4.3. So we omit it for brevity. Theorem 4.4 shows that the approximation algorithm keeps its asymptotic convergence property when dealing with three sources of capacity.

4.5 Experimental Results

In this section, we report on the experimental results of the approximation algorithm for solving $\mathcal{SNCE}_{TWO}$. We mainly focus on two objectives: (i) to show the performance of the approximation algorithm compared to CPLEX MIP solver, and (ii) to show the asymptotically convergence property of the approximation algorithm proved in Theorem 4.3.

We present the experimental results in two parts: in the first part (Section 4.5.1), we consider small and medium-sized problems that can be solved by CPLEX MIP solver within two hours. Therefore, we can compare the solution time of the approximation algorithm and that of CPLEX. In the second part (Section 4.5.2), we report on the performance of the approximation algorithm on large-scale instances that cannot be solved by CPLEX MIP solver within two hours.

The test problems have been generated as follows: We generate random numbers for costs of the problem. The demand for each node of the scenario tree has been independently generated by multiplying the demand of the root node by a random number generated from a lognormal distribution with mean μ and standard deviation σ. For each time period, we increase the mean of the demand by $0.5t_n$ and keep the same standard deviation. This guarantees that we have increasing demand throughout the scenario tree. Note that when we increase the size of the scenario tree, we keep the same structure for the network. This enables us to track the performance of the approximation algorithm for different scenario tree sizes.

All the codes are written in GAMS 24.2.2 calling CPLEX 12.6. The experiments are conducted on a computer with an AMD Opteron 2.79GHz processor and 64 GB of RAM running Microsoft Windows Server 2008 R2.

Table 4.3: Data for instances of $\mathcal{SNCE}_{TWO}$

$\vert\mathcal{N}\vert$	$\vert T.N\vert$	$\vert\mathcal{A}\vert$	T	No. of Variables		No. of	N
				Real	Integer	Constraints	
10	6	36	3	252	644	392	7
			4	540	1,380	840	15
			5	1,116	2,852	1,736	31
			6	2,268	5,796	3,528	63
			7	4,572	11,684	7,112	127
15	9	84	3	588	1,386	798	7
			4	1,260	2,970	1,710	15
			5	2,604	6,138	3,534	31
			6	5,292	12,474	7,182	63
			7	10,668	25,146	14,478	127
20	12	152	3	1,064	2,408	1,344	7
			4	2,280	5,160	2,880	15
			5	4,712	10,664	5,952	31
			6	9,576	21,672	12,096	63
			7	19,304	43,688	24,384	127

4.5.1 Comparing CPLEX and the Approximation Algorithm

In the first experiment, we increase the number of nodes in the network ($|\mathcal{N}|$) from 10 to 20, and the number of periods in the scenario tree (T) from 3 to 7. The number of branches for each non-leaf node in the scenario tree is 2. The combination of number of branches and different scenario tree periods results in various scenario tree sizes which are presented as N. These parameters have been set so that CPLEX MIP solver can solve the model $\mathcal{SNCE}_{TWO}$ within two hours.

Table 4.3 shows the data for 15 instances created for this part. To get a sense of the size of these instances, we present the number of constraints and variables, the number of periods in the scenario tree (T), and the size of the scenario tree (N). We

Table 4.4: Comparison of CPLEX and the approximation algorithm for $\mathcal{SNCE}_{TWO}$

$\lvert\mathcal{N}\rvert$	T	Time (s)		Time	Optimality
		CPLEX	Approx. Alg.	(%)	Gap (%)
	3	0.42	1.91	454.76	0.3028
	4	0.95	1.96	206.31	0.1297
10	5	3.64	2.01	55.21	0.0340
	6	11.42	3.82	33.45	0.0121
	7	44.27	5.26	11.88	0.0044
	3	0.73	3.75	513.69	0.3447
	4	6.51	4.26	65.43	0.0206
15	5	102.05	5.91	5.79	0.0183
	6	187.35	5.37	2.86	0.0179
	7	239.01	7.18	3.00	0.0032
	3	4.71	5.24	111.25	0.1641
	4	11.02	6.49	58.89	0.0499
20	5	125.08	6.57	5.25	0.0046
	6	363.85	7.52	2.06	0.0005
	7	422.17	9.92	2.34	0.0002

also identify the structure of the network by presenting the number of nodes ($|\mathcal{N}|$), the number of arcs ($|\mathcal{A}|$), and the number of transient nodes ($|T.N|$).

In Table 4.4, we compare the solution time by CPLEX and the solution time by the approximation algorithm. Also, we provide the optimality gap between the optimal solution and the solution given by the approximation algorithm. From Time (%) column, we can observe that the approximation algorithm can solve most of instances much faster that CPLEX. The Optimality Gap(%) column shows that as the number of period increases, the gap decreases which is in accordance with the convergence result that we obtained in Theorem 4.3.

Table 4.5: Data for large-scale instances of $\mathcal{SNCE}_{TWO}$

$\lvert\mathcal{N}\rvert$	$\lvert T.N\rvert$	$\lvert\mathcal{A}\rvert$	T	No. of Variables		No. of	N
				Real	Integer	Constraints	
50	30	980	3	6,860	14,420	7,560	7
			4	14,700	30,900	16,200	15
			5	30,380	63,860	33,480	31
			6	61,740	129,780	68,040	63
			7	124,460	261,620	137,160	127
100	60	3,960	3	27,720	56,840	29,120	7
			4	59,400	121,800	62,400	15
			5	122,760	251,720	128,960	31
			6	249,480	511,560	262,080	63
			7	502,920	1,031,240	528,320	127
250	150	24,900	3	174,300	352,100	177,800	7
			4	373,500	754,500	381,000	15
			5	771,900	1,559,300	787,400	31
			6	1,568,700	3,168,900	1,600,200	63
			7	3,162,300	6,388,100	3,225,800	127
500	300	99,800	3	698,600	1,404,200	705,600	7
			4	1,497,000	3,009,000	1,512,000	15
			5	3,093,800	6,218,600	3,124,800	31
			6	6,287,400	12,637,800	6,350,400	63
			7	12,674,600	25,476,200	12,801,600	127

4.5.2 Approximation Algorithm's Performance for Large-scale Instances

For large-scale instances of $\mathcal{SNCE}_{TWO}$, CPLEX cannot find the solution in a reasonable time (we stop the solver when it exceeds two hours time limit). We create 20 large-scale instances and present their data in Table 4.5.

Table 4.6 shows the solution time of the approximation algorithm for large-scale instances. For these instances, since we do not have the optimal solution, we use the optimal objective value of the linear relaxation as the lower bound and report the relative gap between the objective value returned by the approximation algorithm and the lower bound. The reported gap confirms the convergence result of Theorem 4.3.

Table 4.6: Approximation algorithm results for large-scale instances of $\mathcal{SNCE}_{TWO}$

$\|\mathcal{N}\|$	T	Approx. Alg. Time (s)	Gap (%)
50	3	28.02	1.641
	4	29.32	0.807
	5	30.79	0.346
	6	33.20	0.124
	7	40.20	0.037
100	3	116.75	1.710
	4	117.78	0.949
	5	118.18	0.380
	6	125.66	0.134
	7	141.55	0.041
250	3	705.81	1.587
	4	724.78	0.871
	5	730.28	0.377
	6	871.54	0.138
	7	2841.56	0.040
500	3	2803.02	1.475
	4	2919.53	0.823
	5	3488.39	0.354
	6	4361.44	0.126
	7	21264.61*	0.036

*The time to solve the linear relaxation is 18,381.57 seconds.

Chapter 5

Decomposition Algorithms for the Stochastic Network Capacity Expansion Problem

In Chapter 4, we designed an approximation algorithm to solve $\mathcal{SNCE}_{TWO}$ and $\mathcal{SNCE}_{THREE}$. We showed that the approximation algorithm solution converges to the optimal solution, as the number of scenario tree periods increases. In this chapter, we consider some decomposition algorithms for the stochastic capacity expansion problem $\mathcal{SNCE}_{TWO}$.

5.1 Classical Benders' Decomposition

In this section, we consider Benders' decomposition, introduced by Benders (1962), as one of the most used methods to solve stochastic programming problems. The Benders' decomposition method used to solve two-stage stochastic programming problems is called *L-shaped method* and was introduced by Van Slyke and Wets (1969). Multi-stage stochastic programming problems can be seen as a nested series of two-stage problems to which Benders' decomposition can be applied. This method is called *nested L-shaped method* as described in Birge and Louveaux (2011).

Based on Theorem 4.1, solving the stochastic network capacity expansion problem $\mathcal{SNCE}_{TWO}$ for large-scale instances could be time consuming. Therefore in this section, we propose a Benders' decomposition algorithm to solve the model $\mathcal{SNCE}_{TWO}$ to optimality. Then, we discuss some improvements to the algorithm and their limitations in Section 5.2. Finally in Section 5.4, we propose a Benders' decomposition-based algorithm that can find tight bounds for the model $\mathcal{SNCE}_{TWO}$.

The $\mathcal{SNCE}_{TWO}$ is an MIP model including continuous flow variables and integer capacity acquisition variables. If the capacity acquisition variables are fixed, the problem becomes a network flow problem with real variables. This observation motivates us to design the Benders' decomposition algorithm by fixing all integer variables for the subproblems, and keeping all integer variables in the master problem. With this setting, in each iteration of the Benders' decomposition algorithm, we have to solve N subproblems, corresponding to each node in the scenario tree. After solving each subproblem, if it is unbounded, a feasibility cut will be added to the master problem. If all subproblems are optimal, then N optimality cuts will be added to the master problem.

To design the classical Benders' decomposition algorithm, we keep the capacity acquisition (integer) variables in the master problem and move the flow (real) variables to the subproblems. Thus, the initial master problem can be modeled as follows:

$$\begin{aligned}
\text{Min} \quad & \sum_{n\in\mathcal{T}} p_n \left[\sum_{(i,j)\in\mathcal{A}} (c^{up}_{nij} u^p_{nij} + c^{us}_{nij} u^s_{nij}) + \sum_{i\in\mathcal{N}} (c^{vp}_{ni} v^p_{ni} + c^{vs}_{ni} v^s_{ni}) + \eta_n \right] \\
\text{s.t.} \quad & u^p_{nij}, u^s_{nij} \in \mathbb{Z}^+ && (i,j)\in\mathcal{A}, \forall n\in\mathcal{T}, \qquad (5.1)\\
& v^p_{ni}, v^s_{ni} \in \mathbb{Z}^+ && i\in\mathcal{N}, \forall n\in\mathcal{T},\\
& \eta_n \in \mathbb{R}^+ && \forall n\in\mathcal{T},
\end{aligned}$$

where η_n is an auxiliary variable that is used to underestimate the flow cost for node n in the scenario tree. Since the flow costs are non-negative, then for all $n \in \mathcal{T}$, the variable η_n is also non-negative. After each iteration of Benders' decomposition procedure, optimality or feasibility cuts will be added to the above master problem. Now, we fix the capacity acquisition variables for the subproblems. As before, let $\mathcal{Y} = (\mathbf{y}_n)_{n\in\mathcal{T}} = (\mathbf{u}^{\mathbf{p}}_n, \mathbf{v}^{\mathbf{p}}_n, \mathbf{u}^{\mathbf{s}}_n, \mathbf{v}^{\mathbf{s}}_n)_{n\in\mathcal{T}}$ where $\mathbf{u}^{\mathbf{p}}_n = (u^p_{nij})_{(i,j)\in\mathcal{A}}$, $\mathbf{v}^{\mathbf{p}}_n = (v^p_{ni})_{i\in\mathcal{N}}$, $\mathbf{u}^{\mathbf{s}}_n = (u^s_{nij})_{(i,j)\in\mathcal{A}}$, and $\mathbf{v}^{\mathbf{s}}_n = (v^s_{ni})_{i\in\mathcal{N}}$. Then for a fixed $\mathcal{Y}$ and for each $n \in \mathcal{T}$, the subproblem of $\mathcal{SNCE}_{TWO}$ can be formulated as follows:

$$\begin{aligned}
\mathcal{SP}(\mathcal{Y})_n = \text{Min} \quad & \sum_{(i,j)\in\mathcal{A}} c^u_{ij} x_{nij} \\
\text{s.t.} \quad & \sum_{(j,i)\in\delta^-_i} x_{nji} - \sum_{(i,j)\in\delta^+_i} x_{nij} = d_{ni} && \forall i\in\mathcal{N},\\
& x_{nij} \le C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \right) && \forall (i,j)\in\mathcal{A},\\
& \sum_{(i,j)\in\delta^+_i} x_{nij} \le C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \right) && \forall i\in\mathcal{N},\\
& x_{nij} \in \mathbb{R}^+ && \forall (i,j)\in\mathcal{A}.
\end{aligned}$$

Now suppose π^d_{ni}, π^u_{nij}, and π^v_{ni} are dual variables corresponding to demand, arc capacity, and node capacity constraints of the above subproblem, respectively. Then the dual of the above subproblem for a fixed $\mathcal{Y}$ and each $n \in \mathcal{T}$ can be written as follows:

$$\begin{aligned}
\mathcal{DSP}(\mathcal{Y})_n = \text{Max} \quad & \sum_{i\in\mathcal{N}} \left[\pi^d_{ni} d_{ni} - \pi^v_{ni} C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \right) \right] \\
& - \sum_{(i,j)\in\mathcal{A}} \pi^u_{nij} C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \right) \\
\text{s.t.} \quad & -\pi^d_{ni} + \pi^d_{nj} - \pi^u_{nij} - \pi^v_{ni} \le c^u_{ij} && \forall (i,j) \in \mathcal{A}, \\
& \pi^u_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \\
& \pi^d_{ni},\ \pi^v_{ni} \in \mathbb{R}^+ && \forall i \in \mathcal{N}.
\end{aligned} \tag{5.2}$$

If the subproblem $\mathcal{SP}(\mathcal{Y})_n$ is feasible, solving the dual subproblem (5.2) will result in optimality cut for the master problem. This cut can be constructed as:

$$\sum_{i\in\mathcal{N}} \left[\pi^{d*}_{ni} d_{ni} - \pi^{v*}_{ni} C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \right) \right] - \sum_{(i,j)\in\mathcal{A}} \pi^{u*}_{nij} C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \right) \le \eta_n \tag{5.3}$$

where π^{d*}_{ni}, π^{u*}_{nij}, and π^{v*}_{ni} are optimal solutions returned by (5.2). On the other hand, if the subproblem $\mathcal{SP}(\mathcal{Y})_n$ is infeasible, its dual will be unbounded and solving it will result in a feasible direction defined by ψ^d_{ni}, ψ^u_{nij}, and ψ^v_{ni}. Then the feasibility cut can be constructed as:

$$\sum_{i\in\mathcal{N}} \left[\psi^d_{ni} d_{ni} - \psi^v_{ni} C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \right) \right] - \sum_{(i,j)\in\mathcal{A}} \psi^u_{nij} C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \right) \le 0 \tag{5.4}$$

Now, let Z^*_{master} be the objective function value of the master problem and Z^*_n be the objective function value for subproblem associated with node n. The classical Benders' decomposition algorithm is presented in Algorithm 6.

Algorithm 6 Classical Benders' Decomposition for $\mathcal{SNCE}_{TWO}$

```
 1: Set $opt = 0$, $UB_{\text{tmp}} = 0$, $LB = -\infty$ and $UB = +\infty$.
 2: Create master problem (5.1) with no cuts.
 3: while $(UB - LB)/LB > \epsilon$ do
 4:     Solve master problem (5.1) and obtain the incumbent solution $\mathcal{Y}$.
 5:     $LB = Z^*_{master}$.
 6:     $UB_{\text{tmp}} = \sum_{n \in \mathcal{T}} p_n \left[ \sum_{(i,j) \in \mathcal{A}} (c^{up}_{nij} u^p_{nij} + c^{us}_{nij} u^s_{nij}) + \sum_{i \in \mathcal{N}} (c^{vp}_{ni} v^p_{ni} + c^{vs}_{ni} v^s_{ni}) \right]$
 7:     for $n \in \mathcal{T}$ do
 8:         Solve dual subproblem (5.2).
 9:         if dual subproblem is unbounded,
10:             Add feasibility cut (5.4) to the master problem.
11:         else
12:             $opt = opt + 1$.
13:     end for
14:     if $opt = N$
15:         Add optimality cuts (5.3) to the master problem.
16:         $UB = min(UB, \sum_{n \in \mathcal{T}} p_n Z^*_n + UB_{\text{tmp}})$
17:     end if
18: end while
```

There are several considerations about this classical Benders' decomposition algorithm. First, in each iteration, an integer program (master problem) must be solved. Solving this integer program becomes more difficult as the algorithm adds optimality and feasibility cuts in each iteration. Second, the upper bound updating procedure does not guarantee that a better upper bound will be found in each iteration. In order to ease these difficulties, we propose several techniques that can improve the performance of the classical Benders' decomposition.

5.2 Improvements to Classical Benders' Decomposition

In this section, we propose some techniques to improve the performance of the classical Benders' decomposition for model $\mathcal{SNCE}_{TWO}$ depicted in Algorithm 6. In general, there are two types of techniques used to improve the performance of Benders' decomposition: (i) reducing the number of integer master programs that must be solved, (ii) speeding-up the solution of integer master programs.

5.2.1 Valid Inequalities

Adding valid inequalities to the master problem is a common practice in the literature of Benders' decomposition method. Valid inequalities will help to speed up the solution of master problem. For our model $\mathcal{SNCE}_{TWO}$, we came up with three sets of valid inequalities for the master problem:

Node Valid Inequalities: There are three types of nodes in the network of our problem: origin, destination, and transient. Since the network is balanced, all the flow generated in the origin nodes must be able to go out to satisfy the demand of destination nodes. This means that the capacity of origin nodes must be at least equal to the generated flow. This will result in the following valid inequalities for each $n \in \mathcal{T}$ and for each origin node i with $d_{ni} < 0$:

$$-d_{ni} \leq C_i^v \left(\sum_{m \in \bar{\mathcal{P}}(n)} v_{mi}^p + v_{ni}^s \right) \tag{5.5}$$

On the other hand, each destination node must be able to absorb enough flow to satisfy its demand. This means that the total capacity of arcs that enter a destination node must be at least equal to the demand of that destination node. Thus, for each $n \in \mathcal{T}$ and for each destination node i with $d_{ni} > 0$:

$$d_{ni} \leq \sum_{j} C^{u}_{ji} \left(\sum_{m \in \bar{\mathcal{P}}(n)} u^{p}_{mji} + u^{s}_{nji} \right) \tag{5.6}$$

Arc Valid Inequalities: Following the above explanation, there must be enough capacity for arcs starting from each origin node to carry the flow from it. This means that the total capacity acquired for arcs emanating from origin nodes must be at least equal to their demand. Thus, for each $n \in \mathcal{T}$ and for each origin node i with $d_{ni} < 0$:

$$-d_{ni} \leq \sum_{j} C^{u}_{ij} \left(\sum_{m \in \bar{\mathcal{P}}(n)} u^{p}_{mij} + u^{s}_{nij} \right) \tag{5.7}$$

These valid inequalities can improve the performance of Benders' decomposition by speeding up the solution of integer master problem.

5.2.2 LP Relaxation of the Master Problem

Another common practice for Benders' decomposition is to solve the linear relaxation of the master problem. Proposed by McDaniel and Devine (1977), this method tries to reduce the number of integer problems that must be solved in two phases. In the first phase, the feasibility and optimality cuts are generated using the solutions of the linear relaxation of the master problem. After a specified number of iterations, or reaching a pre-defined gap between lower and upper bound, the integrality constraints will be introduced to the master problem. In the second phase, the lower bound and upper bounds will be reset and the procedure restarts with integer master problems. All optimality and feasibility cuts generated in phase one are valid for the original MIP problem. Using the previously generated cuts, we hope this method can find the optimal solution in a few iterations of solving MIP master problem in phase two.

5.2.3 Pareto-Optimal Cuts

Another method that focuses on reducing the number of integer programs that must be solved is introduced by Magnanti and Wong (1981). This method tries to select the best optimality cut, if more than one is available. That cut is called Pareto-optimal. Formally, a Pareto-optimal cut is a cut that cannot be dominated by any other cut.

Having more than one optimality cut happens when the dual subproblem has multiple optimal solutions. The method proposed by Magnanti and Wong (1981) uses a core point of $\mathcal{Y}$ to evaluate all optimality cuts associated with the solution of the dual subproblem, and then selects the non-dominant one. Adding this non-dominant (or Pareto-optimal) cut to the master problem can reduce the number of iterations in Benders' decomposition method.

In order to find the Pareto-optimal cut, a linear program that uses a core point of $\mathcal{Y}$ must be solved. By definition, $\mathcal{Y}_0$ is a core point of $\mathcal{Y}$, if it is in the relative interior of the convex hull of $\mathcal{Y}$, i.e., $\mathcal{Y}_0 \in ri(\mathcal{Y}^c)$. Let $\mathcal{Y}_0 = (u^{p0}_{nij}, u^{s0}_{nij}, v^{p0}_{ni}, v^{s0}_{ni})$ be a core point of $\mathcal{Y}$ and let $\bar{\pi}$ be the optimal solution of the dual subproblem (5.2) and $Z(\bar{\pi})$ be the corresponding objective function value. Then, the solution of the following linear program

$$
\begin{aligned}
\text{Max} \quad & \sum_{i\in\mathcal{N}} \left[\pi^d_{ni} d_{ni} - \pi^v_{ni} C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^{p0}_{mi} + v^{s0}_{ni} \right) \right] \\
& - \sum_{(i,j)\in\mathcal{A}} \pi^u_{nij} C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^{p0}_{mij} + u^{s0}_{nij} \right) \\
\text{s.t.} \quad & \sum_{i\in\mathcal{N}} \left[\pi^d_{ni} d_{ni} - \pi^v_{ni} C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^{p}_{mi} + v^{s}_{ni} \right) \right] \\
& - \sum_{(i,j)\in\mathcal{A}} \pi^u_{nij} C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^{p}_{mij} + u^{s}_{nij} \right) = Z(\bar{\pi}) \\
& -\pi^d_{ni} + \pi^d_{nj} - \pi^u_{nij} - \pi^v_{ni} \le c^u_{ij} && \forall (i,j) \in \mathcal{A}, \\
& \pi^u_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \\
& \pi^d_{ni},\ \pi^v_{ni} \in \mathbb{R}^+ && \forall i \in \mathcal{N}.
\end{aligned}
\tag{5.8}
$$

creates the Pareto-optimal cut. In our Benders' decomposition implementation, we solve problem (5.8) and add the Pareto-optimal cut to the master problem in step 15 of Algorithm 6.

5.2.4 Maximum Feasible Subsystem Cut Generation (MFS)

This method has been proposed by Saharidis and Ierapetritou (2010) for problems in which more feasibility cuts than optimality cuts is being generated in Benders' decomposition procedure. The method generates a semi-optimality cut, called MFS cut, for the master problem each time the Benders' decomposition procedure generates a feasibility cut. The structure of this cut is similar to the feasibility cut, but it involves the decision variable η. This method tries to identify the maximum feasible subsystem (Amaldi and Kann (1995)) of the subproblems and to relax the minimum possible number of infeasible constraints to make them feasible.

In order to find the minimum number of infeasible constraints, a 0-1 program must be solved. This problem has a binary variable associated with each constraint of the subproblem, or each real variable of the dual subproblem, i.e., w_i^d, w_{ij}^u, and w_i^v. This binary problem must be solved for each infeasible subproblem associated with node n in the scenario tree. For node n, this 0-1 program can be formulated as follows:

$$\begin{aligned}
\mathcal{BMFS}_n = \text{Min} \quad & \sum_{i\in\mathcal{N}} \left(w_i^d + w_i^v\right) + \sum_{(i,j)\in\mathcal{A}} w_{ij}^u \\
\text{s.t.} \quad & \sum_{(j,i)\in\delta_i^-} x_{nji} - \sum_{(i,j)\in\delta_i^+} x_{nij} - Mw_i^d = d_{ni} && \forall i \in \mathcal{N}, \\
& x_{nij} - Mw_{ij}^u \leq C_{ij}^u \left(\sum_{m\in\bar{\mathcal{P}}(n)} u_{mij}^p + u_{nij}^s\right) && \forall (i,j) \in \mathcal{A}, \\
& \sum_{(i,j)\in\delta_i^+} x_{nij} - Mw_i^v \leq C_i^v \left(\sum_{m\in\bar{\mathcal{P}}(n)} v_{mi}^p + v_{ni}^s\right) && \forall i \in \mathcal{N}, \\
& w_i^d,\ w_i^v \in \{0,1\} && \forall i \in \mathcal{N}, \\
& x_{nij} \in \mathbb{R}^+,\ w_{ij}^u \in \{0,1\} && \forall (i,j) \in \mathcal{A}.
\end{aligned} \tag{5.9}$$

where M is a very big positive number. In the solution of $\mathcal{BMFS}_n$ problem, a variable w takes value 1 only if its constraint is infeasible. This guarantees that after solving this problem, we identify the minimum number of constraints that must be relaxed to make the subproblem feasible.

The next step is to add M to the right hand side of the constraints that have been chosen by solving the $\mathcal{BMFS}_n$. By doing this, we will have a relaxed subproblem $\mathcal{RSP}_n$ which is always feasible, and as a result, its dual always generates optimality cut. This cut is called Maximum Feasibility Subsystem (MFS) cut. Using the optimal solution of $\mathcal{RSP}_n$, i.e., ϕ^d_{ni}, ϕ^u_{nij}, and ϕ^v_{ni}, the MFS cut can be constructed as follows:

$$\sum_{i\in\mathcal{N}} \left[\phi^d_{ni} d_{ni} - \phi^v_{ni} C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \right) \right] - \sum_{(i,j)\in\mathcal{A}} \phi^u_{nij} C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \right) \le \eta_n \quad (5.10)$$

In summary, the MFS method can be incorporated into the Benders' decomposition algorithm in the following steps: whenever a subproblem is unbounded, (i) add a feasibility cut to the master problem, (ii) solve model $\mathcal{BMFS}_n$ described in (5.9), (iii) solve $\mathcal{RSP}_n$ and generate MFS cut (5.10), (iv) add MFS cut to the master problem.

5.3 Computational Results for Classical Benders' Algorithms

In this section, we present the computational results for $\mathcal{SNCE}_{TWO}$ that compare the solution time among CPLEX, the classical Benders' decomposition algorithm, and four improvements that we introduced in Section 5.2.

For this purpose, we created 10 instances called $S01$ to $S10$, and solved them using classical Benders' decomposition and CPLEX. Then we separately tested all four improvements presented in Section 5.2. Finally, we tested a procedure named *all-in-one*, which includes all four improvement methods at the same time. To begin with, we present the size of these instances in Table 5.1.

All algorithms have been implemented in C++ using IBM/ILOG CPLEX 12.6

Table 5.1: Data for instances of $\mathcal{SNCE}_{TWO}$

Sample	$\lvert\mathcal{N}\rvert$	$\lvert T.N\rvert$	$\lvert\mathcal{A}\rvert$	No. of Variables		No. of	N
				Real	Integer	Constraints	
S01	5	3	8	104	338	234	13
S02	6	3	16	240	660	420	15
S03	4	2	8	320	960	640	40
S04	6	2	18	126	336	210	7
S05	10	6	26	3,146	8,712	5566	121
S06	3	1	6	1,530	4,590	3,060	255
S07	8	4	18	270	780	510	15
S08	6	2	16	640	1,760	1,120	40
S09	20	9	102	714	1,708	994	7
S10	15	8	58	406	1,022	616	7

Concert Technology. All experiments were performed on a 2.5GHz Intel Core i5-2520M processor with 8.00 GB of RAM running Microsoft Windows 8.1.

Table 5.2 summarizes the solution time for 10 instances using CPLEX, classical Benders' method, four improvement methods, and all-in-one method. It can be seen that the improvement methods can enhance the performance of the classical Benders' decomposition. However, none of them can perform better than CPLEX MIP solver.

Table 5.2: Performance of classical Benders' algorithms for $\mathcal{SNCE}_{TWO}$

Sample	CPLEX	CB		VIE		LP Master		Pareto		MFS		all-in-one	
	Time	Itr.	Time	Itr.	Time	Itr.	Time	Itr.	Time	Itr.	Time	Itr.	Time
S01	0.06	6	0.12	3	0.04	7	0.07	6	0.12	5	0.19	4	0.07
S02	0.09	7	0.19	3	0.07	9	0.18	10	0.29	6	0.15	7	0.18
S03	0.10	9	0.57	6	0.34	9	0.35	12	0.99	10	0.21	10	0.71
S04	0.64	34	2.76	24	2.21	36	1.34	32	1.93	18	1.53	24	0.89
S05	0.29	8	4.62	5	2.14	10	6.71	6	5.23	5	3.74	5	3.36
S06	0.71	13	12.75	6	3.51	14	13.81	10	10.43	11	10.68	12	7.43
S07	0.38	52	53.35	41	23.27	44	2.96	66	47.16	49	62.60	51	4.14
S08	0.29	55	120.98	47	109.41	60	27.02	58	103.56	43	98.23	39	66.96
S09	0.85	107	314.05	137	334.87	89	19.79	92	225.06	96	352.46	73	286.07
S10	0.59	84	658.53	73	617.47	79	152.68	68	640.33	81	740.14	52	328.19

All Solution times are in seconds.

Itr.	Number of iterations
CB	Classical Benders
VIE	Valid Inequality
ALL	Method that uses all 4 improvement simultaneously

Table 5.3 shows the following additional information related to these instances. For all methods, we show the number of feasibility and optimality cuts that have been generated during the Benders' decomposition procedure. We also present the number of valid inequalities that have been added to the master problem. For methods that solve linear master program (MLP and all-in-one), we present the total number of linear and integer programs solved. Note that for Pareto method, the number of Pareto-optimal cuts is equal to the number of optimality cuts, and for the MFS method, the number of MFS cuts is equal to the number of feasibility cuts.

Table 5.3: Detailed data of classical Benders' algorithms for instances of $\mathcal{SNCE}_{TWO}$

Sample	CB		VIE			MLP				Pareto		MFS		all-in-one			
	$\mathcal{O}$	$\mathcal{F}$	#V	$\mathcal{O}$	$\mathcal{F}$	$\mathcal{O}$	$\mathcal{F}$	LP	MIP	$\mathcal{O}$	$\mathcal{F}$	$\mathcal{O}$	$\mathcal{F}$	$\mathcal{O}$	$\mathcal{F}$	LP	MIP
S01	26	49	63	26	13	39	49	6	1	26	49	13	56	39	13	3	1
S02	30	39	44	30	4	45	39	6	3	45	52	45	55	60	11	6	1
S03	120	188	119	120	68	120	188	8	1	160	235	160	196	200	115	9	1
S04	42	106	54	42	57	49	112	30	6	49	103	42	98	56	42	21	3
S05	242	340	362	242	35	363	338	8	2	242	256	121	310	121	236	10	3
S06	1020	1288	1040	1020	328	1275	1283	13	1	510	836	510	998	255	836	15	3
S07	75	313	59	75	256	90	276	34	10	165	350	150	282	210	252	46	5
S08	280	701	199	280	490	360	670	42	18	320	490	320	765	200	384	40	12
S09	70	309	33	70	337	70	314	84	5	63	309	56	386	35	186	73	3
S10	63	319	47	63	248	63	278	69	10	42	256	63	360	49	196	51	6

CB	Classical Benders
VIE	Valid Inequality (#V : Number of valid inequalities added)
MLP	Linear Master problem
$\mathcal{O}$	No. of optimality cuts
$\mathcal{F}$	No. of feasibility cuts
LP	No. of linear programs solved
MIP	No. of mixed-integer programs solved
ALL	Method that uses all 4 improvement simultaneously

There are certain reasons why the classical Benders' decomposition and its improvements cannot outperform CPLEX. In MLP method, after introducing the integrality constraints, the algorithm has to solve an integer program with all feasibility and optimality cuts that have been added to it during the steps in phase one. Solving this integer master program could be time-consuming and may result in a large solution time for Benders' decomposition compared to CPLEX. In Pareto method, the auxiliary linear program (5.8) must be solved for each feasible node of the scenario tree. The equality constraint of (5.8) can be troublesome for large-scale problems, and may result in large solution times for this method compared to CPLEX. The MFS method seems promising in the sense that it includes variable η_n in semi-optimality cuts. However, it comes at the price of solving a 0-1 program and an extra dual subproblem for each infeasible node in the scenario tree. This could be a major drawback of this method.

In the next section, we design a Benders' decomposition-based algorithm that can perform well in finding tight bounds for the problem $\mathcal{SNCE}_{TWO}$.

5.4 New Benders' Decomposition-based Algorithm

In Section 5.3, we showed that the classical Benders' decomposition is not efficient for solving the stochastic network capacity expansion problem $\mathcal{SNCE}_{TWO}$. In this section, we design an algorithm based on Benders' decomposition that can produce tight bounds for models $\mathcal{SNCE}_{TWO}$.

The classical Benders' decomposition methods used for multi-stage stochastic programming problems decompose the problem by nodes of the scenario tree, as explained in Section 5.1. In the new Benders' decomposition-based algorithm, we decompose the problem by each resource, i.e., nodes and arcs of the network. By doing this, the master problem will be a linear program containing flow variables and all subproblems will be integer programs including capacity variables. These integer program subproblems are difficult to solve. However, we exploit the totally unimodular property proved in Theorem 3.1 and Theorem 3.5, and as a result, we solve a linear relaxation of them. In the following, we describe this algorithm for $\mathcal{SNCE}_{TWO}$.

Based on the above explanations, the initial master problem can be modeled as follows:

$$
\begin{aligned}
\text{Min} \ & \sum_{n \in \mathcal{T}} p_n \sum_{(i,j) \in \mathcal{A}} c^u_{ij} x_{nij} + \sum_{(i,j) \in \mathcal{A}} \eta_{ij} + \sum_{i \in \mathcal{N}} \theta_i \\
\text{s.t.} \ & \sum_{(j,i) \in \delta^-_i} x_{nji} - \sum_{(i,j) \in \delta^+_i} x_{nij} = d_{ni} && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& \theta_i \in \mathbb{R}^+ && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& x_{nij}, \ \eta_{ij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}.
\end{aligned}
$$

where η and θ are auxiliary variables corresponding to arcs and nodes, respectively. We define subproblems for each resource of the network. For each arc (i,j), the subproblem is $\mathcal{SP}_{(i,j)}$:

$$
\begin{aligned}
\mathcal{SP}_{(i,j)}(\mathcal{X}) = \text{Min} \ & \sum_{n \in \mathcal{T}} p_n (c^{up}_{nij} u^p_{nij} + c^{us}_{nij} u^s_{nij}) \\
\text{s.t.} \ & \sum_{m \in \bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \geq \left\lceil \frac{x_{nij}}{C^u_{ij}} \right\rceil && \forall n \in \mathcal{T}, \\
& u^p_{nij}, \ u^s_{nij} \in \mathbb{R}^+ && \forall n \in \mathcal{T},
\end{aligned}
$$

and for each node i, the subproblem is $\mathcal{SP}_i$:

$$\begin{aligned}
\mathcal{SP}_i(\mathcal{X}) = \text{Min} \; & \sum_{n \in \mathcal{T}} p_n (c^{vp}_{ni} v^p_{ni} + c^{vs}_{ni} v^s_{ni}) & \\
\text{s.t.} \; & \sum_{m \in \bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \geq \left\lceil \frac{\sum_{(i,j) \in \delta^+_i} x_{nij}}{C^v_i} \right\rceil & \forall n \in \mathcal{T}, \\
& v^p_{ni}, \; v^s_{ni} \in \mathbb{R}^+ & \forall n \in \mathcal{T}.
\end{aligned}$$

Note that for both arc and node subproblems, the right-hand-side is integer-valued, so we can relax the integrality constraints. Furthermore, for any $\mathcal{X}$ returned by the master problem, these subproblems are feasible. Therefore, solving these subproblems will always result in optimality cuts for the master problem. These optimality cuts can be derived from solving the dual of these problems. For each arc (i, j), the arc dual subproblem is:

$$\begin{aligned}
\mathcal{DSP}_{(i,j)}(\mathcal{X}) = \text{Max} \; & \sum_{n \in \mathcal{T}} \left\lceil \frac{x_{nij}}{C^u_{ij}} \right\rceil \pi_{nij} & \\
\text{s.t.} \; & \sum_{m \in \bar{\mathcal{T}}(n)} \pi_{mij} \leq p_n c^{up}_{nij} & \forall n \in \mathcal{T}, \\
& \pi_{nij} \leq p_n c^{us}_{nij} & \forall n \in \mathcal{T}, \\
& \pi_{nij} \in \mathbb{R}^+ & \forall n \in \mathcal{T},
\end{aligned} \tag{5.11}$$

and for each node i, the node dual subproblem is:

$$\begin{aligned}
\mathcal{DSP}_i(\mathcal{X}) = \text{Max} \quad & \sum_{n\in\mathcal{T}} \left\lceil \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}}{C_i^v} \right\rceil \pi_{ni} \\
\text{s.t.} \quad & \sum_{m\in\bar{\mathcal{T}}(n)} \pi_{mi} \le p_n c_{ni}^{vp} && \forall n \in \mathcal{T}, \\
& \pi_{ni} \le p_n c_{ni}^{vs} && \forall n \in \mathcal{T}, \\
& \pi_{ni} \in \mathbb{R}^+ && \forall n \in \mathcal{T},
\end{aligned} \tag{5.12}$$

Solving arc dual subproblem (5.11) will yield optimality cut of form

$$\sum_{n\in\mathcal{T}} \left\lceil \frac{x_{nij}}{C_{ij}^u} \right\rceil \pi_{nij}^* \le \eta_{ij} \tag{5.13}$$

and solving node dual subproblem (5.12) will yield optimality cut of form

$$\sum_{n\in\mathcal{T}} \left\lceil \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}}{C_i^v} \right\rceil \pi_{ni}^* \le \theta_i. \tag{5.14}$$

Obviously, both (5.13) and (5.14) are non-linear cuts and cannot be used in Benders' decomposition procedure. However, by proving the following two theorems, we can modify them to find bounds using Benders' decomposition procedure.

Theorem 5.1. *If we use the cuts*

$$\sum_{n\in\mathcal{T}} \frac{x_{nij}}{C_{ij}^u} \pi_{nij}^* \le \eta_{ij} \tag{5.15}$$

and

$$\sum_{n\in\mathcal{T}} \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}}{C_i^v} \pi_{ni}^* \leq \theta_i \tag{5.16}$$

then we will obtain a lower bound of the original master problem.

Proof. Since $\pi_{nij}^* \geq 0$ and $\pi_{ni}^* \geq 0$, for all $n \in \mathcal{T}$, we have $\left\lceil \frac{x_{nij}}{C_{ij}^u} \right\rceil \geq \frac{x_{nij}}{C_{ij}^u}$ and $\left\lceil \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}}{C_i^v} \right\rceil \geq \frac{\sum_{(i,j)\in\delta_i^+} x_{nij}}{C_i^v}$, which means cut (5.13) dominates cut (5.15) and cut (5.14) dominates cut (5.16). □

Theorem 5.2. *If we use the cuts*

$$\sum_{n\in\mathcal{T}} \left(\frac{x_{nij}}{C_{ij}^u} + 1 \right) \pi_{nij}^* \leq \eta_{ij}, \tag{5.17}$$

and

$$\sum_{n\in\mathcal{T}} \left(\frac{\sum_{(i,j)\in\delta_i^+} x_{nij}}{C_i^v} + 1 \right) \pi_{ni}^* \leq \theta_i, \tag{5.18}$$

then we will obtain an upper bound of the original master problem.

Proof. Similar to the proof of Theorem 5.2, cuts (5.17) and (5.18) dominate cuts (5.13) and (5.14), respectively. □

It is noteworthy that we need to keep two master problems, corresponding to each set of cuts. At the end of the Benders' decomposition procedure, the master problem that used cuts (5.15) and (5.16) will generate a lower bound for the problem $\mathcal{SNCE}_{TWO}$, and the master problem that used cuts (5.17) and (5.18) will generate an upper bound for it. The first master problem, to which cuts (5.15) and (5.16) will be added is $\mathcal{MASTER}_{\mathcal{LB}}$:

$$\begin{aligned}
\text{Min} \quad & \sum_{n\in\mathcal{T}} p_n \sum_{(i,j)\in\mathcal{A}} c^u_{ij} x_{nij} + \sum_{(i,j)\in\mathcal{A}} \eta_{ij} + \sum_{i\in\mathcal{N}} \theta_i \\
\text{s.t.} \quad & \sum_{(j,i)\in\delta^-_i} x_{nji} - \sum_{(i,j)\in\delta^+_i} x_{nij} = d_{ni} && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& \sum_{n\in\mathcal{T}} \frac{x_{nij}}{C^u_{ij}} \pi^*_{nij} \leq \eta_{ij} && \forall (i,j) \in \mathcal{A}, \\
& \sum_{n\in\mathcal{T}} \frac{\sum_{(i,j)\in\delta^+_i} x_{nij}}{C^v_i} \pi^*_{ni} \leq \theta_i && \forall i \in \mathcal{N}, \\
& \theta_i \in \mathbb{R}^+ && \forall i \in \mathcal{N}, \\
& \eta_{ij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \\
& x_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}.
\end{aligned} \tag{5.19}$$

and the second master problem to which cuts (5.17) and (5.18) will be added is $\mathcal{MASTER}_{\mathcal{UB}}$:

$$\begin{aligned}
\text{Min} \quad & \sum_{n\in\mathcal{T}} p_n \sum_{(i,j)\in\mathcal{A}} c^u_{ij} x_{nij} + \sum_{(i,j)\in\mathcal{A}} \eta_{ij} + \sum_{i\in\mathcal{N}} \theta_i \\
\text{s.t.} \quad & \sum_{(j,i)\in\delta^-_i} x_{nji} - \sum_{(i,j)\in\delta^+_i} x_{nij} = d_{ni} && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& \sum_{n\in\mathcal{T}} \left(\frac{x_{nij}}{C^u_{ij}} + 1 \right) \pi^*_{nij} \leq \eta_{ij} && \forall (i,j) \in \mathcal{A}, \\
& \sum_{n\in\mathcal{T}} \left(\frac{\sum_{(i,j)\in\delta^+_i} x_{nij}}{C^v_i} + 1 \right) \pi^*_{ni} \leq \theta_i && \forall i \in \mathcal{N}, \\
& \theta_i \in \mathbb{R}^+ && \forall i \in \mathcal{N}, \\
& \eta_{ij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \\
& x_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}.
\end{aligned} \tag{5.20}$$

Now, let $Z^*_{\mathcal{LB}}$ be the objective function value of the $\mathcal{MASTER}_{\mathcal{LB}}$ in each iteration. Also, suppose Z^L_{ij} and Z^L_i are the objective function value of arc (i,j) and node i subproblems solved using the incumbent values returned by $\mathcal{MASTER}_{\mathcal{LB}}$. Similarly, Z^U_{ij} and Z^U_i are the objective function value of arc (i,j) and node i subproblems solved using the incumbent values returned by $\mathcal{MASTER}_{\mathcal{UB}}$. Given these explanations, we present the new Benders' decomposition-based algorithm formally in Algorithm 7.

Algorithm 7 New Benders' Decomposition-based Algorithm for $\mathcal{SNCE}_{TWO}$

```
 1: Set $UB^{\mathcal{LB}}_{\text{tmp}} = UB^{\mathcal{UB}}_{\text{tmp}} = 0$, $LB = -\infty$ and $UB = +\infty$.
 2: Create $\mathcal{MASTER}_{\mathcal{LB}}$ and $\mathcal{MASTER}_{\mathcal{UB}}$ with no cuts.
 3: repeat
 4:     Solve $\mathcal{MASTER}_{\mathcal{LB}}$ and obtain the incumbent solution $\mathcal{X}_{\mathcal{LB}}$
 5:     Solve $\mathcal{MASTER}_{\mathcal{UB}}$ and obtain the incumbent solution $\mathcal{X}_{\mathcal{UB}}$
 6:     $LB = Z^*_{\mathcal{LB}}$
 7:     Using $\mathcal{X}_{\mathcal{LB}}$, let $UB^{\mathcal{LB}}_{\text{tmp}} = \sum_{n\in\mathcal{T}} p_n \sum_{(i,j)\in\mathcal{A}} c^u_{ij} x_{nij}$
 8:     Using $\mathcal{X}_{\mathcal{UB}}$, let $UB^{\mathcal{UB}}_{\text{tmp}} = \sum_{n\in\mathcal{T}} p_n \sum_{(i,j)\in\mathcal{A}} c^u_{ij} x_{nij}$
 9:     for $(i,j) \in \mathcal{A}$
10:         Using $\mathcal{X}_{\mathcal{LB}}$, solve $\mathcal{DSP}_{ij}$ and get $Z^L_{ij}$. Add cut (5.15) to $\mathcal{MASTER}_{\mathcal{LB}}$
11:         Using $\mathcal{X}_{\mathcal{UB}}$, solve $\mathcal{DSP}_{ij}$ and get $Z^U_{ij}$. Add cut (5.17) to $\mathcal{MASTER}_{\mathcal{UB}}$
12:     end for
13:     for $i \in \mathcal{N}$
14:         Using $\mathcal{X}_{\mathcal{LB}}$, solve $\mathcal{DSP}_i$ and get $Z^L_i$. Add cut (5.16) to $\mathcal{MASTER}_{\mathcal{LB}}$
15:         Using $\mathcal{X}_{\mathcal{UB}}$, solve $\mathcal{DSP}_i$ and get $Z^U_i$. Add cut (5.18) to $\mathcal{MASTER}_{\mathcal{UB}}$
16:     end for
17:     $UB = \text{Min}\left(UB, \sum_{ij} Z^L_{ij} + \sum_i Z^L_i + UB^{\mathcal{LB}}_{\text{tmp}}, \sum_{ij} Z^U_{ij} + \sum_i Z^U_i + UB^{\mathcal{UB}}_{\text{tmp}}\right)$
18:     Gap $= (UB - LB)/LB$.
19: until Gap $< \epsilon$ or no gap improvement for 3 iterations.
```

It is important to note that the structure of the node and arc subproblems guarantees that Algorithm 7 will return an integer-valued feasible solution of $\mathcal{SNCE}_{TWO}$.

5.5 Experimental Results for the New Benders' Method

In this section, we present the experimental results related to the New Benders' method. We compare the solution time of CPLEX MIP solver with that of the new Benders' decomposition-based algorithm. Also, we test the quality of the gap provided by Algorithm 7 by comparing it to the gap that can be obtained from solving the linear relaxation of the $\mathcal{SNCE}_{TWO}$ problem. This gap is called the LP gap.

We do these experiments in two parts. In the first part, we select instances that can be solved by CPLEX MIP solver within an hour. As a result, in addition to comparing the new Benders' decomposition-based algorithm gap and the LP gap, we are able to compare the gap between the solution returned by the new Benders' decomposition-based algorithm and the optimal solution. In the second part, we select large-scale instances that cannot be solved by CPLEX MIP solver within an hour. Then, we verify the quality of gaps provided by the new Benders' decomposition-based algorithm by comparing them to the LP gaps.

All algorithms have been implemented in C++ using IBM/ILOG CPLEX 12.6 Concert Technology. All experiments were performed on a 2.5GHz Intel Core i5-2520M processor with 8.00 GB of RAM running Microsoft Windows 8.1.

For the experiments in the first part, we solved 50 small and medium-sized instances. For simplicity of exposition, we present the data for 15 of them. To begin with, we present the size of the instances that have been created for the first part of the experiments in Table 5.4. Then, we compare the solutions returned by CPLEX

Table 5.4: Data for small and medium-sized instances of $\mathcal{SNCE}_{TWO}$

Sample	$\|\mathcal{N}\|$	$\|T.N\|$	$\|\mathcal{A}\|$	No. of Variables		No. of	N
				Real	Integer	Constraints	
N01	5	2	8	$6,248$	$20,306$	$14,058$	781
N02	15	8	58	406	$1,022$	616	7
N03	10	5	28	196	532	336	7
N04	5	2	8	$74,648$	$242,606$	$167,958$	9,331
N05	4	2	6	$117,186$	$390,620$	$273,434$	19,531
N06	20	12	102	612	$1,464$	852	6
N07	25	10	138	966	$2,282$	$1,316$	7
N08	10	6	56	$1,736$	$4,092$	$2,356$	31
N09	30	18	188	$2,444$	$5,668$	$3,224$	13
N10	12	7	92	644	$1,456$	812	7
N11	14	8	172	$1,204$	$2,604$	$1,400$	7
N12	20	10	136	952	$2,184$	$1,232$	7
N13	10	5	68	$1,020$	$2,340$	$1,320$	15
N14	10	4	46	$1,426$	$3,472$	$2,046$	31
N15	35	19	238	$1,666$	$3,822$	$2,156$	7

with the solutions returned by the new Benders' decomposition-based algorithm. The gap for the new Benders' decomposition-based algorithm is returned by Algorithm 7. To obtain the LP gap, we solve the linear relaxation of the problem to get a lower bound, and then round all integer variables up to get a feasible solution. The upper bound is the objective function value for that feasible solution, and the gap is the difference between the lower bound and upper bound. The results are presented in Table 5.5. The last column shows the percentage of gap improvement resulting from using the new Benders' decomposition-based algorithm over solving the linear relaxation of the problem $\mathcal{SNCE}_{TWO}$. The columns "Dev." show the deviation of the solution provided by two methods, from the optimal solution, in percentage.

Table 5.5: Performance of the new Benders' decomposition-based algorithm for small and medium-sized instances of $\mathcal{SNCE}_{TWO}$

Sample	CPLEX	LP			New Benders				Gap
	Time(s)	Time(s)	Gap(%)	Dev.(%)	Itr.	Time(s)	Gap(%)	Dev.(%)	Imp.(%)
N01	0.24	0.16	0.69	0.24	2	1.83	0.44	0.00	56.95
N02	3.12	0.02	6.26	2.08	11	3.20	4.84	0.63	29.41
N03	4.14	0.06	3.19	0.75	6	0.56	2.92	0.44	9.36
N04	4.21	2.19	0.48	0.24	2	122.23	0.25	0.00	96.68
N05	4.75	2.48	0.13	0.04	2	390.02	0.08	0.00	51.02
N06	5.06	0.03	9.41	1.15	9	4.39	10.00	1.53	−5.91
N07	27.19	0.03	8.11	2.47	11	14.91	6.36	0.69	27.45
N08	49.74	0.05	4.24	0.41	13	12.11	5.07	1.18	−16.38
N09	85.48	0.06	3.61	1.00	5	12.07	2.93	0.28	23.40
N10	111.61	0.02	5.89	2.83	10	5.03	5.42	2.32	8.72
N11	114.70	0.05	9.62	3.50	10	15.98	11.16	4.82	−13.81
N12	187.22	0.03	4.18	0.99	10	11.42	4.62	1.36	−9.47
N13	339.71	0.05	4.76	2.45	10	6.88	4.44	1.98	7.19
N14	1065.32	0.06	8.31	4.82	16	12.94	7.52	3.48	10.54
N15	1245.19	0.05	5.86	1.73	12	48.61	5.28	1.10	10.94

This result shows that although the new Benders' method cannot guarantee gap improvement, in 62% of instances (31 out of 50), it could provide gaps tighter than LP gaps. The average gap improvement is 43.05%. Moreover, there are instances for which, the new Benders' decomposition-based algorithm can provide the optimal solution of the original problem (where Dev. = 0.00).

In second part of our experiments, we consider large-scale instances of $\mathcal{SNCE}_{TWO}$. The purpose of these experiment is to check if the gap provided by the new Benders' method is better than the LP gap, and to observe the time needed to obtain feasible solutions. We present the data related to 12 instances that we created for this part in Table 5.6.

Table 5.6: Data for large-scale instances of $\mathcal{SNCE}_{TWO}$

Sample	$\lvert\mathcal{N}\rvert$	$\lvert T.N\rvert$	$\lvert\mathcal{A}\rvert$	No. of Variables		No. of	N
				Real	Integer	Constraints	
P01	7	3	14	8,190	24,570	16,380	585
P02	10	5	36	36,828	94,116	57,288	1,023
P03	10	6	36	39,348	100,556	61,208	1,093
P04	16	7	88	10,648	25,168	14,520	121
P05	25	14	260	203,060	445,170	242,110	781
P06	25	12	260	8,060	17,670	9,610	31
P07	30	15	364	145,600	315,200	169,600	400
P08	30	20	364	5,096	11,032	5,936	14
P09	40	25	576	23,040	49,280	26,240	40
P10	50	25	962	6,734	14,168	7,434	7
P11	50	30	770	11,550	24,600	13,050	15
P12	100	55	2,030	30,450	63,900	33,450	15

These 12 instances cannot be solved by CPLEX MIP solver within an hour. For some of them, CPLEX even terminates before an hour due to out of memory exception. The results are presented in Table 5.7.

These computational results show that the new benders' decomposition-based algorithm does not guarantee better gaps compared to LP gaps. However, it can improve the gap in many instances. Moreover, although the LP solution time is always better than that of the new Benders' decomposition-based algorithm, in some instances, it worth spending more time to find tighter gaps.

Table 5.7: Performance of the new Benders' decomposition-based algorithm for large-scale instances of $\mathcal{SNCE}_{TWO}$

Sample	LP		New Benders			Gap
	Time(s)	Gap(%)	Itr.	Time(s)	Gap(%)	Imp.(%)
P01	0.204	2.14	3	4.34	1.60	34.42
P02	1.094	0.15	3	20.06	0.11	39.09
P03	1.125	0.12	2	20.05	0.67	−82.77
P04	1.172	2.07	16	219.13	1.05	97.15
P05	17.055	0.82	3	417.77	0.31	169.03
P06	0.219	0.86	3	18.69	0.31	179.15
P07	12.948	0.82	3	366.77	0.85	−2.76
P08	0.109	11.77	10	123.02	9.56	23.07
P09	0.438	1.05	5	247.51	0.59	77.57
P10	0.11	1.77	3	56.66	0.62	186.41
P11	0.36	3.93	11	828.75	2.68	46.86
P12	1.078	0.97	4	876.36	0.68	42.57

5.6 Comparison of the Approximation Algorithm and the New Benders' decomposition-based Algorithm

In Chapter 4, we developed an asymptotically convergent approximation algorithm to solve $\mathcal{SNCE}_{TWO}$. The approximation algorithm provides tight bounds for the original problem. On the other hand, in Section 5.4, we presented a new Benders' decomposition-based algorithm that provide bounds for $\mathcal{SNCE}_{TWO}$, too. In this section, we compare the performance of these two algorithms for $\mathcal{SNCE}_{TWO}$.

For this purpose, we test both algorithm on 15 instances. To begin with, we present the size of the instances in Table 5.8.

Table 5.8: Data for instances of $\mathcal{SNCE}_{TWO}$

Sample	$\|\mathcal{N}\|$	$\|T.N\|$	$\|\mathcal{A}\|$	No. of Variables		No. of	N
				Real	Integer	Constraints	
I01	15	8	60	420	$1,050$	630	7
I02	10	6	36	252	644	392	7
I03	10	4	36	540	$1,380$	840	15
I04	5	2	8	680	$2,210$	$1,530$	85
I05	15	7	96	672	$1,554$	882	7
I06	8	3	38	266	644	378	7
I07	20	10	160	$1,120$	$2,520$	$1,400$	4
I08	30	15	188	$2,444$	$5,668$	$3,224$	13
I09	25	13	138	$5,520$	$13,040$	$7,520$	40
I10	25	15	260	$5,460$	$11,970$	$6,510$	21
I11	30	15	364	$5,096$	$11,032$	$5,936$	14
I12	10	5	36	$3,060$	$7,820$	$4,760$	85
I13	10	6	32	$2,720$	$7,140$	$4,420$	85
I14	4	1	6	$1,530$	$5,100$	$3,570$	225
I15	12	5	34	$8,806$	$23,828$	$15,022$	259

For these experiments, we report the solution gap returned by the approximation algorithm. For instances that can be solved by CPLEX MIP solver within an hour, the gap is the difference between the optimal solution and the solution returned by the approximation algorithm. For large-scale instances that cannot be solved by CPLEX MIP solver within an hour, we calculate the gap using the result of Theorem 4.2. We also report the gap provided by the new Benders' decomposition-based algorithm. When the CPLEX MIP solution is in hand, the gap is the difference between the upper bound returned by the new Benders' decomposition-based algorithm and the optimal solution. Otherwise, it is the difference between the upper bound and the lower bound, both provided by the new Benders' decomposition-based algorithm.

Finally, we report the percentage of gap improvement if we use the new Benders' decomposition-based algorithm over the approximation algorithm. The results are presented in Table 5.9.

All algorithms have been implemented in C++ using IBM/ILOG CPLEX 12.6 Concert Technology. All experiments were performed on a 2.5GHz Intel Core i5-2520M processor with 8.00 GB of RAM running Microsoft Windows 8.1.

Table 5.9: Performance comparison for the approximation algorithm and the new Benders' decomposition-based algorithm for $\mathcal{SNCE}_{TWO}$

Sample	Approximation Alg.		New Benders			Gap
	Time(s)	Gap(%)	Itr.	Time(s)	Gap(%)	Imp.(%)
I01	0.070	0.263	5	0.972	0.286	−8.056
I02	0.029	4.671	9	1.298	0.970	381.353
I03	0.063	1.586	8	1.687	5.035	−68.506
I04	0.078	0.000	3	0.251	0.000	0.000
I05	0.078	2.004	10	6.189	2.375	−15.614
I06	0.047	2.111	7	0.735	3.359	−37.160
I07	0.172	1.039	6	7.674	0.591	75.900
I08	0.328	0.459	5	12.068	0.287	60.201
I09	0.578	0.250	4	13.094	0.060	315.079
I10	0.605	0.822	7	46.181	0.653	25.811
I11	0.859	5.025	10	123.015	9.560	−47.438
I12	0.328	0.097	5	3.984	0.039	146.667
I13	0.354	0.015	3	1.665	0.015	0.000
I14	0.470	0.001	2	0.644	0.060	−97.148
I15	1.676	0.005	2	3.156	0.005	0.000

The result presented in Table 5.9 shows that the new Benders' decomposition-based algorithm cannot outperform the approximation algorithm in solution time. However, it can provide tighter bounds in several cases. It is noteworthy that there is no analytical result for the performance of the new Benders' decomposition-based algorithm, while it has been proved that the approximation algorithm is asymptotically convergent. Therefore, for large number of scenario tree periods, the approximation algorithm is likely to outperform the new Benders' decomposition-based algorithm in all cases (e.g., in I14, $T = 8$). On the other hand, based on our observation during the experiments, it seems that the new Benders' decomposition-based algorithm is more efficient that the approximation algorithm in memory usage. When the memory is limited, there are cases that can be solved by the new Benders' decomposition-based algorithm, and cannot be solved by approximation algorithm due to out of memory exception.

Chapter 6

Stochastic Network Capacity Expansion Problem with Budget Constraint

In the stochastic network capacity expansion models $\mathcal{SNCE}_{TWO}$ and $\mathcal{SNCE}_{THREE}$ presented in Chapter 4, there is no restriction on the budget that the decision maker can spend on expanding the capacity of arcs and nodes. Although the solution of such models give us an estimation on the required budget, in real world problems, decision makers often have to deal with budget limitations. In this chapter, we consider the stochastic network capacity expansion problem with budget restriction.

6.1 The Formulation

In order to formulate the budget version of $\mathcal{SNCE}_{TWO}$, we assume that the budget restriction is only applicable to the permanent capacity acquisition, and spot market capacity acquisition does not have budget restriction. This assumption guarantees that the problem is always feasible. Furthermore, we assume that the imposed budgets are the same for all the nodes in the same period of the scenario tree. Now, let B_{t_n} denote the budget for all nodes of the scenario tree which are in the same time period t_n. Using the same notations from Section 4.2, we can state the formulation for the network capacity expansion problem with budget constraints as follows:

Model $\mathcal{BSNCE}_{TWO}$:

$$
\begin{aligned}
\text{Min } & \sum_{n\in\mathcal{T}} p_n \left[\sum_{(i,j)\in\mathcal{A}} (c^u_{ij} x_{nij} + c^{up}_{nij} u^p_{nij} + c^{us}_{nij} u^s_{nij}) + \sum_{i\in\mathcal{N}} (c^{vp}_{ni} v^p_{ni} + c^{vs}_{ni} v^s_{ni}) \right] \\
\text{s.t. } & \sum_{(i,j)\in\mathcal{A}} c^{up}_{nij} u^p_{nij} + \sum_{i\in\mathcal{N}} c^{vp}_{ni} v^p_{ni} \le B_{t_n} && \forall n \in \mathcal{T}, \\
& \sum_{(j,i)\in\delta^-_i} x_{nji} - \sum_{(i,j)\in\delta^+_i} x_{nij} = d_{ni} && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& x_{nij} \le C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \right) && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \\
& \sum_{(i,j)\in\delta^+_i} x_{nij} \le C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \right) && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& x_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \\
& u^p_{nij},\ u^s_{nij} \in \mathbb{Z}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \\
& v^p_{ni},\ v^s_{ni} \in \mathbb{Z}^+ && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}.
\end{aligned}
$$

Similar to $\mathcal{SNCE}_{TWO}$, this problem is also an NP-hard problem and the presence of budget constraints will make it even harder to solve. Some preliminary computational results showed that the solution time for $\mathcal{BSNCE}_{TWO}$ could be up to 10 times more than the solution time of $\mathcal{SNCE}_{TWO}$ with the same parameters.

The approximation algorithm presented for $\mathcal{SNCE}_{TWO}$ in Section 4.3 cannot solve $\mathcal{BSNCE}_{TWO}$ because the budget constraint will subvert the decomposable structure of the model $\mathcal{SNCE}_{TWO}$. Motivated by this observations, we develop a Lagrangian relaxation method for $\mathcal{BSNCE}_{TWO}$ by relaxing the budget constraint.

6.2 Lagrangian Relaxation

Suppose λ_n are the Lagrange multipliers for the budget constraints. Then the Lagrangian relaxation subproblem of $\mathcal{BSNCE}_{TWO}$ can be written as follows:
Model $\mathcal{BSNCE}_{TWO} - \mathcal{LR}_\lambda$

$$\begin{aligned}
\text{Min } & \sum_{n\in\mathcal{T}} p_n \left[\sum_{(i,j)\in\mathcal{A}} (c^u_{ij} x_{nij} + c^{up}_{nij} u^p_{nij} + c^{us}_{nij} u^s_{nij}) + \sum_{i\in\mathcal{N}} (c^{vp}_{ni} v^p_{ni} + c^{vs}_{ni} v^s_{ni}) \right] \\
& + \sum_{n} \lambda_n \left(\sum_{(i,j)\in\mathcal{A}} c^{up}_{nij} u^p_{nij} + \sum_{i\in\mathcal{N}} c^{vp}_{ni} v^p_{ni} - B_{t_n} \right) \\
\text{s.t. } & \sum_{(i,j)\in\delta^+_i} x_{nij} - \sum_{(i,j)\in\delta^-_i} x_{nji} = d_{ni} && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& x_{nij} \le C^u_{ij} \left(\sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} + u^s_{nij} \right) && \forall ij \in \mathcal{A}, \forall n \in \mathcal{T}, \\
& \sum_{(i,j)\in\delta^+_i} x_{nij} \le C^v_i \left(\sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} + v^s_{ni} \right) && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\
& x_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \\
& u^p_{nij},\ u^s_{nij} \in \mathbb{Z}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \\
& v^p_{ni},\ v^s_{ni} \in \mathbb{Z}^+ && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}.
\end{aligned}$$

A general discussion of Lagrangian relaxation method can be found in Guignard (2003). For brevity of exposition, in the following, we only discuss several key tune-ups that improves the efficiency of the Lagrangian relaxation method.

Initial Multipliers: Lagrangian relaxation methods are sensitive to the choice of initial multipliers. We obtained the initial multipliers by solving the linear relaxation of $\mathcal{BSNCE}$ and setting the initial multipliers equal to the dual value of the budget constraint.

Multiplier Updating: In order to solve the Lagrangian dual problem, we use the well-known sub-gradient method (Fisher (2004)). In each iteration of sub-gradient method, the sub-gradient vector is $\sum_{(i,j)\in\mathcal{A}} c^{up}_{nij} u^{p}_{nij} + \sum_{i\in\mathcal{N}} c^{vs}_{ni} v^{s}_{ni} - B_{t_n}$ and the step size in iteration k is

$$\mu_k = \theta_k \frac{UB^k - z(\lambda^k_n)}{\| \sum_{(i,j)\in\mathcal{A}} c^{up}_{nij} u^{p}_{nij} + \sum_{i\in\mathcal{N}} c^{vs}_{ni} v^{s}_{ni} - B_{t_n} \|^2}$$

where $z(\lambda^k)$ is the value of Lagrangian function calculated in iteration k, UB^k is the best upper bound found so far for iteration k, and θ^k is a constant in iteration k, initialized to 2 and will be divided by 2 if two consecutive iterations cannot improve the lower bound. As a result, the multipliers will be updated as follows:

$$\lambda^{k+1}_n = \text{Max}\left\{ \lambda^k_n + \mu_k \left(\sum_{(i,j)\in\mathcal{A}} c^{up}_{nij} u^{p}_{nij} + \sum_{i\in\mathcal{N}} c^{vs}_{ni} v^{s}_{ni} - B_{t_n} \right), 0 \right\}$$

Initial Upper Bound: A key for the success of the Lagrangian relaxation method is to find a good upper bound fast. For this purpose, we first solve the minimum cost flow problem without any capacity variable. Then, we use the obtained flow variables to calculate the values of permanent capacity variables u^p and v^p. These values must be rounded up to satisfy integrality constraints. Then, we check the budget constraints with these integer permanent capacity variables. If they are satisfied, we already have a feasible solution and we can calculate the upper bound. If some budget constraints are not satisfied, say for node n, then we reduce the value of permanent capacity variables by the proportion that the budget constraint is violated (if the new permanent capacity variables are not integers, we round them down). This will affect the feasibility of capacity constraints related to all nodes $m \in \bar{\mathcal{T}}(n)$. Therefore, we add the reduced permanent capacity as spot market capacity to all nodes $m \in \bar{\mathcal{T}}(n)$ in order to preserve the feasibility. This procedure is explained in more details in Algorithm 8. Note that the whole procedure performs local operations at each node, and is very efficient.

Upper Bound Updating: To update the upper bound in each iteration of the Lagrangian relaxation procedure, we use a method similar to the one that we use to find the initial upper bound. This method checks the budget constraints after each iteration. If they are satisfied, it updates the upper bound if a better upper bound is found. Otherwise, it will reduce permanent capacity variables by the proportion that the budget constraint is violated, and increase the same value in the spot market capacity to preserve the feasibility. This procedure is explained in Algorithm 9.

Stopping Criteria: There are several criteria that can be used to terminate the Lagrangian relaxation procedure, such as running a pre-specified number of iterations, reaching a pre-specified minimum, and reaching a predefined gap between lower bound and lower bound. We used the last one in our implementation:

$$\frac{UB^k - z(\lambda^k)}{z(\lambda^k)} < \gamma.$$

Algorithm 8 Initial Upper Bound Construction Algorithm for Lagrangian Relaxation

```
 1: Solve the minimum cost flow problem for $x_{nij}$:
        $$\begin{aligned} &\text{Min} \sum_{n\in\mathcal{T}} p_n \sum_{(i,j)\in\mathcal{A}} c^u_{ij} x_{nij} \\ &\text{s.t.} \sum_{(j,i)\in\delta_i^-} x_{nji} - \sum_{(i,j)\in\delta_i^+} x_{nij} = d_{ni} && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}, \\ &x_{nij} \in \mathbb{R}^+ && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}. \end{aligned}$$
 2: for $n = 1$ to $N$,
 3:     Using $x_{nij}$ from step 1, solve the following system of equations to get
        $u^p$ and $v^p$:
        $$\begin{aligned} x_{nij} &= C^u_{ij} \sum_{m\in\bar{\mathcal{P}}(n)} u^p_{mij} && \forall (i,j) \in \mathcal{A}, \forall n \in \mathcal{T}, \\ \sum_{(i,j)\in\delta_i^+} x_{nij} &= C^v_i \sum_{m\in\bar{\mathcal{P}}(n)} v^p_{mi} && \forall i \in \mathcal{N}, \forall n \in \mathcal{T}. \end{aligned}$$
 4: end for
 5: $u^p_{nij} \leftarrow \lceil u^p_{nij} \rceil$,   $v^p_{ni} \leftarrow \lceil v^p_{ni} \rceil$ ,   $u^s_{nij} \leftarrow 0$ ,   $v^s_{ni} \leftarrow 0$
 6: for $n = 1$ to $N$,
 7:     $R_n \leftarrow \sum_{(i,j)\in\mathcal{A}} c^{up}_{nij} u^p_{nij} + \sum_{i\in\mathcal{N}} c^{vp}_{ni} v^p_{ni}$
 8:     if $R_n > B_{t_n}$
 9:         $u_{\text{temp}} \leftarrow u^p_{nij}$,   $u^p_{nij} \leftarrow \lfloor u^p_{nij} \times B_{t_n}/R_n \rfloor$
10:         $v_{\text{temp}} \leftarrow v^p_{ni}$,   $v^p_{ni} \leftarrow \lfloor v^p_{ni} \times B_{t_n}/R_n \rfloor$
11:         for $m \in \bar{\mathcal{T}}(n)$
12:             $u^s_{nij} \leftarrow u_{\text{temp}} - u^p_{nij}$
13:             $v^s_{ni} \leftarrow v_{\text{temp}} - v^p_{ni}$
14:         end for
15:     end if
16: end for
17: $UB \leftarrow \sum_{n\in\mathcal{T}} p_n \left[ \sum_{(i,j)\in\mathcal{A}} \left( c^{us}_{nij} u^s_{nij} + c^u_{ij} x_{nij} \right) + \sum_{i\in\mathcal{N}} c^{vs}_{ni} v^s_{ni} \right]$
```

Algorithm 9 Upper Bound Updating Algorithm for Lagrangian Relaxation

```
1: for $n = 1$ to $N$,
2:     $R_n \leftarrow \sum_{(i,j)\in\mathcal{A}} c^{up}_{nij} u^p_{nij} + \sum_{i\in\mathcal{N}} c^{vs}_{ni} v^s_{ni}$
3:     if $R_n > B_{t_n}$
4:         $u_{\text{temp}} \leftarrow u^p_{nij}, \quad u^p_{nij} \leftarrow \lfloor u^p_{nij} \times B_{t_n}/R_n \rfloor$
5:         $v_{\text{temp}} \leftarrow v^p_{ni}, \quad v^p_{ni} \leftarrow \lfloor v^p_{ni} \times B_{t_n}/R_n \rfloor$
6:         for $m \in \bar{\mathcal{T}}(n)$
7:             $u^s_{nij} \leftarrow u^s_{nij} + u_{\text{temp}} - u^p_{nij}$
8:             $v^s_{ni} \leftarrow v^s_{ni} + v_{\text{temp}} - v^p_{ni}$
9:         end for
10:    end if
11: end for
12: $UB \leftarrow \sum_{n\in\mathcal{T}} p_n \left[ \sum_{(i,j)\in\mathcal{A}} \left( c^{us}_{nij} u^s_{nij} + c^u_{ij} x_{nij} \right) + \sum_{i\in\mathcal{N}} c^{vs}_{ni} v^s_{ni} \right]$
```

6.3 Experimental Results for the Lagrangian Relaxation

We present the experimental results for the Lagrangian relaxation method designed for $\mathcal{BSNCE}_{TWO}$. We compare the solution time by Lagrangian relaxation method with that of CPLEX MIP solver. We also report the gap between the upper bound and the lower bound given by the Lagrangian relaxation. For these experiments, we set $\gamma = 0.02$ and set a limit of an hour for CPLEX running time.

All algorithms have been implemented in GAMS 24.2.2 using IBM/ILOG CPLEX 12.6. All experiments were performed on a 2.79GHz AMD Opteron processor with 64.00 GB of RAM running Microsoft Windows Server 2008 R2.

Table 6.1 presents the data for 10 instances of $\mathcal{BSNCE}_{TWO}$ that have been created for these experiments.

Table 6.1: Data for instances of $\mathcal{BSNCE}_{TWO}$

Sample	$\lvert\mathcal{N}\rvert$	$\lvert T.N\rvert$	$\lvert\mathcal{A}\rvert$	No. of Variables		No. of	Transient	N
				Real	Integer	Constraints	Nodes	
L01	10	5	18	126	392	273	3	7
L02	10	4	18	270	840	585	6	15
L03	10	6	27	837	2,294	1,488	4	31
L04	12	7	27	837	2,418	1,612	10	31
L05	20	10	76	532	1,344	819	11	7
L06	10	5	18	558	1,736	1,209	5	31
L07	10	6	18	1,134	3,528	2,457	6	63
L08	20	10	76	1,140	2,880	1,755	12	15
L09	20	12	76	2,356	5,952	3,627	5	31
L10	20	8	76	4,788	12,096	7,371	6	63

The results are summarized in Table 6.2. In this table, ”+3600” means that CPLEX could not solve the problem in an hour. As a result, the percentage of time for $S05$ to $S10$ are actually less than the reported number.

Table 6.2: CPLEX and Lagrangian relaxation algorithm comparison for $\mathcal{BSNCE}_{TWO}$

Sample	CPLEX Time(s)	LR Time(s)	% of Time	Gap(%)	Iterations
L01	398.41	48.63	12.21	0.91	41
L02	545.52	86.23	15.81	0.91	34
L03	629.35	191.07	30.35	1.98	20
L04	1,845.07	102.32	5.55	1.94	48
L05	+3,600.00	145.63	4.05	0.62	23
L06	+3,600.00	193.98	5.39	1.84	66
L07	+3,600.00	285.46	7.93	1.92	87
L08	+3,600.00	456.25	12.67	1.37	44
L09	+3,600.00	608.53	16.90	2.00	20
L10	+3,600.00	3,276.42	91.01	1.96	32

Table 6.2 shows that the Lagrangian relaxation is more efficient than the CPLEX MIP solver in solution time and it is capable of providing a feasible solution within 2% of optimality in a reasonable time. One limitation for this algorithm is that as the size of the problem gets very large, a large-scale NP-hard problem must be solved in each iteration of the Lagrangian relaxation algorithm. This issue can be addressed by incorporating that approximation algorithm into the Lagrangian relaxation procedure. This requires some minor changes in the way that the lower bound and the upper bound are updated in each iteration.

Chapter 7

Conclusions and Future Research Directions

In this chapter, we summarize the major contributions of this dissertation and possible future research directions.

7.1 Conclusions

In this dissertation, we studied capacity expansion problem of both single resource and multiple resource models in the presence of permanent, spot market, and contract capacity for acquisition. This problem has applications in areas such as transportation and logistics, telecommunication industry, service industry, water distribution, and manufacturing. A scenario tree approach has been used in order to handle the data uncertainty of the models, and a multi-stage stochastic programming approach has been used to formulate the problems.

To begin with, we presented multi-stage stochastic programming formulation for two versions of single resource capacity expansion problems with different sources of capacity. We developed polynomial-time primal and dual algorithms that can solve them efficiently. Both algorithms can outperform CPLEX LP solver.

Using the multi-stage stochastic programming approach, we then presented a formulation for stochastic network capacity expansion problem with different sources of capacity. This formulating has been done in the context of a min-cost network flow problem and resulted in a large-scale MIP model which is notoriously difficult to solve. Exploring the decomposable structure of the problem led to identifying the previously studied single resource capacity expansion problems as subproblems of the network capacity expansion problem. These subproblems correspond to each resource (e.g., node, arc) of the network. Exploiting the polynomial-time algorithms developed to solve these subproblems, we designed an approximation algorithm to solve the stochastic network capacity expansion problem. The approximation algorithm was proved to be asymptotically convergent to the optimal solution. The presented experimental results confirmed that the approximation algorithm is efficient compared to CPLEX, and asymptotically convergent to the optimal solution.

Moreover, decomposition algorithms have been developed to solve the stochastic network capacity expansion problem. First, a classical Benders' decomposition procedure has been presented for the problem. Then, several improvement techniques were introduced to speed up the classical Benders' decomposition. The experimental results showed that these algorithms cannot outperform CPLEX MIP solver. As a result, a new Benders' decomposition-based algorithm has been presented. This new method can provide a lower bound for the network capacity expansion problem and use a heuristic method to construct an upper bound. The experiments showed that the gap provided by this method can be better than the LP gap. The experimental results for both the approximation algorithm and the new Benders' decomposition-based algorithm confirm that both algorithms can outperform CPLEX MIP solver.

Finally, an extension of network capacity expansion problem with budget restriction imposed on permanent capacity expansion cost has been studied. Introducing the budget constraint to the problem subverts the decomposable structure that had been used to design the approximation algorithm. Therefore, a Lagrangian relaxation method has been designed to solve the stochastic network capacity expansion problem with budget constraint. Computational results showed that the proposed approach is more efficient than CPLEX MIP solver.

7.2 Future Research Directions

Studies in this dissertation lead to several open questions in stochastic capacity expansion context. For all models that we considered in this dissertation, we assumed there is no fixed-charge for buying the capacity. Potentially, a study on the fixed-charge version of our presented model could be done. The designed algorithms will fail for fixed-charge models and new ones must be developed. The stochastic network capacity expansion formulation is defined on a min-cost network flow problem. Alternately, one could study capacity expansion on a multi-commodity flow model which has several applications in areas such as railroad transportation. In this case, the approximation algorithm would fail because multiple MIP subproblems must be solved. Therefore, a different solution methodology must be developed. It would be useful and interesting to study the stochastic capacity expansion in the blocking problem (Barnhart et al. (2000); Ahuja et al. (2007)), and to develop efficient algorithm to solve such NP-hard problem. For the Lagrangian relaxation algorithm designed for the stochastic network capacity expansion with budget constraints, one could study more efficient heuristics to find the initial feasible solution and to update the upper bound in each iteration.

Bibliography

Ahmed, S., King, A., and Parija, G. (2003). A multi-stage stochastic integer programming approach for capacity expansion under uncertainty. *Journal of Global Optimization*, 26(1):3–24.

Ahmed, S. and Sahinidis, N. V. (2003). An approximation scheme for stochastic integer programs arising in capacity expansion. *Operations Research*, 51(3):461–471.

Ahuja, R. K., Batra, J. L., Gupta, S. K., and Punnen, A. P. (1996). Optimal expansion of capacitated transshipment networks. *European Journal of Operational Research*, 89(1):176–184.

Ahuja, R. K., Jha, K. C., and Liu, J. (2007). Solving real-life railroad blocking problems. *Interfaces*, 37(5):404–419.

Amaldi, E. and Kann, V. (1995). The complexity and approximability of finding maximum feasible subsystems of linear relations. *Theoretical Computer Science*, 147(1-2):181–210.

Aneja, Y. P. and Chaouch, B. A. (1993). Capacity expansion and contraction of a facility with economies of scale. *INFOR: Information Systems and Operational Research*, 31(4):247–260.

Atamtürk, A. and Hochbaum, D. S. (2001). Capacity acquisition, subcontracting, and lot sizing. *Management Science*, 47(8):1081–1100.

Atamtürk, A. and Zhang, M. (2007). Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673.

Balakrishnan, A., Magnanti, T. L., Shulman, A., and Wong, R. T. (1991). Models for planning capacity expansion in local access telecommunication networks. *Annals of Operations Research*, 33(4):237–284.

Balakrishnan, A., Magnanti, T. L., and Wong, R. T. (1995). A decomposition algorithm for local access telecommunications network expansion planning. *Operations Research*, 43(1):58–76.

Bansal, P. P. and Jacobsen, S. E. (1975). An algorithm for optimizing network flow capacity under economies of scale. *Journal of Optimization Theory and Applications*, 15(5):565–586.

Barnhart, C., Jin, H., and Vance, P. H. (2000). Railroad blocking: A network design application. *Operations Research*, 48(4):603–614.

Bean, J. C., Higle, J. L., and Smith, R. L. (1992). Capacity expansion under stochastic demands. *Operations Research*, 40(3-supplement-2):S210–S216.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische mathematik*, 4(1):238–252.

Berman, O. and Ganz, Z. (1994). The capacity expansion problem in the service industry. *Computers & Operations Research*, 21(5):557–572.

Bhatnagar, S., Fernández-Gaucherand, E., Fu, M. C., He, Y., and Marcus, S. I. (1998). A Markov decision process model for capacity expansion and allocation (I). *IEEE Conference On Decision and Control*, 2(1):1380–1385.

Bienstock, D., Chopra, S., Günlük, O., and Tsai, C. Y. (1998). Minimum cost capacity installation for multicommodity network flows. *Mathematical Programming*, 81(2):177–199.

Bienstock, D. and Günlük, O. (1996). Capacitated network design-polyhedral structure and computation. *INFORMS Journal on Computing*, 8(3):243–259.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

Chang, S. G. and Gavish, B. (1993). Telecommunications network topological design and capacity expansion: Formulations and algorithms. *Telecommunication Systems*, 1(1):99–131.

Chen, Z. L., Li, S., and Tirupati, D. (2002). A scenario-based stochastic programming approach for technology and capacity planning. *Computers & Operations Research*, 29(7):781–806.

Christie, R. M. E. and David Wu, S. (2002). Semiconductor capacity planning: stochastic modeling and computational studies. *IIE Transactions*, 34(2):131–143.

Christofides, N. and Brooker, P. (1974). Optimal expansion of an existing network. *Mathematical Programming*, 6(1):197–211.

Doulliez, P. J. and Rao, M. R. (1971). Capacity of a network with increasing demands and arcs subject to failure. *Operations Research*, 19(4):905–915.

Doulliez, P. J. and Rao, M. R. (1975). Optimal network capacity planning: A shortest-path scheme. *Operations Research*, 23(4):810–818.

Erkoc, M. and Wu, S. D. (2009). Managing high-tech capacity expansion via reservation contracts. *Production and Operations Management*, 14(2):232–251.

Fisher, M. L. (2004). The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Management Science*, 50(12 supplement):1861–1871.

Freidenfelds, J. (1981). *Capacity expansion: analysis of simple models with applications*. Elsevier, New York, NY: North Holland.

Fulkerson, D. R. (1959). Increasing the capacity of a network: The parametric budget problem. *Management Science*, 5(4):472–483.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

Guignard, M. (2003). Lagrangian Relaxation. *Top*, 11(2):151–200.

Günlük, O. (1999). A branch-and-cut algorithm for capacitated network design problems. *Mathematical Programming*, 86(1):17–39.

Han, X. and Makino, K. (2010). *Approximation and Online Algorithms*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Hsu, N. S., Cheng, W. C., Cheng, W. M., Wei, C. C., and Yeh, W. G. (2008). Optimization and capacity expansion of a water distribution system. *Advances in Water Resources*, 31(5):776–786.

Huang, K. and Ahmed, S. (2008). On a multi-stage stochastic programming model for inventory planning. *INFOR: Information Systems and Operational Research*, 46(3):155–163.

Huang, K. and Ahmed, S. (2009). The value of multistage stochastic programming in capacity planning under uncertainty. *Operations Research*, 57(4):893–904.

Inderfurth, K. and Kelle, P. (2011). Capacity reservation under spot market price uncertainty. *International Journal of Production Economics*, 133(1):272–279.

Inderfurth, K., Kelle, P., and Kleber, R. (2013). Dual sourcing using capacity reservation and spot market: Optimal procurement policy and heuristic parameter determination. *European Journal of Operational Research*, 225(2):298–309.

Kall, P. and Wallace, S. W. (1994). *Stochastic programming.* John Wiley and Sons Ltd.

Karabuk, S. and Wu, S. (2003). Coordinating strategic capacity planning in the semiconductor industry. *Operations Research*, 51(6):839–849.

Karoonsoontawong, A. (2010). Integrated network capacity expansion and traffic signal optimization problem: robust bi-level dynamic formulation. *Networks and Spatial Economics*, 10(4):525–550.

Laguna, M. (1998). Applying robust optimization to capacity expansion of one location in telecommunications with demand uncertainty. *Management Science*, 44(11-Part-2):101–110.

Lee, C. and Kang, H. (2000). Cell planning with capacity expansion in mobile communications: a tabu search approach. *IEEE Transactions on Vehicular Technology*, 49(5):1678–1691.

Lee, D. H., Lim, S. K., Lee, G. C., Jun, H. B., and Kim, Y. D. (1997). Multi-period part selection and loading problems in flexible manufacturing systems. *Computers & Industrial Engineering*, 33(3):541–544.

Leung, J. M. Y., Magnanti, T. L., and Singhal, V. (1990). Routing in point-to-point delivery systems: formulations and solution heuristics. *Transportation Science*, 24(4):245–260.

Levin, Y., Nediak, M., and Topaloglu, H. (2012). Cargo capacity management with allotments and spot market demand. *Operations Research*, 60(2):351–365.

Liu, J., Ahuja, R. K., and Sahin, G. (2007). Optimal network configuration and capacity expansion of railroads. *Journal of the Operational Research Society*, 59(7):911–920.

Logendran, R. and Puvanunt, V. (1997). Duplication of machines and subcontracting of parts in the presence of alternative cell locations. *Computers & Industrial Engineering*, 33(1):235–238.

Logendran, R. and Ramakrishna, P. (1997). A methodology for simultaneously dealing with machine duplication and part subcontracting in cellular manufacturing systems. *Computers & Operations Research*, 24(2):97–116.

Lulli, G. and Sen, S. (2004). A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems. *Management Science*, 50(6):786–796.

Luna, H. and Mahey, P. (2000). Bounds for global optimization of capacity expansion and flow assignment problems. *Operations Research Letters*, 26(5):211–216.

Luss, H. (1982). Operations research and capacity expansion problems: A survey. *Operations Research*, 30(5):907–947.

Luss, H. (1984). Capacity expansion planning for a single facility product line. *European Journal of Operational Research*, 18(1):27–34.

Magnanti, T. L., Mirchandani, P., and Vachani, R. (1995). Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1):142–157.

Magnanti, T. L. and Wong, R. T. (1981). Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria. *Operations Research*, 29(3):464–484.

Magnanti, T. L. and Wong, R. T. (1984). Network design and transportation planning: Models and algorithms. *Transportation Science*, 18(1):1–55.

Marín, A. and Jaramillo, P. (2008). Urban rapid transit network capacity expansion. *European Journal of Operational Research*, 191(1):45–60.

McDaniel, D. and Devine, M. (1977). A modified Benders' partitioning algorithm for mixed integer programming. *Management Science*, 24(3):312–319.

Mudchanatongsuk, S., Ordóñez, F., and Liu, J. (2007). Robust solutions for network design under transportation cost and demand uncertainty. *Journal of the Operational Research Society*, 59(5):652–662.

Ordóñez, F. and Zhao, J. (2007). Robust capacity expansion of network flows. *Networks*, 50(2):136–145.

Oren, S., Jiang, J., Wu, D., Kleindorfer, P. R., and Sun, Y. (2005). Optimal capacity expansion in the presence of capacity options. *Decision Support Systems*, 40(3):553–561.

Pimentel, B. S., Mateus, G. R., and Almeida, F. A. (2013). Stochastic capacity planning and dynamic network design. *International Journal of Production Economics*, 145(1):139–149.

Pratikakis, N. E. (2010). Strategic capacity decision-making in a stochastic manufacturing environment using real-time approximate dynamic programming. *Naval Research Logistics*, 57(3):211–224.

Rajagopalan, S., Singh, M. R., and Morton, T. E. (1998). Capacity expansion and replacement in growing markets with uncertain technological breakthroughs. *Management Science*, 44(1):12–30.

Rasekh, L. and Desrosiers, J. (2010). Solving multi-stage stochastic in-house production and outsourcing planning by two-level decomposition. *International Journal of Mathematics in Operational Research*, 2(2):129–150.

Riis, M. and Andersen, K. A. (2004). Multiperiod capacity expansion of a telecommunications connection with uncertain demand. *Computers & Operations Research*, 31(9):1427–1436.

Rocklin, S. M., Kashper, A., and Varvaloucas, G. C. (1984). Capacity expansion/contraction of a facility with demand augmentation dynamics. *Operations Research*, 32(1):133–148.

Ruszczyski, A. and Shapiro, A. (2003). Stochastic programming models. *Handbooks in operations research and Management Science*, 10:1–64.

Saharidis, G. K. D. and Ierapetritou, M. G. (2010). Improving benders decomposition using maximum feasible subsystem (MFS) cut generation strategy. *Computers & Chemical Engineering*, 34(8):1237–1245.

Saniee, I. (1995). An efficient algorithm for the multiperiod capacity expansion of one location in telecommunications. *Operations Research*, 43(1):187–190.

Sen, S., Doverspike, R. D., and Cosares, S. (1994). Network planning with random demand. *Telecommunication Systems*, 3(1):11–30.

Shapiro, A. (2006). On complexity of multistage stochastic programs. *Operations Research Letters*, 34(1):1–8.

Singh, K. J., Philpott, A. B., and Kevin Wood, R. (2009). Dantzig-Wolfe decomposition for solving multistage stochastic capacity-planning problems. *Operations Research*, 57(5):1271–1286.

Tarhan, B. and Grossmann, I. E. (2008). A multistage stochastic programming approach with strategies for uncertainty reduction in the synthesis of process networks with uncertain yields. *Computers & Chemical Engineering*, 32(4):766–788.

Van Mieghem, J. A. (2003). Commissioned paper: Capacity management, investment, and hedging: Review and recent developments. *Manufacturing & Service Operations Management*, 5(4):269–302.

Van Slyke, R. M. and Wets, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17(4):638–663.

Wolsey, L. A. and Nemhauser, G. L. (1988). *Integer and combinatorial optimization*. Wiley-Interscience, New York, 18 edition.

Zhang, F., Roundy, R. O., Cakanyildirim, M., and Huh, W. T. (2004). Optimal capacity expansion for multi-product, multi-machine manufacturing systems with stochastic demand. *IIE Transactions*, 36(1):23–36.