# Stochastic Integer Programming: Decomposition Methods and Industrial Applications

by

Cheng (Marshal) Wang

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Department of Mechanical and Industrial Engineering
University of Toronto

# Abstract

Stochastic Integer Programming:

Decomposition Methods and Industrial Applications

Cheng (Marshal) Wang

Doctor of Philosophy

Department of Mechanical and Industrial Engineering

University of Toronto

2014

Many practical problems from industry that contain uncertain demands, costs and other quantities are challenging to solve. Stochastic Mixed Integer Programs (SMIPs) have become an emerging tool to incorporate uncertainty in optimization problems. The stochastic and mixed integer nature of SMIPs makes them very challenging to solve. Decomposition methods have been developed to solve various practical problems modeled as large-scale SMIPs.

In the thesis, we propose a scenario-wise decomposition method, the Dynamic Dual Decomposition method ($D^3$ method), to decompose large-scale SMIPs in order to solve practical facility location problems more efficiently. The Lagrangian bounds are dynamically determined. We also consider alternative ways to represent non-anticipativity conditions to improve computational performance. The $D^3$ method efficiently solved moderate and large sized instances whose deterministic equivalent problem could not be solved or solved much slower by a state-of-the-art commercial solver. Three-stage models are also studied and solved by the $D^3$ method, which is found to be effective as well.

We combine the $D^3$ method and Column Generation approach to solve a stochastic version of the set packing problem. The proposed method is quite effective in numerical experiments and outperforms the commercial solver dramatically in most cases. We study the sensitivity of different density of patterns and find our proposed method is more robust than solving the whole problem by the commercial solver or by implementing the conventional column generation method directly.

We use the $D^3$ method as a framework and combine it with Benders Decomposition Method (BDM) to solve the Stochastic Multi-plant Facility Location Problem. The mathematical formulation of the model is a two-stage SMIP problem with integer first stage variables and mixed integer second stage variables. The integer variables in both stages cause the large-scale SMIP model to be intractable. Some strategies for accelerating BDM have been developed in solving scenario subproblems which are decomposed by the $D^3$ method. We develop the aggregation method to aggregate scenarios in solving decomposed Benders Dual subproblems and approximate the solution of the original problems. Our computational experiments on benchmark data and randomly generated data shows that the proposed method can solve large-sized problems more efficiently than conventional Benders Decomposition or a commercial solver. The computational comparisons also show that the aggregation method can reduce the number of scenarios in large problems and obtain approximately optimal solutions in much shorter computational time.

# Dedication

***To my family, especially to my son Luke Wang.***

# Acknowledgements

Many people have contributed, both directly and indirectly, to this dissertation and to my long exciting journey of discovery and exploration.

First and foremost, I would like to thank Dr. Timothy C. Y. Chan, without whose guidance, support and patience I could not finish my Ph.D study. I would like to express my sincere gratitude and appreciation to Dr. Christopher J. Beck for his guidance and suggestions throughout my research and study.

I am grateful to other committee members, Dr. Scott J. Rogers, Dr. Chi-Guhn Lee and Dr. Markus Bussmann, for their valuable help and guidance in my dissertation, coursework, and academic pursuit. I would like to thank Dr. Bernard Gendron and Dr. Yuri Lawryshyn for their generous participations on the SGS examination committee and for their reviews of the dissertation and suggestions.

My heartfelt thanks to many other professors in the department of Mechanical and Industrial Engineering (MIE) at the University of Toronto for making my life here so wonderful: Dr. Michael W. Carter, Dr. Chul B. Park, Dr. Daniel M. Frances, Dr. Jean Zu and others. I thank my colleagues in MIE: Stephen Stoyan, Zhong Ma, Judy Geng, John Liu, Zhijian Yin and Sheena Yau who have helped me a lot in my doctoral study. I also thank my friends: Dr. Richard Caron, Dr. Brad Bass and Dr. Carrie Dang. Thank you all for the good times we shared.

Finally, overwhelmingly thanks to my family, especially to my son Luke Wang, for their unconditional love and constant support. I appreciate their understanding and supporting.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Introduction

Uncertainty is an inherent feature in most manufacturing systems that needs to be controlled or analyzed. Many practical problems from industry involve uncertain quantities such as unknown demands, costs, shipping conditions, inventory reservations and production patterns. Decision-making in those problems is concerned with the challenging issue of how to make optimal decisions in uncertain environments. Uncertainty may sometimes be forecasted based on statistical distribution assumptions (see Ahmed and Shapiro (2002)), time-series projections, regression analysis (see Prekopa (2003)), or simulation modeling (see Jonsson and Silver (1996)). However, in many cases, estimating the parameters to describe uncertain quantities is difficult, inaccurate, or even impossible. In such cases, the stochastic nature of uncertainty might be better described by using a set of forecasted discrete scenarios, each with a probability of future occurrence. Stochastic programming (SP) is an Operational Research (OR) method to solve problems with this kind of uncertainty (see Kall and Wallace (1994) and Birge and Louveaux (1997) ).

SP is concerned with studying, analyzing and solving mathematical programming problems where uncertain parameters and data are represented by random variables. The

SP problem, which incorporates random variables as parameters, can model the uncertain real world more closely compared to conventional deterministic methods. Therefore, SPs have drawn increasingly more attention in the applied optimization area (see Birge and Louveaux (1997) ).

In many industrial applications, some data may be known deterministically, or may be estimated based on assumptions, whereas the uncertain future is only characterized by a probability distribution. The decision being made with known data at this point is referred to as the first stage decision. Subsequently, when information about the future is revealed, there will be an opportunity to augment the first stage solution in order to optimize the whole problem for the realized scenario. The second stage is called the *recourse stage.* This type of model is called SP with *recourse* (see Birge and Louveaux (1997), Kall and Wallace (1994)), which is a primary type of SP model. Another type of SP model is called SP with chance constraints (see Prekopa (2003), Nemirovski and Shapiro (2006)), which is a means of handling uncertainty by assuming that stochastic constraints will be only satisfied within a specified confidence level.

In this thesis, we will focus on the recourse model. The goal of a stochastic model with recourse is to optimize the first and the second stage decision variables so as to minimize the expected cost or maximize the expected profit over both stages.

In the past decade, stochastic programs with mixed integer variables and linear recourse, also called Stochastic Mixed Integer Programs (SMIP), have become well-established decision tools in many industrial applications (see Klein and van der Vlerk (1999)). SMIPs are among the most challenging large-scale optimization problems because both stochastic programs and mixed integer programs are generally difficult to solve. The size of the SMIP can easily explode as the number of random parameters and their possible realizations increase.

The research in the thesis is aimed at developing a scenario-wise decomposition method as the framework to decompose two-stage stochastic recourse models into sub-problems, which have the same structure as the original problem. By using tailored methods to solve sub-problems, the proposed methods can solve large-size stochastic recourse problems with first-stage integer variables more efficiently.

## 1.2 Thesis Structures and Contributions

The theme of this thesis is to develop models, algorithms and industrial applications for SMIP problems. The Dual Decomposition method of Carøe and Schultz (1999) is a primary decomposition method that our algorithms in the thesis are based upon. When we implement the Dual Decomposition method in solving the Stochastic Uncapacitated Facility Location Problem (SUFLP), we observe that the algorithm cannot efficiently converge in the process of branch and bound, which motivates us to develop a new algorithm called the Dynamic Dual Decomposition method or $D^3$ method. First, we compare the performance of different representations of non-anticipativity and other techniques to refine the Dual Decomposition method. Then, using the framework of the $D^3$ method, we implement and solve two-stage and three-stage SUFLPs. We also combine the $D^3$ method with the Column Generation (CG) method to solve the Stochastic Set Packing Problem (SSPP). Both SUFLP and SSPP have only binary variables in the first stage. Lastly, we apply a refined Benders Decomposition method to solve scenario subproblems, which are decomposed from Stochastic Multi-plant Facility Location Problems (SMpFLP). The SMpFLPs are more complex problems that have general integer variables in both the first and the second stage.

The SUFLP is a special case of SMpFLP. Obviously, all three types of problems studied in the thesis, SUFLP, SSPP and SMpFLP, are NP-hard problems since the SMpFLP and SUFLP are extended cases with uncertainty from the classical NP-hard Uncapacitated Facility Location Problem (UFLP) and the SSPP is an extended case from the classical NP-hard Set Packing Problem (see Escudero *et al.* (2011)).

The proposed $D^3$ method decomposes two-stage stochastic recourse models with first-stage integer variables into scenario sub-problems by relaxing non-anticipativity constraints on first-stage variables. The decomposed sub-problems for all scenarios will have the same structure of the deterministic model or stochastic model except that only one scenario is represented in each sub-problem. The deterministic model, or the stochastic model with only one scenario can be solved efficiently by using conventional or tailored methods. Therefore, the $D^3$ method provides the framework to decompose a broad range of two-stage stochastic recourse models. Enhancements of the $D^3$ method will accelerate convergence to a global optimal solution of mixed integer variables.

The three main chapters of this dissertation (Chapters 3, 4, and 5) are devoted to studying decomposition methods with several industrial applications.

> In Chapter 3, the main contribution is the development of a Dynamic Dual Decomposition method ($D^3$). We apply this method to the solution of SUFLPs. First, the computation of the Lagrangian bound at each sub-problem is dynamically determined based on the projected value of the bound. Second, additional heuristics are incorporated (e.g., at the root node) to help accelerate the solution process. Third, we consider alternative ways to represent non-anticipativity that improves computational performance. The proposed method is found to be quite effective in solving various models of SUFLPs. We find that the $D^3$ method cannot only decompose and solve SUFLPs, but also provide a very promising framework to facilitate other conventional algorithms in solving more challenging practical industrial problems with uncertainty.

In Chapter 4, the stochastic version of the Set Packing Problem (SSPP) is studied and modeled as a two-stage SMIP with binary variables in both the first stage and the second stage. We combine the $D^3$ method with Column Generation (CG) to solve multiple test problems and examine the sensitivity of different density patterns. We find that our proposed method is more effective and robust in solving the SSPP compared to a state-of-the-art IP solver trying to solve the whole problem directly. The solution times of solving SSPPs directly with the CPLEX solver are very sensitive to the density over the same size problems, but using the proposed method, the solution times vary within a small range in spite of the different densities.

In Chapter 5, we extend our $D^3$ method to general integer first stage variables and mixed integer second stage variables and apply the method to the stochastic multi-plant facility location problem. The general integer (as opposed to binary) variables in both the first and the second stage cause the SMIP to be intractable. We develop new strategies for accelerating Bender's Decomposition and tailor it for scenario subproblems that are decomposed by the $D^3$ method. We also develop an aggregation method on large scale problems to approximate the optimal solution of the original problem.

Next, in Chapter 2 we begin with a literature review of different decomposition methods on SMIPs and follow with some industrial applications modeled as SMIPs.

# Chapter 2

# Background and Literature Review

## 2.1 Stochastic Linear Programs with Integer Variables

The field of stochastic programming is concerned with studying, analyzing and solving mathematical programming problems where uncertain parameters and data are represented by random variables. Stochastic Linear Programming (SLP) with linear constraints and a linear objective function is a suitable framework to incorporate uncertainty in a linear optimization problem. SLP models with recourse represent here-and-now decisions that are made before uncertainty is resolved, but with the consideration of future corrective action (also known as recourse) in the future under linear constraints in various scenarios (see Kall and Wallace (1994), and Birge and Louveaux (1997)).

The dynamic decision process (Carøe and Schultz (1999)) with uncertain data follows the two stage scheme in SP with recourse:

$$\text{Decision in } 1^{st} \text{ stage } \mathbf{x}_1 \mapsto \text{Observation of uncertainty } \omega \mapsto \text{Decision in } 2^{nd} \text{ stage } \mathbf{x}_2(\omega)$$

Here the decision variables are split into two parts, $\mathbf{x}_1$ for the first stage and $\mathbf{x}_2$ for the second stage with the convention that $\mathbf{x}_2$ will depend on uncertainty parameter $\omega$.

The vector $\mathbf{x}_2$ is commonly referred to as *recourse variables* to reflect the idea that the components of $\mathbf{x}_2$ will be chosen to compensate for random events after $\mathbf{x}_1$ has been fixed and $\omega$ has been observed.

The two-stage model can be generalized to dynamic optimization problems extending over a finite, discrete time horizon (also called as multi-stage). The basic scheme has the form

$$\cdots \mapsto \text{Decision in (t-1)}^{th} \text{ stage } \mathbf{x}_{t-1}(\omega_{t-1}) \mapsto \text{Observation of } \omega_t \mapsto \text{Decision in t}^{th} \text{ stage } \mathbf{x}_t(\omega_t) \mapsto \cdots$$

The process extends over time horizon $t = 1, \ldots, T$ stages with $\omega_t$ as a random variable that will be observed at the start of the $t$-th stage and $\mathbf{x}_t$ as a decision that will be made in the $t$-th stage.

We first present a formulation for the two-stage SLP with recourse:

$$z = \min \quad \mathbf{c}'\mathbf{x} + \mathbf{E}_\xi[\min_{\mathbf{y}} \mathbf{q}(\omega)'\mathbf{y}(\omega)] \tag{2.1}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.2}$$

$$\mathbf{T}(\omega)\mathbf{x} + \mathbf{W}(\omega)\mathbf{y}(\omega) = \mathbf{h}(\omega) \quad \forall \omega \in \Omega \tag{2.3}$$

$$\mathbf{x} \geq \mathbf{0}; \quad \mathbf{y}(\omega) \geq \mathbf{0} \quad \forall \omega \in \Omega \tag{2.4}$$

The vector $\mathbf{x}$ is the decision variables for the first stage with size $n_1$ and the vector $\mathbf{y}$ is decision variables for the second stage with size $n_2$. The primes on vectors are used as transpose. Matrices $\mathbf{c}$, $\mathbf{b}$, and $\mathbf{A}$ are of sizes $n_1 \times 1$, $m_1 \times 1$ and $m_1 \times n_1$, respectively. In the second stage, a random event $\omega \in \Omega$ may be realized. For a given realization of $\omega$, the second stage problem parameters $\mathbf{q}(\omega), \mathbf{h}(\omega), \mathbf{T}(\omega)$ and $\mathbf{W}(\omega)$ become known, where $\mathbf{q}(\omega)$ is $n_2 \times 1$, $\mathbf{h}(\omega)$ is $m_2 \times 1$, $\mathbf{T}(\omega)$ is $m_2 \times n_1$ , and $\mathbf{W}(\omega)$ is $m_2 \times n_2$.

For a realization $\omega$, the vector $\xi(\omega) = (\mathbf{q}(\omega), \mathbf{h}(\omega), \mathbf{T}(\omega), \mathbf{W}(\omega))$ represents a scenario. Let $\Xi \subseteq \mathbf{R^n}$ be the support of $\xi$. $\mathbf{E}_\xi$ is the mathematical expectation with respect to $\xi$. For a realization $\omega$, let

$$Q(\mathbf{x}, \xi(\omega)) = \min_{\mathbf{y}} \{\mathbf{q}(\omega)'\mathbf{y} \mid \mathbf{W}(\omega)\mathbf{y} = \mathbf{h}(\omega) - \mathbf{T}(\omega)\mathbf{x}, \mathbf{y} \geq 0\}$$

be the second stage value function, also called in the literature as *recourse function.* The matrices $\mathbf{T}(\omega)$ and $\mathbf{W}(\omega)$ are referred to as the *technology* matrix and the *recourse* matrix, respectively. If the technology matrix is deterministic, that is, $\mathbf{T}(\omega) = \mathbf{T}\ \forall\ \omega \in \Omega$, the SLP (2.1–2.4) is called to have *fixed tenders.* If the recourse matrix is deterministic, that is, $\mathbf{W}(\omega) = \mathbf{W}\ \forall\ \omega \in \Omega$, the SLP (2.1–2.4) is said to have *fixed recourse.* When the second stage problem is feasible for all $\mathbf{x} \in \mathbf{R}^{n_1}$, the SLP (2.1–2.4) is considered as possessing the *complete* recourse property. When the second stage problem is feasible for all $\mathbf{x} \in \mathbf{R}^{n_1} \bigcap \{\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, the SLP (2.1–2.4) is considered as possessing the *relatively complete* recourse property.

The scenarios represent different possibilities for the uncertain parameters. It is assumed that each scenario occurs with a known probability. Typically, the distribution of $\omega$ is multivariate. To avoid complications when computing the integral of $\mathbf{E}_\xi$, we assume that there are only a finite number of scenarios where the $s$-th scenario occurs with probability $p^s$. We use $S$ to represent the finite set of scenarios and $|S|$ to denote the total number of scenarios, respectively.

In the past decade, stochastic programs with mixed integer variables and linear recourse, also called Stochastic Mixed Integer Programs (SMIP), have become well-established decision tools in many industrial applications. SMIPs are among the most challenging large-scale optimization problems because both stochastic programs and mixed integer programs are generally difficult to solve. The size of the SMIP can easily explode as the number of random parameters and their possible realizations increase. In this thesis, we confine our study to SMIP, therefore variables are mixed with integer and continuous variables and all constraints and objective are linear functions. The scope of these models is sufficiently wide to fit many industrial applications, since many types of non-linear problems can be handled by using integer variables and linearizing mathematical formulations. (see Klein and van der Vlerk (1999) and Schultz (2003) for a discussion of the challenges in solving general SMIP).

In order to solve large-scale SMIP problems efficiently, various decomposition methods have been developed. This thesis is concerned with the study of models, algorithms and industrial applications of SMIP problems. The main decomposition methods will be reviewed in this chapter.

The formulation of a SLP (2.1–2.4) can be specialized to the Deterministic Equivalent Problem (DEP), which is the most common way to represent scenario-based stochastic programs with a finite number of scenarios, as SLP or SMIP. The DEP will be a large-scale, block structured LP, if all variables of $\mathbf{x}$ and $\mathbf{y}$ are continuous, or Mixed Integer Problem (MIP), if some variables of $\mathbf{x}$ and $\mathbf{y}$ are integer.

$$z = \min \quad \mathbf{c}'\mathbf{x} + \sum_{s \in S} p_s \mathbf{q}_s' \mathbf{y}_s \tag{2.5}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.6}$$

$$\mathbf{T}_s\mathbf{x} + \mathbf{W}_s\mathbf{y}_s = \mathbf{h}_s \qquad \forall s \in S \tag{2.7}$$

The above DEP is a Mixed Integer Program (MIP), if integer variables are applicable, and can be solved by a commercial MIP solver directly, which may apply Branch and Bound (B&B) method, Branch and Cut (B&C) method, or other conventional methods to MIP.

The recourse function in SMIP (2.5–2.7) inherits the properties of MIP if only the first stage variables include integer restrictions. Otherwise, if the integer restrictions appear in the second stage, the SMIP is much more challenging in terms of complexity. The objective function in SMIP (2.5–2.7) is generally discontinuous and nonconvex in the first stage variables, $\mathbf{x}$, for all scenarios $s \in S$, and the expected recourse function is discontinuous and nonconvex even under the assumption of complete recourse and a weak covariance condition on the technology matrix and the recourse matrix (Schultz (2003) and Ntaimo (2004)).

The integrality of the first stage or second stage decision variables in SMIP (2.5–2.7) increases complexity of the problem. Integer programming is generally NP-hard and therefore SMIP (2.5–2.7) inherits the NP-hard property and computational difficulty. The evaluation of each recourse function is a NP-hard problem if there are integer restrictions on the second stage variables since the convexity properties of the continuous recourse functions no longer hold.

The basic idea behind decomposition methods is to decompose a large-scale SMIP into several subproblems, each of which corresponds to a scenario or a stage, and then to solve the smaller sized subproblems separately in the decomposition-coordination setting. Therefore, decomposition approaches can be divided into two categories based on how the two-stage SMIP is decomposed. *Vertical directive* decomposition methods decompose the SMIP scenario-wise into similar subproblems for each individual scenario. On the other hand, *horizontal directive* decomposition methods decompose a large-scale SMIP stage-wise into a master problem for the first stage and a number of subproblems for the second stage.

## 2.2 Heuristic Decomposition Methods for Stochastic Linear Problem with Recourse

The progressive hedging (PH) method for solving multi-stage SLP is a scenario-wise decomposition technique proposed in Rockafellar and Wets (1991), which is the foundation of a number of heuristic methods for SMIP with recourse.

In order to model uncertainty, Rockafellar and Wets (1991) proposed the scenario bundling structure in the multi-stage scenario tree. The scenarios in a bundle are observably indistinguishable at the start time before scenarios happened. Then the multi-stage SLP can be rewritten as a scenario formulation and all decision variables and constraints are replicated for each scenario. The replications of the decision variables of scenarios in a bundle have to guarantee the decisions made are identical before the time stage of the bundle by adding a set of non-anticipativity constraints. The original SLP are decomposed into scenario subproblems with all original constraints and non-anticipativity constraints. A solution is called implementable if it satisfies the non-anticipativity constraints but may not satisfy the original constraints in SLP. A solution is called admissible if it satisfies the original constraints but may not satisfy the non-anticipativity constraints. A solution is feasible to the original SLP if it is both implementable and admissible. The hedging decomposition method iteratively solve problems for an implementable solution and an admissible solution. The non-anticipativity constraints are relaxed by the augmented Lagrangian function which incorporates multiplier and quadratic penalty terms in the objective function. The convergence theorem in Rockafellar and Wets (1991) claims that the PH algorithm can converge in linear time to the unique global optimal solution in the convex case of SLP. The theoretical studies also show if the algorithm converges in the non-convex case and the subproblems are solved to a local optimal solution, then it converges to a local optimal solution of the original SLP.

The PH method is combined with a tabu search algorithm as a heuristic method for solving multi-stage stochastic mixed binary problems by Lokketangen and Woodruff (1996). Computational experiments verified that the method is effective on solving benchmark instances.

However, integer constraints on decision variables make stochastic programming problems with recourse non-convex and cause considerable difficulty to optimal solutions. The non-convergence and unacceptable long time running are major issues in applying PH-based heuristic method on solving large-scale SMIP. The PH-based algorithmic enhancements are proposed in Watson and Woodruff (2010) that enable convergence and solve the discrete cases in tractable times. They develop critical techniques for computing effective values of heuristic approaches, accelerating convergence, adding better termination criteria, and detecting cycling behavior in the PH-based heuristic algorithm on solving SMIPs.

In Crainic *et al.* (2011), a two-stage stochastic fixed-charge capacitated multi-commodity network design problem with uncertain demand is solved by a meta-heuristic framework which is inspired by the PH method.

A dual decomposition method, which is also a scenario-wise decomposition algorithm, was proposed in Mulvey and Ruszczynski (1995) to solve large, multistage SLP with recourse. The authors exploited the tree structure of the non-anticipativity constraints and treat non-anticipativity in a compact way. They were the first to develop a dual decomposition method for multi-stage SLP by applying augmented Lagrangian approach to separate the copies of the first stage variables into scenario. The quadratic terms in the augmented Lagrangian function were handled by the diagonal quadratic approximation in order to completely decompose the large, multistage SLP into scenario sub-problems. The sub-problems were modified in separable quadratic terms in order to coordinate the scenario solutions and were solved by a nonlinear interior point algorithm in parallel computer systems. The computational tests showed the proposed algorithm could greatly reduce the large sparse matrix computation and achieve 14 to 20 times speedup on solving large SLPs with recourse.

An exact method in Ahmed (2013) can decompose two-stage stochastic programming with binary variables scenario-wise and obtain an optimal solution by iteratively exploring and cutting-off feasible solutions solved from scenario subproblems. The computational results using a distributed implementation show the proposed method can solve standard two-stage SIP and nonlinear SIP test problems efficiently.

In the thesis, we will also apply Lagrangian dual to relax equalities which represent the non-anticipativity constraints and decompose large scale SMIP scenario-wise.

## 2.3 Scenario-wise Decomposition Methods

The DEP (2.5–2.7) can be easily decomposed and solved if the first stage variables are not restricted to be identical over all scenarios. However the solutions of the first stage are required to be identical over all scenarios in the SLP.

The Dual Decomposition method in Carøe and Schultz (1999) decomposes a single SMIP into scenario subproblems by relaxing the non-anticipativity constraints. The decomposed subproblem of each scenario has the same structure of the corresponding deterministic problem or one-scenario problem. The only difference lies in the coefficients of the first stage variables which include Lagrangian dual parameters.

The idea of scenario-wise decomposition in the Dual Decomposition method is to introduce copies $\mathbf{x}_1, \cdots, \mathbf{x}_{|S|}$ of the first stage variable $\mathbf{x}$ and then rewrite the DEP (2.1–2.4) in the form with the constraint to restrict all copies of the first stage variable to be identical:

$$z = \min \quad \sum_{s \in S} p_s \left( \mathbf{c}' \mathbf{x}_s + \mathbf{q}_s' \mathbf{y}_s \right) \tag{2.8}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x}_s = \mathbf{b} \qquad \forall s \in S \tag{2.9}$$

$$\mathbf{T}_s \mathbf{x}_s + \mathbf{W}_s \mathbf{y}_s = \mathbf{h}_s \qquad \forall s \in S \tag{2.10}$$

$$\mathbf{x}_1 = \cdots = \mathbf{x}_{|S|} \tag{2.11}$$

Condition (2.11) is called non-anticipativity, which ensures that all decisions in the first stage must not be influenced by the scenarios that will occur in the next stage. One of the major computational challenges in SLP (2.8–2.11) is enforcing non-anticipativity constraints.

Carøe and Schultz (1999) presented a relaxation of the non-anticipativity conditions and solved the relaxed problem using sub-gradient methods. Their method is called Dual Decomposition in SLP. First, the non-anticipativity is represented by the equality

$\sum_{s=1}^{|S|} \mathbf{H}_s \mathbf{x}_s = 0$. The Lagrangian relaxation with respect to the non-anticipativity condition will completely decompose the SLP into scenario subproblems which is defined as

$$\min \left\{ \sum_{s \in S} L_s(x_s, y_s, \lambda) \mid \mathbf{A}\mathbf{x}_s = \mathbf{b}, \mathbf{T}_s\mathbf{x}_s + \mathbf{W}_s\mathbf{y}_s = \mathbf{h}_s, \mathbf{y}_s \in \mathbf{Y}, \mathbf{x}_s \in \mathbf{X}, \forall s \in S \right\} \tag{2.12}$$

where $\mathbf{X}$ and $\mathbf{Y}$ are corresponding feasible regions of the original SLP, $\lambda$ is the Lagrangian multiplier with dimension $n_1$ and

$$L_s(x_s, y_s, \lambda) = p_s(\mathbf{c}'\mathbf{x}_s + \mathbf{q}_s'\mathbf{y}_s) + \lambda \mathbf{H}_s \mathbf{x}_s \text{ for } s = 1, ..., |S|.$$

The Lagrangian dual of (2.12) becomes the problem

$$z_{LD} = \max_{\lambda} \left\{ \sum_{s \in S} L_s(x_s, y_s, \lambda) \right\}. \tag{2.13}$$

The optimal solution of the Lagrangian dual is a relaxed bound of the original problem SLP (2.5–2.7) due to the weak duality of integer programming.

**Proposition 1.** *(Proposition 1 in Carøe and Schultz (1999)) The optimal value of the Lagrangian dual of (2.13) is a lower bound on the optimal value of (2.5–2.7). If for some choice $\lambda$ of Lagrangian multipliers the corresponding solution $(x_s, y_s), s = 1, ..., |S|$ of the Lagrangian relaxation (2.12) is feasible, then $(x_s, y_s), s = 1, ..., |S|$ is an optimal solution of problem (2.8–2.11) and $\lambda$ is an optimal solution of (2.13).*

When we refine the Dual Decomposition method and propose the Dynamic Dual Decomposition method ($D^3$ method) in Chapter 3, we use subgradient methods to solve the Lagrangian dual (2.13) which is a convex non-smooth program. An advantage of the Lagrangian relaxation is that it splits the problem into separate subproblems for each scenario, while maintaining structure of the original SLP:

$$\begin{aligned} z_s = \ \min \quad & \widehat{\mathbf{c}}'\mathbf{x}_s + \mathbf{q}_s'\mathbf{y}_s && (2.14) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x}_s = \mathbf{b} && (2.15) \\ & \mathbf{T}_s\mathbf{x}_s + \mathbf{W}_s\mathbf{y}_s = \mathbf{h}_s && (2.16) \end{aligned}$$

where

$$\widehat{\mathbf{c}}' = (\mathbf{c}' + \frac{1}{p_s}\lambda \mathbf{H}_s) \tag{2.17}$$

We may then use the same method in solving the non-stochastic problems to solve the decomposed scenario subproblems (2.14–2.16).

When the Dual Decomposition method was introduced in Carøe and Schultz (1999), it was assumed, for notational convenience, that all first stage decisions are required to be binary, that is, $\mathbf{X} = \{0,1\}^{n_1}$. Then they expressed non-anticipativity by the single constraint: $\mathbf{x}_1 = p_1\mathbf{x}_1 + ... + p_{|S|}\mathbf{x}_{|S|}$ where $p_1, ..., p_{|S|}$ are probabilities of corresponding scenarios. When we implement the Dual Decomposition method, the computational results show this equality with probability coefficients provides a loose lower bound. Consequently, the B&B process does not converge efficiently. We developed a new matrix representation of non-anticipativity equality. The chart (see Figure 1.1 and 1.2 in Chapter 3) of the tracks of bounds in the B&B process shows the new equality leads to a quickly convergent solution.

## 2.4 Stage-wise Decomposition Methods

### 2.4.1 L-shaped Method

In order to solve large-scale SMIPs efficiently, some stage-wise decomposition methods have been well studied. Stage-wise decomposition scales well with the number of scenarios in the two-stage case.

A well-known decomposition method for solving large MIPs is Benders Decomposition Method (BDM), which was first described in Benders (1962). The conventional BDM decomposes the MIP into two parts. One part is called the Benders master problem, which only contains integer variables. The other part is called the Benders sub-problem, which only contains continuous variables. The BDM iteratively solves each part by fixing one set of partitioned variables and solving for the other part. The algorithm solves the dual problem of the Benders sub-problem, also called the Benders dual problem, with the fixed feasible solution of integer variables. Then, the BDM adds a cut on the feasible region of integer variables and solves the Benders master problem with fixed continuous variables obtained in the solution of the previous iteration.

The "L-shaped method" was first proposed in Van Slyke and Wets (1969) for two-stage stochastic recourse problems with linear constraints and continuous variables. It is similar to the BDM by decomposing the two-stage stochastic problems into the Benders master problem with first-stage variables and the Benders sub-problem with second-stage variables. The algorithm can solve stochastic problems, which even fail to satisfy the complete recourse assumptions. The limitations of the "L-shaped method" have been studied and improved by Sherali and Fraticelli (2002).

The main idea of using conventional BDM to solve the DEP (2.5–2.7) is to separate the two-stage SMIP into subproblems with second stage variables and the master problem with first stage variables and an artificial variable. In this case, the decomposed subproblem for each scenario with given fixed first stage variables $\bar{\mathbf{x}}$ is

$$\min_{\mathbf{y}_s} \ \beta_s(\bar{\mathbf{x}}) = \mathbf{q_s}'\mathbf{y}_s \tag{2.18}$$

$$\text{s.t.} \ \mathbf{W}_s\mathbf{y}_s = \mathbf{h}_s - \mathbf{T}_s\bar{\mathbf{x}} \tag{2.19}$$

The subproblem contains only the second stage variables $\mathbf{y}_s$ and it can be decomposed and solved independently for each scenario. The associated dual problem for each scenario is

$$\max_{\mathbf{v}_s} \ \beta_s(\bar{\mathbf{x}}) = \mathbf{v}_s'(\mathbf{h}_s - \mathbf{T}_s\bar{\mathbf{x}}) \tag{2.20}$$

$$\text{s.t.} \ \mathbf{v}_s'\mathbf{W}_s = \mathbf{q_s} \tag{2.21}$$

After the dual problems are solved instead of the subproblem, the Benders decomposition main problem with given fixed second stage dual variables $\bar{\mathbf{v}}_s$ is

$$\min_{\mathbf{x}} \ \alpha(\bar{\mathbf{y}}) = \mathbf{c}'\mathbf{x} + \vartheta \tag{2.22}$$

$$\text{s.t. } \vartheta \geq \sum_{s \in S} p_s \bar{\mathbf{v}}_s (\mathbf{h}_s - \mathbf{T}_s \mathbf{x}) \tag{2.23}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.24}$$

The above gives a general idea of how the conventional BDM is applied to decompose a two-stage SMIP. There are some derivatives developed such as replacing the second stage problems with a convexification process or adding optimal cuts to accelerate the convergence (see Birge and Louveaux (1997)). Since the optimality cut (2.23) is generally weak when the BDM is used to decompose SMIPs with some binary variables in the first stage, an additional optimality cut for the piecewise linear approximation of the expected recourse functions is derived in the "Integer L-shaped method" by Laporte and Louveaux (1993). The "Integer L-shaped method" was developed to decompose a class of stochastic recourse problems with binary variables in the first and second stage. The two-stage SMIP with complete recourse is decomposed and solved by the proposed algorithm, which implements a conventional Branch-and-Cut procedure to handle the first-stage binary variables. The improved optimality cuts guarantee the algorithm to obtain an optimal solution, if it exists, in a finite number of iterations.

Let $L$ denote the lower bound on the expected value of the recourse functions. In the $k$-th iteration of BDM with the feasible solution of the first stage, $\mathbf{x}^k$, let $f^k = \mathbf{E}[Q^k(\mathbf{x}^k, \xi(\omega))]$ be the corresponding expected value of the second stage recourse functions. Define two sets of indices of $\mathbf{x}^k$ as $S^k = \{i \in n_x | x_i = 1\}$ and $\bar{S}^k = \{i \in n_x | x_i = 0\}$. The optimality cut is defined as

$$\vartheta \geq (f^k - L)(\sum_{i \in S^k} x_i - \sum_{i \in \bar{S}^k} x_i) - (f^k - L)(|S^k| - 1) + L$$

where $|S^k|$ is the cardinality of $S^k$. The value calculated in $\sum_{i \in S^k} x_i - \sum_{i \in \bar{S}^k} x_i$ is always no more than $|S^k|$ for any feasible solution. Hence, the cut is sharp at $\mathbf{x}^k$ and the constraint is alway satisfied at other binary solutions of the first stage variable. When more information is available on $\mathbf{E}[Q^k(\mathbf{x}^k, \xi(\omega))]$, the optimality cuts could be improved in the "Integer L-shaped method" (Laporte and Louveaux (1993)). Birge and Louveaux (1997) show how to improve the optimality cut when more information is available on the recourse function. The master problem and the scenario subproblems are all MIPs and can be solved directly by the commercial MIP solver. Since the scenario subproblems with mixed binary variables and continuous variables are solved at each iteration over each scenario in the "Integer L-shaped method," it is observed in Ntaimo (2004) that the performance of "Integer L-shaped method" on solving large-scale SMIPs is poor.

In Chapter 5, we apply Benders Decomposition method to the decomposed multi-plant facility location problem. However, instead of using BDM to decompose the two-stage SMIP into a master problem with first stage variables and the dual subproblems with second stage variables, we use the proposed $D^3$ method to decompose the whole SMIP into scenario subproblems, then apply BDM to further decompose the subproblems with mixed integer variables. The master problem in BDM contains all integer variables from both the first stage and the second stage. The dual subproblems of BDM contain only continuous variables from the second stage.

### 2.4.2 Disjunctive Decomposition Method

The Disjunctive Decomposition ($D^2$) method, developed by Sen and Higle (2005), is an extension to the "Integer L-shaped method." The $D^2$ method develops cutting-plane-based approximations on the feasible set of the second stage problem. The core of the $D^2$ method is the common cut coefficient theorem, which provides a mechanism for transforming cuts derived for one outcome of the second stage problem into cuts that are valid for another outcome.

For a class of SMIPs with the assumptions:

1) binary first stage,

2) mixed-binary second stage,

3) fixed recourse matrix $W$,

4) relative complete recourses,

the *common-cut-coefficient* ($C^3$) theorem and the *Disjunctive Decomposition* ($D^2$) method are defined by Sen and Higle (2005) to solve this class of problems. Developed on the "Integer L-shaped method," the $D^2$ method can further add cuts in the second stage, which are called $D^2$ cuts. The cut for one scenario subproblem can be easily translated into a cut that is valid for another scenario subproblem by taking advantage of the assumptions.

Based on the assumptions, the SMIP (2.5–2.7) is decomposed into two parts, namely. The first stage part is:

$$z = \min \quad \mathbf{c}'\mathbf{x} + \sum_{s \in S} p_s f(\mathbf{x}, s) \tag{2.25}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.26}$$

$$\mathbf{x} \in \mathbf{B}^{n_x} \tag{2.27}$$

The second stage part is

$$f(\mathbf{x}, s) = \min \quad \mathbf{q}_u'\mathbf{u}_s + \mathbf{q}_z'\mathbf{z}_s \tag{2.28}$$

$$\text{s.t.} \quad \mathbf{W_u}\mathbf{u}_s + \mathbf{W_z}\mathbf{z}_s = \mathbf{h}_s - \mathbf{T}_s\mathbf{x}, \qquad \forall s \in S \tag{2.29}$$

$$\mathbf{u}_s \in \mathbf{R}_+^{n_u}, \mathbf{z}_s \in \mathbf{B}^{n_z}, \qquad \forall s \in S \tag{2.30}$$

where $\mathbf{B}^{n_x}$ and $\mathbf{B}^{n_z}$ are the set of binary vectors with the corresponding dimensions of variables of $\mathbf{x}$ and $\mathbf{z}$, respectively.

In the $D^2$ method, let $\mathbf{x^k}$ denote the first stage solution associated with the $k^{th}$ algorithm iteration, whereas the subproblems (2.28–2.30) are solved with the linear relaxation on the second stage binary variables $\mathbf{z}$. If the optimal solution of the relaxed subproblem in some scenarios has a non-integer basis for some variables in $\mathbf{z}$, the disjunction restrictions will be added like branching in the B&B tree. For example, let $j(k)$ denote an index for which $z_j^k(s)$ is non-integer for some scenario $s \in S$, and let $\bar{z}_j^k$ denote the incumbent non-integer value. To eliminate the non-integer solution, the disjunction restrictions

$$-z_j^k \geq -\lfloor \bar{z}_j^k \rfloor \text{ and } z_j^k \geq \lceil \bar{z}_j^k \rceil$$

will be added to create two subproblems. When the integer restrictions are binary, the right hand sides of two disjunction restrictions are set to zero and one, respectively. Then, the disjunction used in lift-and-project cuts of Balas *et al.* (1993) can be precisely used to add the common cuts for all subproblems.

However, the common cutting planes generated within the $D^2$ algorithm are typically inadequate for solving large SMIPs. A Branch-and-Cut (BAC) algorithm is incorporated into the $D^2$ method, called $D^2$-BAC (Sen and Sherali (2006)), to enhance the common cuts in scenario subproblems.

The airline fleet assignment problem in Zhu (2006) contains mixed binary variables in both the first stage and the second stage, and the value functions in the second stage are in general non-convex and discontinuous. Therefore, the $D^2$ method cannot be directly applied to solve the two-stage stochastic airline fleet assignment problems, since the method requires the first stage subproblems to have convex value functions. To make the algorithm more efficient, a certain partial convex hull is constructed using the relaxed second stage constraints and the restrictions enforcing the first stage variables to lie in a hyper-rectangle. With the development of the $D^2$ method, two-stage SMIPs with 0-1 mixed-integer variables in both stages and practical applications to airline fleet assignment problems can be solved efficiently.

Although we name the proposed method in the thesis as "$D^3$" method, which stands for "Dynamic Dual Decomposition" method (see details in Chapter 3), there is no connection with the aforementioned "$D^2$" method, which stands for "Disjunctive Decomposition" method. As we have discussed before, these two methods are developed following different decomposition streams: the $D^2$ method is developed as a stage-wise decomposition method, whereas the $D^3$ method is developed as a scenario-wise decomposition method.

### 2.4.3 Tender Method

If there are some continuous variables in the first stage, conventional IP algorithms such as B&B, B&C or Column Generation, applied to the continuous variables may not reach a finite termination. A finite B&B algorithm for a class of SMIP called the *Tender Method* is developed by Ahmed *et al.* (2004), based on global optimization. The Tender Method is another stage-wise decomposition method on specific two-stage SMIP with the following assumptions:

1) the first stage variables are all continuous,

2) the second stage variables are all integer,

3) the SMIP has fixed tenders ($\mathbf{T}(\omega) = \mathbf{T}, \forall\, \omega \in \Omega,$), and

4) the recourse matrix has all integer items.

Although the technology matrix is fixed, the recourse matrix is allowed to be random over all scenarios. The main discovery by Ahmed *et al.* (2004) is that the value function in the second stage with pure integer variables $\mathbf{y}(\omega)$ and integer recourse matrix $\mathbf{W}(\omega)$ is constant over hyper-rectangles. Additionally, if the first stage variables $\mathbf{x}$ has a bounded feasible region as $\mathbf{X} = \{\mathbf{x}|\mathbf{A}\mathbf{x} = b, \mathbf{x} \geq 0\}$, then there are only a finite number of such hyper-rectangles. In order to obtain global optimal solutions in a finite number of steps, first the problem is transformed into a space of "*tender variables*" $\chi = Tx$ :

$$z = \min \quad \mathbf{\Phi}(\chi) + \sum_{s \in S} p_s \mathbf{\Psi}_s(\chi) \tag{2.31}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.32}$$

$$\mathbf{T}\mathbf{x} = \chi \tag{2.33}$$

$$\mathbf{x} \geq 0 \tag{2.34}$$

The second stage part is as follows:

$$\mathbf{\Psi}_s = \min \quad q_s\mathbf{y}_s \tag{2.35}$$

$$\mathbf{W}_s\mathbf{y}_s \geq h_s - \chi \tag{2.36}$$

$$\mathbf{y}_s \geq 0 \quad \forall\, \omega \in \Omega \tag{2.37}$$

Then, they proposed to search the space of tender variables for the global optima instead of the space of first stage continuous variables. One may notice that the feasible region of $\chi$ is a linear transformation of $\mathbf{X}$ with a fixed matrix $T$.

In some generated test problems with integer first stage variables and binary second stage variables, the Tender method performs better than the Dual Decomposition method. Also a collection of practical product substitution problems are modeled as the two-stage SMIP and solved by the proposed method.

All variations of two-stage recourse SMIPs with a finite number of scenarios (2.1–2.4) can be formulated as MIP version of the DEP in (2.5–2.7). Therefore, conventional direct methods to solve MIPs can be applied directly to solve DEP, i.e., B&B, Benders Decomposition, Column Generation methods, etc. However, conventional methods that are not tailored for the stochastic version of large-scale MIPs with recourse are not of interest to the thesis. The focus of the thesis is on technical decomposition approaches that exhibit potential for decomposing large-scale SMIPs and consequently solving them more efficiently.

## 2.5 Multi-stage Stochastic Programs with Recourse

The decomposition methods on two-stage SMIPs can be theoretically extended to multi-stage problems; however, scalability becomes a major issue. Therefore, some specific multi-stage methods have been developed from several perspectives. Here, we give a brief summary of multi-stage SMIPs. For more detailed reviews on multi-stage SMIPs, we refer the reader to Ravi and Sinha (2006), Higle *et al.* (2004) and Huang (2005).

The nested Benders Decomposition Method developed by Birge (1985) is extended in the same way on the two-stage approach as to approximate the value function by a piecewise linear approximation in each stage. Multistage stochastic linear programming is also studied and solved by the PH method in Rockafellar and Wets (1991).

The PH method combined with a tabu search is developed to solve multi-stage SMIP with mixed binary variables in Lokketangen and Woodruff (1996). The tabu search algorithm is applied to obtain solutions in the induced quadratic binary MIP subproblems.

A scenario decomposition method for multi-stage convex SP in Higle *et al.* (2004) is a cutting plane algorithm with an appropriate column aggregation method, which is created based on previous statistical tests. However, these methods are just for solving Stochastic Linear Problems with only continuous variables but not for mixed integer variables.

A branch-and-fix coordination approach was derived by Alonso-Ayuso *et al.* (2003) for solving a class of multi-stage SMIP, where all decision variables are binary. The method coordinates the selection of the branching nodes and branching variables in the scenario subproblems to be jointly optimized.

As was presented in Section 2.1, let $t = 1, \dots, T$ denote the index of multi-stage horizon over a finite sequential decision process and assume that the information is revealed by a discrete time stochastic process $\{w_t\}_{t=1}^{T}$. The decision at time $t$ is based on the set of decisions and the outcomes in the previous stages and the revealed information on the current stage. Let $\underline{\mathbf{x}}_t = (x_1, ..., x_t)$ denote the vector of decision variables from stage 1 to stage $t$ and $\underline{\mathbf{w}}_t = (w_1, ..., w_t)$ denote the corresponding vector of the random outcomes from stage 1 to stage $t$. Then a multi-stage SMIP can be described as follows:

$$\text{Min } \{\mathbf{c'}_1(\mathbf{w}_1)\mathbf{x}_1 + Q_1(\mathbf{x}_1)\} \tag{2.38}$$

$$\text{s.t. } \mathbf{A}_1\mathbf{x}_1 \le \mathbf{h}_1(\mathbf{w}_1),\ \mathbf{x}_1 \in \mathbf{X_1} \tag{2.39}$$

where for $t = 1, ..., T-1$,

$$Q_t(\underline{\mathbf{x}}_t) = \mathbf{E}_{\widetilde{\mathbf{w}}_{t+1}|\underline{\mathbf{w}}_t}\{\text{Min } \mathbf{c}'_{t+1}(\widetilde{\mathbf{w}}_{t+1})\mathbf{x}_{t+1} + Q_{t+1}(\underline{\mathbf{x}}_{t+1})\} \tag{2.40}$$

$$\text{s.t.} \quad \mathbf{T}_{t+1}(\widetilde{\mathbf{w}}_{t+1})\underline{\mathbf{x}}_t + \mathbf{A}_{t+1}\mathbf{x}_{t+1} \le \mathbf{h}_{t+1}(\widetilde{\mathbf{w}}_{t+1}),\ \mathbf{x}_{t+1} \in \mathbf{X_{t+1}} \tag{2.41}$$

and $Q_T = 0$. Let $\mathbf{E}_{\widetilde{\mathbf{w}}_{t+1}|\underline{\mathbf{w}}_t}$ denote the expectation with the distribution of $\widetilde{\mathbf{w}}_{t+1}$ on the observation of $\underline{\mathbf{w}}_t$. So $\mathbf{w}_1$ is known at time $t = 1$. Assume that $\mathbf{T}_t(\widetilde{\mathbf{w}}_t), \mathbf{A}_t, \mathbf{c}_t(\widetilde{\mathbf{w}}_t)$ and $\mathbf{h}_t(\widetilde{\mathbf{w}}_t)$ are corresponding matrices and vectors with appropriate dimensions. The set $\mathbf{X_t}$ denotes restrictions on variables $\mathbf{x}_t$ if integrality or other restrictions are required.

Motivated by applications in semiconductor planning, a general multi-stage stochastic capacity planning model involving discrete capacity acquisition decisions is formulated in Huang (2005).

$$\text{Min} \quad \sum_{t=1}^{T}[\alpha_t\mathbf{x}_t + \beta_t\mathbf{y}_t] \tag{2.42}$$

$$\text{s.t.} \quad \mathbf{A}_t\mathbf{y}_t \le \sum_{\tau=1}^{t}\mathbf{x}_\tau, \quad \forall t \tag{2.43}$$

$$\mathbf{B}_t\mathbf{y}_t \ge \delta_t, \quad \forall t \tag{2.44}$$

$$\mathbf{y}_t \in \mathbf{R}_+^{\mathbf{J}}, \mathbf{x}_t \in \mathbf{Z}_+^{\mathbf{I}}, \quad \forall t \ . \tag{2.45}$$

In the model (2.42–2.45), the vector of decision variables $\mathbf{x}_t$ denotes the capacity acquisition decisions for a set of $\mathbf{I}$ resources, and the vector of decision variables $\mathbf{y}_t$ represents the operational level allocation of capacity to a set of $\mathbf{J}$ tasks in period $t$. The vectors $\alpha_t$, $\beta_t$ and $\delta_t$ represent acquisition costs, allocation costs, and demands, respectively. The matrices $\mathbf{A}_t$ and $\mathbf{B}_t$ indicate resource-task utilization coefficients. The multi-stage stochastic version of generic capacity acquisition and allocation models are very hard to solve directly by any commercial MIP solvers.

By studying a special lot-sizing structure inherent in the model, they developed an efficient approximation method for the multi-stage SMIP and proved that the proposed method is able to obtain the optimal solution asymptotically.

We applied the proposed $D^3$ method to decompose the three-stage Stochastic Uncapacitated Facility Location Problem in Chapter 3, as a demonstration of the ability of the $D^3$ method to solve problems beyond two stages. However, the focus of the thesis is mainly on decomposition methods for two-stage SMIPs. Developing specific methods to solve multi-stage SMIPs which are based on the $D^3$ method will be promising future work.

Although our algorithms are implemented on a serial processor, they can also be used as the basis for future implementation on a parallel or distributed computing platform or in a cloud computing system, which will be promising future work as well.

## 2.6 Some Industrial Applications with Uncertainty

Although many applications from industry inherit uncertainty and integrality, most SMIP models only started to be developed and solved in the last decade due to the lack of efficient algorithms to tackle such problems (see Klein and van der Vlerk (1999)).

Traditional location models typically assume that the demand distribution and constraints are fully specified, and the cost structure is fairly simple. However, real-world situations often involve many other considerations (see Birge and Louveaux (1997)).

Manufacturing usually involves complicated operations and scheduling problems, when uncertainties cannot be ignored. In addition, the demand and supply aspects of manufacturing are often characterized by randomness.

The facility location problem is a well-known and fundamental problem raised from logistics and supply chain management in manufacturing. The problem is to locate facilities among potential facility sites and to assign demand to opened facilities so as to minimize total cost (i.e., location and transportation costs). The problems with facilities assumed to be unlimited or limited in capacity are called the uncapacitated or capacitated facility location problems, respectively (see Klose and Drexl (2005), and Owens and Daskin (1998)).

### 2.6.1 Stochastic Uncapacitated Facility Location Problems

Uncertain demand is common to most real-life supply chain cases and can sometimes be forecasted with a certain degree of accuracy using statistical distributions. However, in some cases, estimating the parameters to describe uncertain demand is difficult, or inaccurate, or impossible. Therefore, the stochastic nature of demand and other various aspects have been incorporated in a broad class of Uncapacitated Facility Location Problems (UFLPs) (Sinha (2004)). The term "uncapacitated" refers to the problems for which any number of clients can be assigned to an open facility. The UFLP, which is an NP-hard problem, is to select locations to set up facilities with unlimited capacity and to allocate products and services to customers in given locations. The stochastic version of the UFLP (SUFLP) involves uncertain issues where the demands are unknown when facility locations are decided but additional facilities can be set with higher costs after the demands are revealed.

The deterministic UFLP has been studied extensively in the operational research community, and more recently in the computer science community as well (see Owens and Daskin (1998), Sinha (2004)).

An SUFLP that includes uncertainty in demand, selling prices, as well as in production and transportation costs is modeled as a two-stage SMIP by Louveaux and Peeters (1992). In the model, they simply split the location decisions in the first stage and the demand allocation in the second stage. In the SUFLPs in Chapter 3, the location decisions are modeled in both the first stage and the second stage to augment production and minimize serving costs on revealed demands.

The SUFLP with mixed integer variables is formulated in Sinha (2004) with a constant-factor approximation algorithm by rounding the linear programming relaxation of the formulation. The main insight of their work is that the optimal solution to the linear relaxation can be surprisingly useful in guiding the decisions about the facility setting in the present and future stages. The approximation algorithm is based on following the fractional solution in deciding which variables are rounded to integers in each scenario.

The two-stage SUFLP is described below, and will be extensively studied and solved in Chapter 3. Let

$N =$ the set of potential facility locations, $N = \{i \mid i = 1, 2, ...|N|\}$;

$D =$ the set of clients, $D = \{j \mid j = 1, 2, ...|D|\}$;

$S =$ the set of scenarios, $S = \{s \mid s = 1, 2, ...|S|\}$;

$f_i =$ the setup costs for locating a facility at location $i \in N$ in the first stage;

$h_i^s =$ the setup costs for locating a facility at location $i \in N$ in scenario $s \in S$ in the second stage;

$c_{i,j}^s =$ the cost per unit of serving client $j \in D$ from facility $i \in N$ in scenario $s \in S$;

$d_j^s =$ the demand from client $j \in D$ in scenario $s \in S$;

$p^s =$ the probability of scenario $s \in S$;

$M =$ a large number much larger than all the problem data;

$u_i = 1$ if a facility will be opened at location $i$ in the first stage, 0 otherwise;

$x_i^s = 1$ if a facility will be opened at location $i$ in scenario $s$, 0 otherwise;

$y_{i,j}^s =$ the number of products served from facility $i$ to client $j$ in scenario $s$.

$$\min \quad \sum_{i\in N} f_i u_i + \sum_{s\in S} p^s [\sum_{i\in N} h_i^s x_i^s + \sum_{i\in N, j\in D} c_{i,j}^s y_{i,j}^s] \tag{2.46}$$

$$\text{s.t.} \quad \sum_{i\in N} y_{i,j}^s \geq d_j^s \qquad \forall j \in D, \forall s \in S \tag{2.47}$$

$$y_{i,j}^s \leq M(u_i + x_i^s) \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{2.48}$$

$$u_i, x_i^s \text{ binary} \qquad \forall i \in N, \forall s \in S \tag{2.49}$$

$$y_{i,j}^s \geq 0 \text{ integer} \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{2.50}$$

In the two-stage SUFLP model, there are a total number of $|D||S| + |N||D||S|$ constraints and $|N| + |N||S| + |N||D||S|$ variables consisting of $|N| + |N||S|$ binary and $|N||D||S|$ integer variables. The objective function (2.46) minimizes the total costs including the setup costs of opening facilities in the first stage and the expected value under all scenarios of the recourse costs, which involve the setup costs of opening facilities and costs of servicing clients in the second stage. Constraints (2.47) enforce that the demands will be satisfied for each client under each scenario in the second stage. Constraints (2.48) ensure that a client will only be served from facility $i$, if it is opened either in the first or second stage. Constraints (2.49) ensure all location decisions are binary, and constraints (2.50) ensure that a non-negative integer amount of the products is shipped from a facility to a client in each scenario. The model considered in Chapter 3 is similar to Sinha (2004), except that the serving variables in the second stage are non-negative integer values, not binary. In Sinha (2004), for simplicity, they only studied the case when all $y_{i,j}^s$ variables are either 0 or 1. We also consider $y_{i,j}^s$ as non-negative integer and continuous variables in competition tests. It is worthwhile to notice that the recourse parameters in our model exist not only in the recourse constraints, but in the objective function as well. More scenario parameters with uncertainty will cause large derivation on the solutions of scenario subproblems, hence more computational effort will be taken to hedge the identical solution of the first stage variables over all scenarios.

### 2.6.2 Stochastic Multi-plant Facility Location Problems

The capacitated facility location problem is also a classical combinatorial optimization problem. It amounts to building some facilities with limited capacity in a given list of locations and allocating customer demands to them, in such a way that the sum of the fixed costs associated with opening the facilities and the total cost of supplying customers is minimized (see Ghiani *et al.* (2002a)).

The Multi-plant Facility Location Problems (MpFLP) is one kind of capacitated facility location problem where several plants can be built in each location. Therefore, the facility location decision variables are general integer (as opposed to binary) variables. The aforementioned SUFLP can be extended to the Stochastic Multi-plant Facility Location Problems (SMpFLP) (Ghiani *et al.* (2002b)). The SMpFLP is also strongly NP-hard and is related to a large variety of industrial applications including distribution system design, lot sizing in production planning and telecommunication network design.

An early attempt for capacitated facility location problem to represent random demand in single product plant of each location can be found in Balachandran and Jain (1976). The objective function for the facility cost is piecewise, nonconvex linear although the transportation costs are linear. The capacity of each facility is chosen in the first stage then the uncertain demands are allocated to facilities in the second stage with the deviations from the capacities of facilities penalized with costs for over-capacity and under-capacity. They employ the conventional B&B method to solve the problem. The branching is processed on capacity intervals where the piecewise linear objective function has breakpoint. The objective function is bounded by the best linear function.

The capacitated facility location problem with both long-run and short-run costs and uncertain customer demands is formulated as a two-stage SMIP in Schütz *et al.* (2008). They apply a Lagrangian relaxation method to solve a practical slaughter house location case from the Norwegian meat industry. They analyze the facility location problem with non-linear facility costs and stochasticity in serving costs and customer demands. First, they model the capacities of facilities as the first stage decision variables in a designated interval rather than a fixed value. Then, they provide a thorough description and motivation of the short-run cost function of facilities which leads to a general piecewise linear convex cost function in the second stage. A hybrid approach is proposed to combine capacitated and uncapacitated facilities. Stochastic set-up costs and stochastic demands are allowed in the model. They choose the Lagrangian relaxation method to decompose the whole problem and solve the large-size problems originating from real life cases. The algorithm makes the problem separable in scenarios by imposing non-anticipativity constraints on the first stage variables. Then, the subproblem is further decomposed for each facility by relaxing the demand constraints, then the subproblem is solved based on a method for a continuous knapsack problem.

The Sole-sourcing Stochastic Facility Location problem (SSFLP) is studied in Silva (2004), where each customer's demand will be completely satisfied by a sole facility. In the model, the demands, capacities and costs in the future are not known by a manufacturer. For simplicity, they assume the aggregate demand for a facility may exceed the facility's production capacity and the facility will pay a penalty based on the unsupplied amount. The SSFLP has a similar formulation as SUFLP except for a group of binary variables to decide if a customer is assigned and served to a sole facility. They rewrite the model to fit into the format of a column-oriented two-stage stochastic program which may theoretically require much less B&B enumeration to solve than a standard SSFLP .

The Branch-and-Price (B&P) method has been applied to SSFLP by stabilizing dual variables during column generation, performing strong branching, and inserting multiple near-optimal columns from each subproblem (see Silva (2004)). It uses Column Generation to decompose the whole MIP and uses reduced costs to identify appropriate assignments in sub-problems. Their approach, which is similar to the "$D^3$+Column Generation" method in Chapter 4, uses Column Generation to solve a pricing subproblem for each scenario and leave non-anticipativity constraints in the master problem. However, the full set of feasible assignments may not be easy to generate for general stochastic facility location problems.

In Fazel-Zarandi and Beck (2012), a new stochastic facility location/fleet management problem is modeled as an IP which is a sole-sourcing facility location and vehicle assignment problem with each customer's demand being served by one full return shipment. With the uncertain travel-time conditions and bounded penalties in the model, the whole model is hard to solve directly by the IP solver. Two-level and three-level logic-based Benders decomposition methods are developed to solve the model for global optimal solutions. Computational experiments and comparisons illustrate that the proposed logic-based Benders decomposition methods outperform the state-of-the-art commercial IP solver.

The polling stations problem in an Italian municipality is modeled as the capacitated plant location problem with multiple facilities in the same site in Ghiani *et al.* (2002a). In the model, $U$, the set of potential facilities, is partitioned into $n$ subsets of potential facilities, each of which is associated to a site. Then $U_k$ is the set of potential facilities that can be located in site $k$ ($k = 1, ..., n$); hence, $U = U_1 \bigcup U_2 \bigcup ...U_n$. Therefore, the decision variables on facility locations are still binary for each location although identical facilities may be set up in one site.

The polling stations problems can be modeled as the SMpFLP that is introduced in Chapter 5. Instead of binary variables for deciding each facility location, general integer variables of each location will be assigned, and the upper bound of the integer variable at location $k$ will be the cardinality of set $U_k$. The tailored Lagrangian heuristic method in Ghiani *et al.* (2002a) may not provide high quality lower and upper bounds on the SMpFLP with general integer variables.

The practical problem of high speed telecommunication network design is modeled as a two-stage stochastic problem with discrete variables in the first stage in Andrade *et al.* (2004). The problem of designing backbone telecommunication networks under uncertainty consists of installing sufficient capacities of network links to meet future demands. A telecommunication company would pay high penalty costs in the event of unmet demands to its customers. The resulting problem is formulated as a two-stage SMIP problem. The investment decision variables are determined in the first stage. The expected penalty costs referred to the decision variables are considered in the second stage.

Two formulations of the problem are created. The first one is with general integer investment variables that is considered as indivisible capacities. The second one is a variant of the first model, with 0-1 investment variables that are considered as modular capacities with fixed charge. In Andrade *et al.* (2005), they investigate three solution approaches to solve the models: the integer L-shaped method, a tailored B&B method and a disjunctive cutting method. The integer L-shaped decomposition is presented as a variant of the standard approach reported in Laporte and Louveaux (1993). The B&B algorithm is to solve the linear problem relaxation of the model to optimality using Benders decomposition before the partitioning process. They use the scenario solutions to obtain optimality cuts that can be added to the relaxed Benders master problem. The disjunctive cutting method is an extension of lift-and-project cuts that are a direct approach for general MIPs. The computational results using three algorithms for solving the stochastic version of telecommunication network problems shows that the B&B algorithm outperforms the disjunctive cutting plane method for both formulations and the B&B algorithm can work better than Benders Decomposition. However, there are only integer variables in the first stage and only continuous variables in the second stage in their formulations.

### 2.6.3 Stochastic Set Packing Problem

The Set Packing Problem (SPP) is fundamental to tackle numerous practical issues that have been well studied over the last 40 years (see Garey and Johnson (1979)). Only a few attempts to study and solve the Stochastic version of Set Packing Problem (SSPP) have been found in the last couple of years, e.g., Escudero *et al.* (2011), Dean *et al.* (2008) and Dean *et al.* (2005).

The formulation of the conventional SPP is as follows:

$$\max \quad \sum_{i \in N} c_i x_i \tag{2.51}$$

$$\text{s.t.} \quad \sum_{i \in N} a_{k,i} x_i \leq 1 \qquad \forall k \in K \tag{2.52}$$

$$x_i \quad \text{all binary} \qquad \forall i \in N, \tag{2.53}$$

where $N$ and $K$ are the set of available patterns and the set of elements in each pattern, respectively. In the objective function, $c_i$ is the reward of picking the $i$-th pattern. The binary coefficients in the constraints, $a_{k,i}$ indicates if the $k$-th element is packed in the $i$-th pattern. The goal of SPP is to obtain the maximal benefit by picking the patterns to pack all available elements.

The SPP belongs to the class of NP-hard combinatorial optimization problems (see Garey and Johnson (1979)). Although approximation techniques and heuristics can provide quick solutions for SPP, the uncertainty often originates from the difficulty of choosing only one deterministic situation where the problems occur. In practice, it is often the case that some patterns are not known precisely until in later terms. Beforehand, it is known only in terms of a probability distribution. This motivates us to consider a stochastic variant of the set packing problem where items are deterministic but patterns in the second stage are independent random variables with known, completely arbitrary distributions.

An approximate method is proposed in Escudero *et al.* (2011) for solving a specified SSPP, which is also called a simple recourse problem. The model is different from the two-stage SSPP with recourse model which is described in Chapter 4. They design a one-stage simple recourse SSPP with a mean-risk function as the objective function. Their model is described below:

$$\max \quad \sum_{s\in S}\sum_{i\in N} p^s c_i^s x_i - \beta \sum_{s\in S} p^s \delta^s \tag{2.54}$$

$$\text{s.t.} \quad \sum_{i\in N} a_{k,i} x_i \le 1 \qquad \forall k \in K \tag{2.55}$$

$$\sum_{i\in N} c_i^s x_i \ge \phi - \phi\delta^s \qquad \forall s \in S \tag{2.56}$$

$$x_i, \delta^s \quad \text{binary} \qquad \forall i \in N, \forall s \in S \tag{2.57}$$

In the above model, the benefits $c_i^s$ of picking pattern $i$ is associated with scenario $s$. The fixed penalty $\beta$ is associated with the scenarios whose objective value is smaller than a given threshold $\phi$. The objective function involves maximizing the expected value of benefits over all scenarios minus the penalty $\beta$ with probabilities of scenarios, where benefits in these scenarios are smaller than a given threshold $\phi$. Note that $\delta^s$ is defined as 1 if the benefit of scenario $s$ less than a threshold, or 0 otherwise. So the only stochastic variable is a one-dimensional binary variable for each scenario.

By exploiting the structure of the model and presenting the splitting variable representation in Escudero *et al.* (2011), they obtain a lower bound on the optimal value, then use the volume algorithm to update the multipliers in the Lagrangian decomposition and consequently develop a tight upper bound for better feasible solutions.

We describe a different model of the two-stage SSPP with recourse in Chapter 4, which involves choosing determined patterns then selecting uncertain patterns over scenarios. Uncertainties happen not only in the presence of elements in each pattern, but also in the rewards of patterns and the total number of available patterns over scenarios in the recourse stage. Therefore the model is more complex than the aforementioned simple recourse SSPP (2.54 - 2.57).

The aforementioned industrial applications of stochastic facility location problems and stochastic set packing problems will be modeled and studied in this thesis. The following applications are for introductory purposes and for integrating the broad scope of the stochastic applications of readers' interests.

### 2.6.4 Stochastic Scheduling Problems

The Stochastic Scheduling Problem (SSP) was also studied and modeled in the mathematical programming formulation in Dean (2005). Practical task problems can be modeled as SSPs, where uncertainty lies in the processing of each job and the goal is to compute a schedule whose expected value is close to one of an optimal adaptive schedule. In their models, the processing time of each job is random, known in advance only as a probability distribution. SSPs are also at least NP-hard since they are variants of the knapsack problem. Some heuristic algorithms to solve SSPs in polynomial time have been developed.

The framework for planning and scheduling problems with uncertainty is coined in Bidot *et al.* (2009). The stochastic model provides a generic representation and it can integrate three complementary techniques to deal with uncertainty.

Since the notation and terminology of scheduling problems are much different than the facility location problems, we will not review the regime of scheduling problems for the sake of succinctness of the thesis. Interested readers may refer to Dean (2005) and Bidot *et al.* (2009) for more on SSP.

Due to the increasing popularity of Constraint Programming (CP) in the last couple of decades, complex scheduling problems or Job-Shop Problems (JSP) have been modeled in the format of CP and solved much faster than as MIP.

Large-scale SSPs can be modeled as the combination of MIP and CP, then be decomposed by our $D^3$ method. The main idea is that we model the SSP as a two-stage stochastic programming with recourse. The first stage variables and constraints are modeled in the mathematical formulation of MIP and the second stage parts are modeled in the formulation of CP. Then we will use the $D^3$ method to decompose the SSP and invoke both CP and MIP solvers to solve the decomposed subproblems. However, due to the limits of time period, hardware resources, and software capacity, the uncompleted study have to leave in the future work which will be discussed in the future research section in Chapter 6.

More study on the variants of SSPP, Stochastic Knapsack Problem (SKP) is extended in Dean *et al.* (2008), where the NP-hard 0/1 knapsack problem is modeled to pack a maximum-value collection of items with stochastic vector-valued sizes.

## 2.7 List of model types and applications

As we have reviewed the literature in terms of decomposition methodology, there are mainly two branches of decomposition method on two-stage SMIPs with recourse: stage-wise decomposition and scenario-wise decomposition. In the following table, we have included some important decomposition methods that pertain to the discussions in this thesis and accompanying practical applications by classifying the variable types in both stages. The list is not exhaustive covering all SMIP models with recourse and the papers in the literature column are not the only examples or citations of each class. In the table, we will use the following notations to indicate the variable types.

$B$: indicates Binary variables.

$I$: indicates Integer variables.

$C$: indicates Continuous variables.

$C_b$: indicates Continuous variables scaled onto the range $[0, 1]$.

| First stage | Second stage | Application | Algorithm | Literature |
|---|---|---|---|---|
| $B$ | $C$ | Telecommunication Network | L-shaped method and other methods | Andrade *et al.* (2004) |
| | | Polling station problems | Lagrangian Heuristic | Ghiani *et al.* (2002b) |
| $B$ | $B+C$ | Stochastic server location problem | L-shaped method<br>$D^2$ method | Ntaimo and Sen (2008) |
| | | SUFLP | $D^3$ | Chapter 3 |
| $B+C_b$ | $B+C_b$ | Airline Fleet Assignment | Advanced L-shaped method | Zhu (2006) |
| $B$ | $I$ | Hydro-Thermal Power System | Dual Decomposition | Carøe and Schultz (1999) |
| $C$ | $I$ | Product substitution problem | Tender method | Ahmed *et al.* (2004) |
| $B$ | $B$ | Stochastic Facility Location/Fleet Management | Logic-based BDM | Fazel-Zarandi *et al.* (2013) |
| | | SSPP | $D^3$+CG | Chapter 4 |
| $I$ | $I+C$ | SMpFLP | $D^3$+PBDM | Chapter 5 |

Table 2.1: List of some two-stage SMIP models with recourse

## 2.8 The generality of the proposed methods

As it has been reviewed in Section 2.6, many applications from industry inheriting uncertainty and integrality can be modeled as a two-stage stochastic recourse model with integer first-stage variables. The whole model is still a MIP in terms of mathematical formulation. However, the mixed integrality of the whole model is generally difficult to solve. Also the size of the whole model can easily explode as the number of possible realizations and uncertain parameters increase.

The proposed $D^3$ method provides a framework to decompose two-stage stochastic recourse models with integer first-stage variables. The first-stage variables in the two-stage stochastic recourse models should be decided before the future occurrence, and hence be identical over all scenarios in the second-stage. One may re-write the model by assigning the copies of the first-stage variables to each scenario and add the non-anticipativity conditions to constrain the copies of the first-stage variables to be identical in the final optimum. The $D^3$ method can relax the non-anticipativity condition and decompose the whole model into sub-problems for each scenario. Unlike stage-wise decomposition methods, the proposed $D^3$ method, which is a scenario-wise decomposition method, can decompose the whole model into sub-problems for each scenario. The scenario sub-problems inherit the same structure of the original deterministic model or the original stochastic model with one scenario except that only the coefficients of the first-stage variables changed. In this way, well-developed algorithms to solve the deterministic models of original problems can be combined with the $D^3$ method to solve scenario sub-problems efficiently.

The Lagrangian approach of the $D^3$ method can tighten the relaxed bounds of the stochastic problem and a heuristic method can calculate feasible solutions of the first-stage variables. The B&B approach in the $D^3$ method will search the feasible regions of first-stage variables and obtain a global optimal solution, if it exists, in a finite number of iterations.

If there exists continuous variables in the first-stage of two-stage SLPs with recourse, the $D^3$ method, which is proposed as a direct method in the thesis, cannot guarantee optimality in a finite number of iterations. We have to use additional approaches to handle continuous variables in the first-stage for a global optimum. So continuous first-stage variables may limit the direct implementation of the $D^3$ method. Our study in the thesis focuses on a direct method to solve linear stochastic recourse problems with integer first-stage variables for a global optimum.

The structure and the feature of the second-stage has no impact on the decomposition performance in the $D^3$ method, whether or not there are continuous or integer, or mixed with continuous and integer variables, and whether or not there are non-linear constraints in the second-stage. All these challenging situations will be addressed with tailored algorithms or commercial solvers which are invoked to solve scenario sub-problems after the $D^3$ method decomposed the whole model into scenario sub-problems.

Since the $D^3$ method is independent of the structure of subproblems and second-stage parts, an appropriate method or ad-hoc algorithm is required to solve the subproblems after implementing the $D^3$ method. Therefore the processing of the $D^3$ method has no assistance in advance on solving subproblems, which is a weakness of the $D^3$ method as a scenario-wise decomposition framework.

In the thesis, we use the $D^3$ method to solve various problems where the first-stage variables are either binary (in Chapter 3 and Chapter 4) or general integer (in Chapter 5) and the second-stage variables are all binary (in Chapter 4) or all general integer (in Chapter 3) or mixed with integer and continuous variables (in Chapter 3 and Chapter 5). However, all constraints in stochastic recourse problems we studied are linear. As we mentioned, scenario sub-problems decomposed by the $D^3$ method can be solved by efficient methods which are tailored for the deterministic problems. In Chapter 3, we combine the

$D^3$ method with the MIP solver to solve the scenario sub-problems directly. In Chapter 4, we include the refined Column Generation method to solve the sub-problems which are set packing problems. In Chapter 5, we use the developed multi-tier Pareto-Benders decomposition methods to solve the sub-problems which are multi-plant facility location problems.

Unlike some stage-wise decomposition methods, no further assumptions on the stochastic recourse models, like fixed recourse matrix or technology matrix, complete recourse or relatively complete recourse, fixed right-hand-side, etc., are required to directly implement the proposed $D^3$ method. However, we may add some assumptions for implementing well-developed algorithms on solving scenario sub-problems efficiently. So the $D^3$ method can decompose two-stage stochastic recourse models with any random parameters and variables in the second-stage, which include broader stochastic models from practical applications.

In order to demonstrate the efficiency of the proposed methods, we compare the computational times (CPU times) and global optimal solutions of the developed methods and the state-of-art commercial solver, ILOG CPLEX MIP solvers. All the tests were initially programmed by invoking the CPLEX 9.0 solver since 2005. Since September 2012, we have re-run all tests with the latest version, the CPLEX 12.4 solver. The new empirical results still support our theoretical study on the proposed methods, although the CPLEX 12.4 MIP solver are much more significantly improved on solving large MIPs than the older CPLEX 9.0 MIP solver.

# Chapter 3

# Dynamic Dual Decomposition ($D^3$) Method on Stochastic Uncapacitated Facility Location Problem (SUFLP)

## 3.1 Introduction to SUFLP Problem

Facility location decisions often need to be made before demands from clients are known. Usually, estimates of demands are made and then decisions are based on these estimated parameters. Ideally, location decisions should be robust to uncertainties so that adjustments made after uncertainty resolutions are minimal. In general, it is impossible to find a single solution that would be optimal for all possibilities of demands.

We consider stochastic versions of the Uncapacitated Facility Location Problem (UFLP) where the demands are uncertain and additional location decisions can be made in subsequent time periods. This version of the UFLP will be referred to as the Stochastic Uncapacitated Facility Location Problem (SUFLP). Incorporating uncertainty in location problems is the next frontier of facility location research and is an important consideration since the factors in location analysis (e.g., cost and demand) are uncertain in reality (see Sinha (2004), Ravi and Sinha (2006), and Snyder (2005)). In some industrial applications, manufacturers will not fully know the demand when they have to decide the plant locations and even have produced and shipped out the products. Although quite often forecasts of future uncertainty are available that can be used in the facility planning models, forecasts, by nature, are imprecise and provide at best a range of possible futures. Some facility location decisions are costly and difficult to reverse, and their impact spans a long time horizon. During the time when design decisions are in effect, some parameters, such as costs, demands and distances, may fluctuate widely. For a more recent application of stochastic location models see Schütz *et al.* (2008) where both economies and dis-economies of scale are considered as well as uncertainty in client demand.

We model the SUFLP as two and three (multi) stage stochastic linear integer programs with mixed integer recourse. In the SUFLP, we employ a finite number of scenarios to represent uncertain serving costs, additional setup costs and client demands. In addition, our formulations for SUFLP including recourse decisions will involve additional location decisions in the future. We also consider three-stage models with the third stage also involving integer recourse of locating and serving behaviours.

Although there has been some consideration of stochastic programming formulations for the UFLP, there has been almost no work on exact approaches for this class of problems. Birge and Louveaux (1997) consider two and multi-stage stochastic programming formulations of UFLP, but no methods are suggested to solve these stochastic programs. Sinha (2004) and Ravi and Sinha (2006) consider approximation algorithms for stochastic programming versions of UFLP. These methods are based on various primal-dual techniques that exploit the linear programming relaxations of the stochastic location problems and in general, are not guaranteed to find an optimal solution. The main aim in these approaches is to construct worst-case bounds. There has been algorithmic work in solving two-stage stochastic programming versions of the $p$-median capacitated facility location problem (see Ntaimo and Sen (2008)). There, the authors describe a branch and cut approach to approximate good solutions to two-stage stochastic integer programs.

The Lagrangian dual is a convex non-smooth program. It can be solved by sub-gradient methods (see Fisher (1981)) or bundle methods (see Kiwiel (1990)). We use the sub-gradient method in our experiments, because the sub-gradient method performs well in the $D^3$ method for solving SUFLP and is easily implemented without involving nonlinear programming problems.

The two-stage stochastic MIP with mixed binary and continuous variables in both the first stage and the second stage is described in Escudero *et al.* (2013). The cluster partitioning method is proposed to decompose the whole model into clusters. A cluster is a group of scenarios and the model in each cluster is still the same as the original two-stage stochastic problem. The non-anticipativity constraints for clusters are rewritten as circular inequalities in order to avoid the use of non-signed vectors of Lagrangian multipliers. Then they use the Lagrangian dual to relax inequalities of cluster non-anticipativity. Four methods to iteratively calculate and update Lagrangian multipliers are tested. The goal of experimenting with four methods on calculating Lagrangian multipliers is to find better relaxed lower bound in considerably shorter CPU times. One of the four methods, the subgradient method, is implemented in the $D^3$ method which is described in Section 3.2.3.

The general strategy in our approach for solving SMIP is based on the dual decomposition method and the adapted branch and bound framework. Central to our approach is the use of heuristics to generate lower and upper bounds at the root node, matrix (and non-matrix) representations of non-anticipativity, and the selective and dynamic computation of Lagrangian dual bounds at sub-problems in the B&B tree. All of our constructs attempt to mitigate the fact that the dual decomposition approach may require excessive time to compute optimal solutions for SMIP.

Our objective is to construct an exact algorithm to solve the SUFLP that will apply to both the two- and multi-stage formulations. Many of the state-of-the-art exact algorithms for stochastic integer programming that could be relevant for SUFLP apply only to two-stage problems (see for example Ahmed *et al.* (2004), Ahmed and Shapiro (2002), Engell *et al.* (2004), Sen and Sherali (2006), Santoso *et al.* (2005), and Sen and Higle (2005)). A promising multi-stage exact approach called "Branch and Fix Coordination" was proposed by Alonso-Ayuso *et al.* (2003), but only considers pure binary decisions at each stage.

### 3.1.1 The Formulation of SUFLP

We have presented the following two-stage SUFLP in Section 2.6.1, which will be studied and solved in the section: let

$N$ = the set of potential facility locations, $N = \{i \mid i = 1, 2, ...|N|\}$;

$D$ = the set of clients, $D = \{j \mid j = 1, 2, ...|D|\}$;

$S$ = the set of scenarios, $S = \{s \mid s = 1, 2, ...|S|\}$;

$f_i$ = the setup costs for locating a facility at location $i \in N$ in the first stage;

$h_i^s$ = the setup costs for locating a facility at location $i \in N$ in scenario $s \in S$ in the second-stage;

$c_{i,j}^s$ = the cost per unit of serving client $j \in D$ from facility $i \in N$ in scenario $s \in S$;

$d_j^s$ = the demand from client $j \in D$ in scenario $s \in S$;

$p^s$ = the probability of scenario $s \in S$;

$M$ = a large number much larger than all the problem data;

$u_i$ = 1 if a facility will be opened at location $i$ in the first stage, 0 otherwise;

$x_i^s$ = 1 if a facility will be opened at location $i$ in scenario $s$, 0 otherwise;

$y_{i,j}^s$ = the number of products served from facility $i$ to client $j$ in scenario $s$.

$$\min \quad \sum_{i\in N} f_i u_i + \sum_{s\in S} p^s[\sum_{i\in N} h_i^s x_i^s + \sum_{i\in N, j\in D} c_{i,j}^s y_{i,j}^s] \tag{3.1}$$

$$\text{s.t.} \quad \sum_{i\in N} y_{i,j}^s \geq d_j^s \qquad \forall j \in D, \forall s \in S \tag{3.2}$$

$$y_{i,j}^s \leq M(u_i + x_i^s) \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{3.3}$$

$$u_i, x_i^s \text{ binary} \qquad \forall i \in N, \forall s \in S \tag{3.4}$$

$$y_{i,j}^s \geq 0 \text{ integer} \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{3.5}$$

The objective function (3.1) minimizes the total costs including the setup costs of opening facilities in the first stage and the expected value under all scenarios of the recourse costs, which involve the setup costs of opening facilities and costs of servicing clients in the second stage. Constraints (3.2) enforce that the demands will be satisfied for each client under each scenario in the second stage. Constraints (3.3) ensure that a client will only be served from facility $i$, if it is opened either in the first or second stage. Constraints (3.4) ensure all location decisions are binary, and constraints (3.5) ensure that a non-negative integer amount of the products is shipped from a facility to a client in each scenario.

Although the model is for uncapacitated facility location problem, a big ”M” in Constraints (3.3) is defined as a large number much larger than all the problem data. But the big number “M” should be avoided in the computational experiment, because Constraints (3.3) will yield poor linear programming relaxation bounds. Actually the big “M” can be replaced by $d_j^s$ as in the following constraints, without altering the uncapacitated facility nature:

$$y_{i,j}^s \leq d_j^s(u_i + x_i^s) \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{3.6}$$

In the formulation of SUFLP (3.1–3.5), we may remove the integer restriction (3.5) on the serving variables $y$. We refer to this modified model as the semi-equivalent MIP of the original SUFLP model. If the demand parameters $d_j^s$ are all integer, the optimal solutions of semi-equivalent MIP are identical to the original MIP of the SUFLP model. The integer character of serving variables $y$ will not be affected. Indeed, for each fixed scenario $s$ and client $j$, serving variables in objective function (3.1) and constraint (3.2) form a set covering problem as follows:

$$\begin{aligned} \min \quad & ... + \sum_{i \in N} c_i y_i \\ \text{s.t.} \quad & \sum_{i \in N} y_i \geq d \end{aligned}$$

Therefore, with integer values of demands $d$, the optimal solution of $y$ will be integer. One may notice that the semi-equivalent property only exists in our simple SUFLP and may not work if other constraints are added, such as capacity constraints or non-continuous shipping costs.

The model of two-stage SUFLP presented here is simple, yet general. Since we only use the examples of this model as a test bed to compare the computational results of proposed methods, the general structure will not affect the computation when solving the decomposed scenario subproblems. Accordingly, we will not implement a specific heuristic or exact algorithm to solve the scenario subproblem, but invoke the CPLEX solver to solve the MIPs directly.

We also consider a three-stage SUFLP model defined as follows: let

$S$ = the set of scenarios in the second-stage, $S = \{s \mid s = 1, 2, ..., |S|\}$;

$R_s$ = the set of scenarios in the third stage following scenario $s$ in the second-stage, $R_s = \{\xi_s \mid \xi_s = 1, 2, ..., |R_s|\}$;

$g_i^{\xi_s}$ = the setup costs for locating a facility at location $i \in N$ in scenario $\xi_s \in R_s$ in the third stage;

$m_{i,j}^{\xi_s}$ = the cost per unit of serving client $j \in D$ from facility $i \in N$ in scenario $\xi_s$ in the third stage;

$e_j^{\xi_s}$ = the demand from client $j \in D$ in scenario $\xi_s$ in the third stage;

$q^{\xi_s}$ = the probability of scenario $\xi_s$ in the third stage, given scenario $s$ happening; $\sum_{\xi_s \in R_s} q^{\xi_s} = 1$ for each scenario $s \in S$.

$z_i^{\xi_s}$ = 1 if a facility will be opened at location $i$ in scenario $\xi_s$ in the third stage, 0 otherwise;

$w_{i,j}^{\xi_s}$ = the number of products served from facility $i$ to client $j$ in scenario $\xi_s$ in the third stage.

All the other variables and parameters are defined as in the two stage SUFLP.

$$\min \sum_{i \in N} f_i u_i + \sum_{s \in S} p^s \left[ \sum_{i \in N} h_i^s x_i^s + \sum_{i \in N, j \in D} c_{i,j}^s y_{i,j}^s + \sum_{\xi_s \in R_s} q^{\xi_s} \left( \sum_{i \in N} g_i^{\xi_s} z_i^{\xi_s} + \sum_{i \in N, j \in D} m_{i,j}^{\xi_s} w_{i,j}^{\xi_s} \right) \right] \tag{3.7}$$

$$\text{s.t.} \sum_{i \in N} y_{i,j}^s \geq d_j^s \qquad \forall j \in D, \forall s \in S \tag{3.8}$$

$$y_{i,j}^s \leq d_j^s (u_i + x_i^s) \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{3.9}$$

$$\sum_{i \in N} w_{i,j}^{\xi_s} \geq e_j^{\xi_s} \qquad \forall j \in D, \forall s \in S, \forall \xi_s \in R_s \tag{3.10}$$

$$w_{i,j}^{\xi_s} \leq e_j^{\xi_s} (u_i + x_i^s + z_i^{\xi_s}) \qquad \forall i \in N, \forall j \in D, \forall s \in S, \forall \xi_s \in R_s \tag{3.11}$$

$$u_i \text{ binary} \qquad \forall i \in N \tag{3.12}$$

$$x_i^s \text{ binary}, y_{i,j}^s \geq 0 \text{ integer} \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{3.13}$$

$$z_i^{\xi_s} \text{ binary}, w_{i,j}^{\xi_s} \geq 0 \text{ integer} \qquad \forall i \in N, \forall j \in D, \forall s \in S, \forall \xi_s \in R_s \tag{3.14}$$

The objective function (3.7) minimizes the cost of opening facilities in the first stage and the expected costs of opening facilities and meeting client demands in the second-stage and the third stages. Constraints (3.8) and (3.10) enforce that the demands will be satisfied for each client in the second-stage and the third stage, respectively. Constraints (3.9) and (3.11) ensure that clients are served only from the facilities that are opened in the previous stages or the current stage. Constraints (3.12)-(3.14) ensure that location decisions are binary and service variables are non-negative integer values for all stages and scenarios. As we have discussed before, the big "M" can be replaced by the corresponding demands in Constraints (3.9) and (3.11).

By adding non-anticipativity constraints on the second-stage variables, we may rewrite the three-stage SUFLP as a two stage SUFLP as follows:

$T$ = the set of all scenarios, $T = \{\xi |\ \xi = 1, 2, ...|T|\} = \bigcup_{s\in S} R_s$ and $|T| = \sum_{s\in S} |R_s|$;

$g_i^\xi$ = the setup costs for locating a facility at location $i \in N$ in scenario $\xi \in T$;

$m_{i,j}^\xi$ = the cost per unit of serving client $j \in D$ from facility $i \in N$ in scenario $\xi \in T$;

$e_j^\xi$ = the demand from client $j \in D$ in scenario $\xi \in T$;

$z_i^\xi = 1$ if a facility will be opened at location $i$ in scenario $\xi$, 0 otherwise;

$y_{i,j}^\xi$ = the number of products served from facility $i$ to client $j$ in scenario $\xi$.

$$\min \sum_{i\in N} f_i u_i + \sum_{\xi\in T} p^\xi [\sum_{i\in N} h_i^\xi x_i^\xi + \sum_{i\in N, j\in D} c_{i,j}^\xi y_{i,j}^\xi + \sum_{i\in N} g_i^\xi z_i^\xi + \sum_{i\in N, j\in D} m_{i,j}^\xi w_{i,j}^\xi] \tag{3.15}$$

$$\text{s.t.} \sum_{i\in N} y_{i,j}^\xi \geq d_j^\xi \qquad \forall j \in D, \forall \xi \in T \tag{3.16}$$

$$y_{i,j}^\xi \leq d_j^\xi (u_i + x_i^\xi) \qquad \forall i \in N, \forall j \in D, \forall \xi \in T \tag{3.17}$$

$$\sum_{i\in N} w_{i,j}^\xi \geq e_j^\xi \qquad \forall j \in D, \forall \xi \in T \tag{3.18}$$

$$w_{i,j}^\xi \leq e_j^\xi (u_i + x_i^\xi + z_i^\xi) \qquad \forall i \in N, \forall j \in D, \forall \xi \in T \tag{3.19}$$

$$x_i^{1_s} = \cdots = x_i^{R_s} \qquad \forall i \in N, \forall s \in S \tag{3.20}$$

$$y_{i,j}^{1_s} = \cdots = y_{i,j}^{R_s} \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{3.21}$$

$$u_i \text{ binary} \qquad \forall i \in N \tag{3.22}$$

$$x_i^\xi \text{ binary}, y_{i,j}^\xi \geq 0 \text{ integer} \qquad \forall i \in N, \forall j \in D, \forall \xi \in T \tag{3.23}$$

$$z_i^\xi \text{ binary}, w_{i,j}^\xi \geq 0 \text{ integer} \qquad \forall i \in N, \forall j \in D, \forall \xi \in T \tag{3.24}$$

The objective function (3.15) minimizes the cost of opening facilities in the first stage and the expected costs of opening facilities and meeting client demands in later stages. Constraints (3.16) and (3.18) enforce that the demands will be satisfied for each client in each stage for all scenarios. Constraints (3.17) and (3.19) ensure that clients are served only from the facilities that are opened in the previous stages or the current stage. Constraints (3.20) and (3.21) ensure the non-anticipativity of the recourse decisions. Constraints (3.22)-(3.24) ensure that location decisions are binary and service variables are non-negative integer values for all stages and scenarios.

## 3.2 A Solution Methodology for the SUFLP

We consider a Lagrangian dual decomposition-based branch and bound approach, called the Dual Decomposition method, which was developed by Carøe and Schultz (1999). This method considers the deterministic equivalent problem of the two- and multi-stage SUFLP, which is a mixed integer program (MIP), and then relaxes the non-anticipativity constraints. The Lagrangian dual is computed through standard sub-gradient methods and may be used at each sub-problem of the branch and bound tree as a lower bound instead of the linear programming relaxation of the deterministic equivalent MIP. A major advantage of this approach is that the Lagrangian dual is decomposable in terms of independent scenario sub-problems. However, there are some significant challenges in this approach where, for large problems, it may be too expensive to compute the Lagrangian dual at sub-problems in the branch and bound tree.

### 3.2.1 Matrix Representation of Non-anticipativity

The different forms of the non-anticipativity constraints in SLP have been exploited in Mulvey and Ruszczynski (1995). A general format and a compact one to represent non-anticipativity are proposed and studied in SMIP in Carøe and Schultz (1999). It motivate us to quantitatively analyze the efficiency of two explicit representations in computational experiments, when we implement and develop Dual Decomposition method.

Following the studies in Carøe and Schultz (1999), we further explore two different representations of the non-anticipativity constraints. For SUFLP with pure binary variables in the first stage, it is possible to compactly represent non-anticipativity. In particular, Carøe and Schultz (1999) propose the equality

$$\sum_{s\in S} H^s x^s = 0, \text{ where } H^1 = p^1 - 1 \text{ and } H^s = p^s, \text{ for } s = 2, \dots, |S|. \tag{3.25}$$

where $p^s$ is the probability of corresponding scenario $s$.

We refer to this representation as "$H$ constant." In our experiments, the lower bound generated by Lagrangian relaxation is found to be very weak if using this equality to replace the non-anticipativity constraints.

In Carøe and Schultz (1999), although it is claimed that the general format to represent the non-anticipativity will generate better relaxation bounds than the "$H$ constant" approach, a suitable matrix $H^s$ was not explicitly given and was not further studied in computational tests. Therefore, we explicitly give the following representation for the non-anticipativity constraints, which we determine empirically to perform better.

We first rewrite the non-anticipativity constraints as the following system of equalities:

$$\begin{aligned} x^s - x^{s+1} &= 0, && s = 1, \dots, |S| - 1; \\ -x^1 + x^{|S|} &= 0. \end{aligned}$$

then writing this system of equations in the form of $\sum_{s\in S} H^s x^s = 0$ can be done with the following definitions of $H^s$:

$$H^1 = \begin{pmatrix} I \\ 0 \\ \vdots \\ \vdots \\ 0 \\ -I \end{pmatrix}, H^2 = \begin{pmatrix} -I \\ I \\ 0 \\ \vdots \\ \vdots \\ 0 \end{pmatrix}, \dots, H^j = \begin{pmatrix} 0 \\ \vdots \\ -I \\ I \\ 0 \\ \vdots \end{pmatrix} \begin{array}{l} \\ \\ (j\text{-}1)^{th}\text{block} \\ j^{th}\text{block} \\ \\ \\ \end{array}, \dots, H^{|S|} = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ -I \\ I \end{pmatrix}, \quad (3.26)$$

where $I$ is the identity matrix.

We refer to this representation as "$H$ matrix." If there are $|N|$ first stage variables, then the $H$ matrix is a $|S||N| \times |N|$ matrix with all binary entries. We find the Lagrangian bound generated from using this representation of non-anticipativity to be much stronger than "$H$ constant."

**Corollary 2.** *The "H constant" is only a necessary but not a sufficient condition for the non-anticipativity condition.*

**Proof**: First, if the copies of the first-stage variables, $x^s, s = 1, ..., |S|$, satisfy the non-anticipativity condition, that means

$$x^1 = x^2 = \cdots = x^{|S|}, \quad (3.27)$$

which also satisfy the "$H$ constant" equality (3.25), since

$$\begin{aligned} \sum_{s \in S} H^s x^s &= (p^1 - 1)x^1 + p^2 x^2 + \cdots + p^{|S|} x^{|S|} \\ &= p^1 x^1 - x^1 + p^2 x^1 + \cdots + p^{|S|} x^1 \\ &= (p^1 + p^2 + \cdots + p^{|S|}) x^1 - x^1 \\ &= x^1 - x^1 = 0. \end{aligned}$$

However, the copies of the first-stage variables, $x^s, s = 1, ..., |S|$, that satisfy the "$H$ constant" equality may not be identical. For example, if $x^{s'} = 0$, for $s' = 3, 4, ..., |S|$,

$x^2 = 1$, and let

$$x^1 = \frac{p^2}{1-p^1},$$

then the "$H$ constant" equality (3.25) is still satisfied, since

$$\begin{aligned}\sum_{s\in S} H^s x^s &= (p^1-1)x^1 + p^2x^2 \\ &= (p^1-1)\frac{p^2}{1-p^1} + p^2 = 0.\end{aligned}$$

Therefore, the "$H$ constant" is only a necessary but not a sufficient condition for the non-anticipativity condition.

**Corollary 3.** *The "H matrix" is a necessary and sufficient condition for the non-anticipativity condition.*

**Proof**: Rewriting the system of equations $\sum_{s\in S} H^s x^s = 0$ with the definition in (3.26) for matrix $H^s$:

$$\begin{array}{cccccl} Ix^1 & -Ix^2 & & & & = 0 \\ & Ix^2 & -Ix^3 & & & = 0 \\ & & \dots & & & = 0 \\ & & & Ix^{|S|-1} & -Ix^{|S|} & = 0 \\ -Ix^1 & & & & +Ix^{|S|} & = 0. \end{array}$$

It implies $x^1 = x^2$ in the first equation, $x^2 = x^3$ in the second equation, ..., $x^{|S|-1} = x^{|S|}$ in the second last equation. The last equation is redundant and therefore $x^{|S|} = x^1$. So the "$H$ matrix" equality (3.26) implies the copies of the first-stage variables are all identical and vice versa. Therefore, it proves that "$H$ matrix" is a necessary and sufficient condition for the non-anticipativity condition.

### 3.2.2 Dynamic Lagrangian Bound Computation

One reason the dual decomposition method (Carøe and Schultz (1999)) is time consuming is because it computes the Lagrangian bound at each node of the branch and bound

(B&B) tree in the instances with a large number of scenarios. In our algorithm, we dynamically determine whether the Lagrangian dual process will be computed for a current node in the B&B tree, thereby allowing us to avoid some unnecessary computations. This is the first time that the dynamic computation of the Lagrangian bound has been implemented in searching a B&B tree to solve a stochastic MIP.

In the B&B literature, the root node is at level 0 (highest level) and the branching region at the root node is the entire feasible region of the original problem. Nodes at deeper levels of the B&B tree have feasible regions that are smaller. If we compute the Lagrangian dual at higher level nodes, the resulting relaxed bound may be weak since the branching region is over a larger feasible region. On the other hand, it may not be desirable to compute the Lagrangian dual at deeper nodes since there are so many and any potential fathoming that would occur would apply to a smaller number of nodes. We seek a trade-off between the level of the node at which to compute the Lagrangian dual bound and the effectiveness of the bound.

Intuitively, computing the lower bound starting at some "mid-level" depth in the tree should produce an effective trade-off. We verify this claim in our computational results. If the Lagrangian dual is not computed at some node, some component of the first stage variable will be chosen and fixed to be 0 or 1 based upon the dispersion calculated at the root node. We use $l$ to indicate a depth and $k$ to record the depth level of the node currently being searched in the B&B tree. For a node, if its depth $k$ is greater than $l$, the Lagrangian dual will be computed, otherwise it will not be computed.

The following lines of pseudo code illustrate when to compute Lagrangian bounds in our dynamic framework.

```
initial $l = \text{int}(|N|/2)$                                  (1)
$k \leftarrow$ the level of the current node (sub-problem)       (2)
if $k \geq l$                                                     (3)
    solve the Lagrangian dual;                                    (4)
    if relaxed bound $\geq$ global feasible bound                 (5)
        current node is fathomed;                                 (6)
        $l = l - 1$;                                              (7)
    else                                                          (8)
        if $l \geq \text{int}(|N|/2)$                             (9)
            $l = l + j_1$;                                        (10)
        else                                                      (11)
            $l = l + j_2$;                                        (12)
        endif                                                     (13)
        branch on the current node;                               (14)
    endif                                                         (15)
endif                                                             (16)
return to line (2) with a new subproblem                          (17)
```

Box 1: Dynamic Lagrangian Computation

The threshold $l$ is initially set to be the integer part of $N/2$ (line (1)) which has the effect of initially allowing only nodes whose depth are greater than or equal to the mid-depth of the deepest possible tree (i.e., a tree with $|N|$ levels) to have their Lagrangian bounds to be computed.

If the Lagrangian bound of the current node is computed and the node is fathomed (lines (5)–(7)), then no child nodes will be generated. In this case, the next node to be considered will be either the sibling or the parent of the current node since we use the depth-first method in node searching. Because it is often the case in experiments that such nodes could provide useful Lagrangian bounds to further fathom the B&B tree, we set $l = l - 1$ (line (7)). The rationale is that we can already fathom some nodes below the level of current threshold, we would like to reduce the threshold to check whether nodes at higher level can be fathomed.

On the other hand, if the current node has its Lagrangian dual bound computed and it cannot be fathomed, then two child nodes will be generated and added to the B&B tree. Now we may assume that the Lagrangian dual bound calculated at the child nodes may probably not be tight enough to fathom the node. In order to more efficiently fathom nodes at the lower level of the B&B tree, we increase the threshold $l$. So $j_1$ and $j_2$ should be both greater than 1. If $j_1 = j_2 = 1$, the child nodes will still be computed in the Lagrangian dual.

We want to distinguish different jump steps based on the current $l$ level. The rationale here is that if the Lagrangian bound is computed at a node at a higher level (corresponding to $l < \text{int}(|N|/2)$) in the B&B tree but the node is not fathomed, then the threshold $l$ should be increased more than in the case with a node at a deeper level (corresponding to $l \geq \text{int}(|N|/2)$). Therefore, we set $j_2 > j_1$.

In all our SUFLP tests in this chapter, the range of the first stage variable is not large, $5 \leq N \leq 30$. Therefore we only set $j_1 = 2$ and $j_2 = 3$. We also test on $j_1 = 2,\ j_2 = 4$ and $j_1 = 3,\ j_2 = 4$ and observe the iterative jumping forth and back happened so frequently that it affected the efficiency in the process of searching B&B tree. For further work on solving other SMIPs with a large number of first stage variables, different jump steps should be tested in order to efficiently compute the Lagrangian bound and fathom more nodes in the B&B tree.

Since $l$ is set according to whether the nodes are fathomed or not, and that is not known *a priori*, the decision to compute the Lagrangian dual is dynamic. We will see in our experiments that the dynamic strategy is effective in conjunction with the basic dual decomposition framework.

## 3.2.3 Dynamic Dual Decomposition ($D^3$) Method

We now present in detail the entire dual decomposition-based branch and bound algorithm with dynamic bound selection as defined above for the two-stage SUFLP.

Define $(P)^t$ to be the collection of all of the scenario subproblems at the $t$-th iteration. Define a scenario subproblem $(P_s^t)$ in scenario $s$ as:

$$\begin{aligned}
z_t^s = \min \quad & \sum_{i \in N} f_i^t u_i^{s,t} + \sum_{i \in N} h_i^{s,t} x_i^{s,t} + \sum_{i \in N, j \in D} c_{i,j}^{s,t} y_{i,j}^{s,t} \\
\text{s.t.} \quad & \sum_{i \in N} y_{i,j}^{s,t} \geq d_j^{s,t} && \forall j \in D, \forall s \in S \\
& y_{i,j}^{s,t} \leq M(u_i^{s,t} + x_i^{s,t}) && \forall i \in N, \forall j \in D, \forall s \in S \\
& u_i^{s,t}, x_i^{s,t} \text{ binary} && \forall i \in N, \forall s \in S \\
& y_{i,j}^{s,t} \geq 0 \text{ integer} && \forall i \in N, \forall j \in D, \forall s \in S
\end{aligned}$$

Let $\bar{u}$ and $\bar{z}$ denote the incumbent best solution and corresponding objective value for the SUFLP. Define the dispersion of $u_i$ (after solving scenario sub-problems in $(P)^t$) as $\Delta u_i = \max_{s',s''}\{|u_i^{s'} - u_i^{s''}|, \text{ for } s', s'' \in S\}$. Define the probability weighted average of the objective values and first stage variables over all scenario solutions as $z' = \sum_{s \in S} p^s z^s$ and $u' = \sum_{s \in S} p^s u^s$.

```
Step 1. (Initialization): Set $L = \emptyset, k = 0, t = 0, \bar{z} = +\infty, l = \text{int}(|N|/2)$.
Step 2. (Root-node): Solve $(P)^0$.
        If $u^{s'} = u^{s''}, \forall s', s'' \in S$, then $\bar{z} = z'$, $\bar{u} = u'$. Terminate.
        Otherwise, $u'_i = \sum_{s \in S} p^s u^s_i$.
Step 3. (Feasible solution): $\bar{u}_i = \text{round}(u'_i)$. Then solve $(P)^0$ with fixed $\bar{u}_i$.
        Calculate $z'$ and set $\bar{z} = z'$.
Step 4. (Branching): Select $i$ with the largest $\Delta u_i$ and add to $L$
        two new problems $(P)^{t+1}$ with $u^s_i = 0, \forall s$ and $(P)^{t+2}$ with $u^s_i = 1, \forall s$.
        Set $t = t + 2$.
Step 5. (Problem selection and relaxation): Move $(P)^t$ from $L$ and solve it.
        If $u^s \in \{0,1\}^{|N|}, \forall s$; set $\bar{u} = u^s$.
            Solve $(P)^t$ with fixed $\bar{u}$.
            Calculate $z'$ and set $\bar{z} = \min\{\bar{z}, z'\}$. Go to Step 7.
        Otherwise, $k \leftarrow$ the node level of $(P)^t$.
        If $k > l$,
            Solve the Lagrangian dual of $(P^t)$ and obtain a lower bound $z_{LD}$.
            Go to Step 6.
        else
            Go to Step 4.
Step 6. (Bounding): If $z_{LD} \geq \bar{z}$, $l = l - 1$. Go to Step 7.
        Otherwise, if $u^{s'} = u^{s''}, \forall s'$ and $s''$, let $\bar{z} = \min\{\bar{z}, z_{LD}\}$. Go to Step 7.
        If $\exists s'$ and $s'', u^{s'} \neq u^{s''}$,
            calculate $u'_i$ and dynamically decide $l$ (ref to Box 1).
            Go to Step 3.
Step 7. (Termination): If $L = \emptyset$, then optimal solution $(u^*, x^*, y^*)$.
        If no such $(u^*, x^*, y^*)$ exists (e.g., $\bar{z} = +\infty$), then SUFLP infeasible.
        Otherwise, if $L \neq \emptyset$, go to Step 5.
```

Box 2: Pseudo code of $D^3$ method

**Depth-first Tree Search**

We use the depth-first method as our node searching strategy to develop the feasible bound.

**Optimality**

The algorithm will stop in a finite number of iterations because the feasible region of the first stage variables has a finite number of integer points. The final global feasible bound will be the optimal solution. All nodes in the B&B tree have been either computed or fathomed and the gap between the relaxed bound and the feasible bound vanishes.

**Sub-gradient heuristic**

In Step 5, the Lagrangian dual computation is time-consuming. We do not always need the sub-gradient method to converge and may stop at any iteration and use the current solution of the Lagrangian dual as the relaxed bound. In our algorithm, we dynamically set the varying number of iterations as the stopping criterion in the sub-gradient method. The varying number of iterations is based on the level of the node being searched in the B&B tree as well. At high levels in the B&B tree, the sub-gradient iterations are designed to be fewer. Since the feasible region is larger at the node in higher level, the sub-gradient method will take a longer time to converge and the relaxed bound will not be very tight. When the algorithm dives deeper, we usually find a good global feasible bound. More iterations will lead to a tighter relaxed bound, which is critical to the fathoming process.

## 3.3 Computational Experiments

### 3.3.1 Two-stage SUFLP Computational Results

To illustrate the effectiveness of the $D^3$ method on SUFLP problems, we compare it with the CPLEX 12.4 integer programming solver (IBM ILOG CPLEX Optimization Studio, 2011) to solve the corresponding DEP. The CPU times for the following experiments account for finding a global optimal solution. Since the $D^3$ approach is an exact method, all of the nodes in the B&B tree are either calculated or fathomed in order to verify that the solution is a global optimum.

The proposed $D^3$ is programmed in Visual Studio 6.0 by VC++ language. It includes implementing the B&B processing, Lagrangian dual calculations, H-matrix data manipulations and subgradient algorithms by VC++. The subproblems in the B&B processing are solved in VC++ invoking the CPLEX callable library.

In order to compare the computations of the CPLEX MIP solver to solve the whole model directly, we populated the whole model in the CPLEX callable library by VC++ directly. As we have mentioned before that the scenario sub-problems have the same structure of the original problem, the VC++ program of populating the whole model data to CPLEX callable library is similar to the VC++ program of populating a scenario sub-problem to CPLEX. Unfortunately, without any pre-processing techniques, VC++ fails to populate all data of large-size models to CPLEX callable library directly. Then we have to use AMPL to model the whole SUFLP and invoke CPLEX MIP solver to solve it. As the state-of-the-art commercial modeling software, AMPL conducts various advanced techniques on pre-process data or pre-solve the model before it invokes CPLEX solver to solve problems, but the $D^3$ method lacks the capacity of pre-processing data or pre-solving problems.

In the literature (see Silva (2004), Chu and Beasley (1997) and Kwon *et al.* (2008)), test instances of SMIP are based on randomly generated data. We generate the data for two-stage SUFLP instances as follows. We recognize that different data may have different distributions, but our primary goal is to show that the $D^3$ method can solve problem instances of SMIP, although we did not use benchmark data, for example, benchmark instances which were generated or collected in Beasley (1990). In Chapter 5, we present computational experiments on an extension of $D^3$, called $D^3$+PBDM, using the well-known OR-Library benchmark set on capacitated facility location problems (Beasley (1990)), as well as randomly generated data which are generated in a similar way as here. One may observe that the proposed $D^3$+PBDM algorithm and the CPLEX MIP solver can solve instances with benchmark data more easily than randomly generated data which we generate in Chapter 5 following the same parameter configuration as here.

We generate setup costs in the first stage in the range [370, 730] with uniform distribution. The setup costs in the second-stage are generated by adding additional costs, $\Delta h_i^s$, which are uniformly distributed in the range [30, 70], to corresponding setup costs in the same location in the first stage. Similarly, the serving costs and the demands are also randomly generated for different scenarios in the range [0.2, 2.0] and [500, 1500], respectively. The ranges of data are adapted from facility distributions in Management Information Systems of Municipal and Provincial Power Bureaus in Yantai city, Qingdao city and Tianjin city in 1998 - 2001. The sizes of instances are listed in Table 1 in the Appendix A.

In Table 3.1 and Table 3.2, we list the CPU times and the solutions from the $D^3$ method and from the CPLEX 12.4 solver. Table 3.1 summarizes the results of the comparison where the serving variables $y$ are integer and Table 3.2 summarizes the results when continuous serving variables are used. After removing the integer restriction (3.5), the CPLEX MIP 12.4 solver is able to solve some moderate size SUFLP problems. The "Nodes" column under the $D^3$ method lists the total nodes searched in the B&B tree. A problem instance listed as $x$F_$y$P_$z$S_I means the problem has $x$ potential facility locations, $y$ clients, $z$ scenarios and the serving variables with integer constraints. In some experiments, the CPLEX solver failed to find the optimal solution and stopped with a feasible solution reporting "unrecoverable failure with integer solution" even for the small size examples in Table 3.1. This was because the linear programming relaxation of the DEP was extremely weak to the point that the CPLEX solver did not allow the B&B process to continue. In one of these cases, for example, the CPLEX solver created about half a million nodes of the B&B tree, of which over 90% were still active since the gap between the relaxed bound and the feasible bound was over 46%. The cases with unrecoverable failure are shown with the times marked *. "Time" is the total CPU time in seconds until the optimal solution is obtained or until the solver stops. All experiments

| Experiment | | $D^3$ | | | CPLEX MIP | |
|---|---|---|---|---|---|---|
| | Problem | Time | Solution | Nodes | Time | Solution |
| 1 | 10F_50P_50S_I | 169.5 | 23594.5 | 115 | 2.0 | 23594.5 |
| 2 | 12F_40P_50S_I | 137.4 | 18887.0 | 71 | 10.9 | 18887.0 |
| 3 | 18F_25P_50S_I | 103.5 | 12091.2 | 67 | 63.6 | 12091.2 |
| 4 | 18F_25P_100S_I | 190.3 | 11995.9 | 67 | 216.5 | 11995.9 |
| 5 | 18F_25P_200S_I | 457.3 | 12073.6 | 67 | 481.7 | 12073.6 |
| 6 | 18F_25P_250S_I | 558.7 | 11985.9 | 67 | 816.6* | 11986.0 |
| 7 | 18F_25P_300S_I | 708.6 | 11861.9 | 67 | 903.6* | 11861.9 |
| 8 | 20F_30P_50S_I | 146.1 | 13720.8 | 81 | 65.2 | 13720.8 |
| 9 | 20F_30P_80S_I | 247.6 | 13742.9 | 81 | 841.9 | 13742.9 |
| 10 | 20F_30P_100S_I | 328.9 | 13799.0 | 81 | 787.5 | 13799.0 |
| 11 | 20F_30P_150S_I | 541.4 | 13625.4 | 81 | 889.6* | 13693.6 |
| 12 | 20F_30P_200S_I | 647.3 | 13687.0 | 81 | 788.8* | 13687.0 |
| 13 | 20F_50P_50S_I | 415.8 | 20949.0 | 113 | 485.0 | 20949.0 |
| 14 | 20F_50P_80S_I | 1046.9 | 20978.5 | 167 | 900.3* | 20978.5 |
| 15 | 30F_50P_100S_I | 2947.8 | 20068.1 | 219 | 3216.2* | 20098.4 |
| 16 | 40F_50P_50S_I | 2537.1 | 19536.9 | 261 | 1656.1* | 19547.8 |
| 17 | 50F_50P_100S_I | 10604.1 | 19298.9 | 387 | 3540.9* | 19391.9 |

Table 3.1: Computational results of two-stage SUFLP with integer serving variables

were tested on a Laptop with Quad Intel Core i3 2.5 GHz CPU processors and 4 GB of RAM.

In Table 3.1, we see that the $D^3$ method outperforms the CPLEX solver on large-size problem instances. In some cases CPLEX terminated without solving to optimality (e.g., #6, #7, #11 etc.). For small-size problems (e.g., #4, #5, #9, #10 and #13), the $D^3$ method performs better than the CPLEX MIP solver if both can solve the problems.

In Table 3.1 and subseqent tables, the shorter CPU times of either the $D^3$ method or the CPLEX MIP solver are shaded in light yellow when both methods solve the global

| Experiment | | $D^3$ | | | CPLEX MIP | |
|---|---|---|---|---|---|---|
| | Problem | Time | Solution | Nodes | Time | Solution |
| 1 | 15F_50P_50S_C | 211.3 | 22066.7 | 95 | 31.8 | 22066.7 |
| 2 | 15F_50P_100S_C | 599.4 | 22110.6 | 113 | 1389.7* | 22110.8 |
| 3 | 15F_50P_200S_C | 1273.7 | 22092.8 | 129 | 8288.7* | 22092.9 |
| 4 | 20F_50P_50S_C | 281.1 | 20949.0 | 129 | 418.9 | 20949.0 |
| 5 | 20F_50P_80S_C | 817.8 | 20978.5 | 167 | 1350.3* | 20979.3 |
| 6 | 20F_50P_100S_C | 609.4 | 20919.8 | 119 | 2182.4* | 20919.9 |
| 7 | 20F_50P_120S_C | 868.3 | 20936.5 | 133 | 2561.5* | 20953.8 |
| 8 | 20F_50P_150S_C | 1249.9 | 20996.5 | 141 | 2203.2* | 21101.5 |
| 9 | 20F_50P_200S_C | 1601.9 | 20964.1 | 137 | 1096.6* | 20986.9 |
| 10 | 20F_50P_250S_C | 2266.1 | 20999.8 | 143 | 1954.6* | 21023.2 |
| 11 | 30F_50P_30S_C | 492.0 | 20125.0 | 205 | 2067.9* | 20150.4 |
| 12 | 30F_50P_50S_C | 931.7 | 20134.9 | 222 | 2473.3* | 20138.8 |
| 13 | 30F_50P_100S_C | 1826.5 | 20068.1 | 205 | 1168.1* | 20096.5 |
| 14 | 40F_50P_50S_C | 1699.7 | 19536.9 | 261 | 2521.4* | 19540.6 |

Table 3.2: Computational results of two-stage SUFLP with continuous serving variables

optimum. We also mark the times in light yellow for cases when CPLEX MIP solver cannot solve for a global optimum and collapse in a longer time than the CPU time of the $D^3$ method solving a global optimum (e.g., #6, #7, #11, etc. in Table 3.1; #2, #3, #5, etc. in Table 3.2). We mark the times in light green for cases when the CPU time of the $D^3$ method is longer than the time when CPLEX solver collapses with unrecoverable failure (e.g., #14, #16 and #17 in Table 3.1; #10 and #13 in Table 3.2).

In Table 3.1, we also examine two groups of sensitivity analyses. We solve a group of instances with the same number of facilities and clients, but with an increasing number of scenarios. The first group of instances are in tests from #3 to #7 with 18 facilities and 25 clients, but the number of scenarios increases from 50 to 300. The CPU times of the proposed $D^3$ method are nearly linear when compared to the number of scenarios, but the CPU times of CPLEX solver increase almost exponentially until the solver fails with memory errors. The second group of instances are in tests #8 - #12 with 20 facilities and 30 clients, but the number of scenarios increases from 50 to 200. One may also notice the CPU times of the $D^3$ method are close to linear in the number of scenarios, but CPU times of CPLEX solver increase dramatically.

The $D^3$ method decomposes the whole model into scenario subproblems. In a group of aforementioned sensitivity tests, the scenario subproblems have the same size, which is only related to the number of facilities and clients. Although the subproblems are still MIPs, the MIP solver can solve each small size subproblem quickly. The processing times of the B&B method and Lagrangian dual algorithm in VC++ dominate the CPU times of solving subproblems. Therefore, the $D^3$ method solves a group of problems with the same number of facilities and clients almost linearly in the number of scenarios. On the contrary, the CPLEX MIP solver cannot solve the whole models, which are large size MIPs, in linear CPU times as the number of scenarios increases.

In Table 3.2, we notice that the CPLEX solver performs better for a small instance #1. But $D^3$ outperforms CPLEX on a large size instance #4, when both $D^3$ and the CPLEX solver obtained the optimal solutions. For problems #2, #3, and #5 - #14, the CPLEX solver terminated without solving to optimality.

The sensitivity analysis presented in Table 3.2 is not as obvious as the one in Table 3.1. Since we have relaxed serving variables from integer to continuous, the whole model contains only binary variables with location decisions and a large number of continuous variables with serving demands from facilities to clients.

One may wonder how many of the randomly generated data are the same in the instances with the same number of facilities and clients but only different number of scenarios. In our VC++ implementation, the coefficients $f_i$, $h_i^s$, $c_{i,j}^s$, and $d_j^s$ are generated in loops with an array of uniformly distributed random numbers in [0 1] by calling a function "rand()." The pseudo code of the data generation part is as follows:

```
Set $k = 1$;
**To generate $f_i$:**
For $i = 1$ to $|I|$
    $r_k = \text{rand}()$;
    $f_i = 370 + (730 - 370) \times r_k$ ;
    $k = k + 1$;
End for
**To generate $h_i^s$:**
For $i = 1$ to $|I|$
    For $s = 1$ to $|S|$
        $r_k = \text{rand}()$;
        $\Delta h_i^s = (70 - 30) \times r_k$ ;
        $h_i^s = f_i + \Delta h_i^s$ ;
        $k = k + 1$;
    End for
End for
**Later on, generate $r_k$ in serial and generate $c_{i,j}^s$ in corresponding loops of $i$, $j$, $s$
and $d_j^s$ in loops of $j$, $s$.**
```

Box 3: Pseudo Code of Random Data Generation in $D^3$ method

For example, if comparing the data of #8 and #10 in Table 3.1 with 20 facility and 30 clients, but 50 or 100 scenarios respectively, one may notice that the array of random data, $r_k, k = 1, 2, \cdots$, will be same in both tests since our VC++ implementation will always use same random seeds by default. We would be able to first generate random seeds by CPU clock time, then the random data, $r_k$, which were generated in different times, would always be different. But in this way, we can not repeat a test with the same set of data. Although the random data array, $r_k$, are the same, only $f_i$ for $i = 1, \cdots, 20$ and $h_i^s$ for $i = 1, s = 1, \cdots, 50$ are the same. Then later $h_i^s, i = 2, s = 1, \cdots, 50$ in #8 and $h_i^s, i = 1, s = 51, \cdots, 100$ in #10 will continue using the same random data array $r_k$ for $k = 71, 72, \cdots$. Hence, all other coefficients will be different sine they are generated by different random data. There are 32,520 coefficients ($= 20\ (f_i) + 20 \times 50\ (h_i^s) + 20 \times 30 \times 50\ (c_{i,j}^s) + 30 \times 50 (d_j^s)$) in #8 and 65,020 coefficients in #10, but only 70 coefficients are the same in both #8 and #10.

In Table 3.1 and subsequent tables, we only run one test for each parameter configuration, for example, we only run one test for the SUFLP with 20 facility locations, 30 clients and 50 scenarios of Instance #8 in Table 3.1. In Table 3.3, we also run multiple tests on three groups of data set with the same model size and the same parameter configuration of Instance #8 in Table 3.1. First we only change the increment of setup costs in the

| Experiment | | $D^3$ | | | CPLEX MIP | |
|---|---|---|---|---|---|---|
| | Problem | Time | Solution | Nodes | Time | Solution |
| 20F_30P_50S_I with $\Delta h$ = [30 70] | | | | | | |
| 8_$\Delta h$_1 | $r_{\text{skip}} = 0$ | 146.1 | 13720.8 | 81 | 65.2 | 13720.8 |
| 8_$\Delta h$_2 | $r_{\text{skip}} = 30$ | 173.1 | 13967.9 | 81 | 371.4 | 13967.9 |
| 8_$\Delta h$_3 | $r_{\text{skip}} = 60$ | 157.6 | 13872.4 | 81 | 142.5 | 13872.4 |
| 8_$\Delta h$_4 | $r_{\text{skip}} = 120$ | 136.3 | 13904.0 | 81 | 145.5 | 13904.0 |
| 8_$\Delta h$_5 | $r_{\text{skip}} = 240$ | 153.5 | 13891.7 | 81 | 93.2 | 13891.7 |
| 8_$\Delta h$_6 | $r_{\text{skip}} = 360$ | 137.1 | 13840.0 | 81 | 30.0 | 13840.0 |
| 8_$\Delta h$_7 | $r_{\text{skip}} = 500$ | 156.0 | 13897.8 | 81 | 234.8 | 13897.8 |
| 20F_30P_50S_I with $\Delta h'$ = [45 105] | | | | | | |
| 8_$\Delta h'$_1 | $r_{\text{skip}} = 0$ | 206.3 | 13885.9 | 109 | 197.6 | 13885.9 |
| 8_$\Delta h'$_2 | $r_{\text{skip}} = 30$ | 251.7 | 14122.0 | 111 | 2416.8 | 14122.0 |
| 8_$\Delta h'$_3 | $r_{\text{skip}} = 60$ | 263.6 | 14032.0 | 107 | 916.8 | 14032.0 |
| 8_$\Delta h'$_4 | $r_{\text{skip}} = 120$ | 152.8 | 14053.4 | 81 | 422.3 | 14053.4 |
| 8_$\Delta h'$_5 | $r_{\text{skip}} = 240$ | 207.8 | 14043.6 | 99 | 225.9 | 14043.6 |
| 8_$\Delta h'$_6 | $r_{\text{skip}} = 360$ | 160.9 | 13997.7 | 81 | 1017.6* | 14272.7 |
| 8_$\Delta h'$_7 | $r_{\text{skip}} = 500$ | 246.2 | 14057.1 | 109 | 407.9 | 14057.1 |
| 20F_30P_50S_I with $\Delta h''$ = [60 140] | | | | | | |
| 8_$\Delta h''$_1 | $r_{\text{skip}} = 0$ | 406.2 | 14047.7 | 157 | 871.8* | 14047.7 |
| 8_$\Delta h''$_2 | $r_{\text{skip}} = 30$ | 400.5 | 14272.5 | 155 | 1140.9* | 14272.7 |
| 8_$\Delta h''$_3 | $r_{\text{skip}} = 60$ | 436.3 | 14187.1 | 159 | 1092.8* | 14187.1 |
| 8_$\Delta h''$_4 | $r_{\text{skip}} = 120$ | 221.7 | 14200.0 | 95 | 948.0 | 14200.0 |
| 8_$\Delta h''$_5 | $r_{\text{skip}} = 240$ | 339.9 | 14191.0 | 131 | 1237.3 | 14191.0 |
| 8_$\Delta h''$_6 | $r_{\text{skip}} = 360$ | 267.6 | 14151.5 | 111 | 710.8 | 14151.5 |
| 8_$\Delta h''$_7 | $r_{\text{skip}} = 500$ | 430.4 | 14211.1 | 165 | 1505.4* | 14211.1 |

Table 3.3: Multiple tests for one instance on #8 in Table 3.1

second stage, i.e., $\Delta h = [30\ 70]$, the original increment in Table 3.1, in the first group, $\Delta h' = \Delta h \times 1.5 = [45\ 105]$ in the second group and $\Delta h'' = \Delta h \times 2 = [60\ 140]$ in the third group. Then in each group we skip various numbers of random data, $r_k$, in order to generate different coefficients by the same parameter configuration but the different arrays of uniformly distributed random numbers in each run. So the coefficients will be generated by the same parameter configuration but use the different arrays of random numbers, i.e. $r_k, k = 1, 2, \cdots$ in Instance #8_$\Delta h'$_1 and $r_k, k = 61, 62, \cdots$ in Instance #8_$\Delta h'$_3, respectively. More details on randomly generating data for multiple tests may refer to the pseudo code in Box 4.

```
Set $k = 1$;
**To skip some random $r_k$:**
For $i = 1$ to $r_{\text{skip}}$
    $r_k = \text{rand}()$;
    $k = k + 1$;
End for
**To generate $f_i$:**
For $i = 1$ to $|I|$
    $r_k = \text{rand}()$;
    $f_i = 370 + (730 - 370) \times r_k$;
    $k = k + 1$;
End for
**To generate $h_i^s$:**
For $i = 1$ to $|I|$
    For $s = 1$ to $|S|$
        $r_k = \text{rand}()$;
        $\Delta h_i^{s\prime} = (105 - 45) \times r_k$;      $(\Delta h_i^{s\prime\prime} = (140 - 60) \times r_k;)$
        $h_i^s = f_i + \Delta h_i^{s\prime}$;
        $k = k + 1$;
    End for
End for
**Later on, generate $r_k$ in serial and generate $c_{i,j}^s$ in corresponding loops of $i$, $j$, $s$ and $d_j^s$ in loops of $j$, $s$.**
```

Box 4: Pseudo Code of Random Data Generation in multiple tests

In Table 3.3, one may notice the optimal solutions are different in a group of multiple tests with the same setup cost increment, ($\Delta h$, $\Delta h'$, or $\Delta h''$) and all other parameter configurations, but the different arrays of random data $r_k$. In three groups of multiple tests with same parameter configuration, the $D^3$ method can solve the multiple tests in similar computational times but the CPLEX solver solves the multiple tests in much different computational times. Especially the CPLEX solver can not solve an optimal solution in Instance #8_$\Delta h'$\_6, but it can solve Instance #8\_$\Delta h'$\_1 and #8\_$\Delta h'$_5 for only around 200 seconds.

### 3.3.2 Comparing dual decomposition method to $D^3$ method

We graph the computational results in the B&B procedure of the dual decomposition method (Carøe and Schultz (1999)), which uses the "$H$ constant" approach to represent non-anticipativity constraints, and the $D^3$ method, which combines the "$H$ matrix" representation and the dynamic technique, in solving an instance of the SUFLP with 10 locations, 50 clients and 50 scenarios.

We graph the incumbent global upper bound and lower bound at each node. Results from using the original dual decomposition method with "$H$ constant" are shown in Figure 3.1. The Dual Decomposition method solved the problem after 6377 seconds of computation and searching 1053 nodes in B&B procedure. The lower bounds in most nodes are far from the incumbent upper bounds.

The $D^3$ method was used on the same instance and it took 282 seconds and searched 115 nodes to solve the problem. The lower bounds are much closer to the incumbent upper bounds and the incumbent upper bounds decrease very quickly to the optimal solution, as shown in Figure 3.2.

The algorithm applied to models with the "$H$ matrix" representation is more memory-intensive than with the "$H$ constant" representation. The "$H$ matrix" approach requires storage and manipulation of $|S|$ matrices with size $|S||N| \times |N|$ with binary elements,

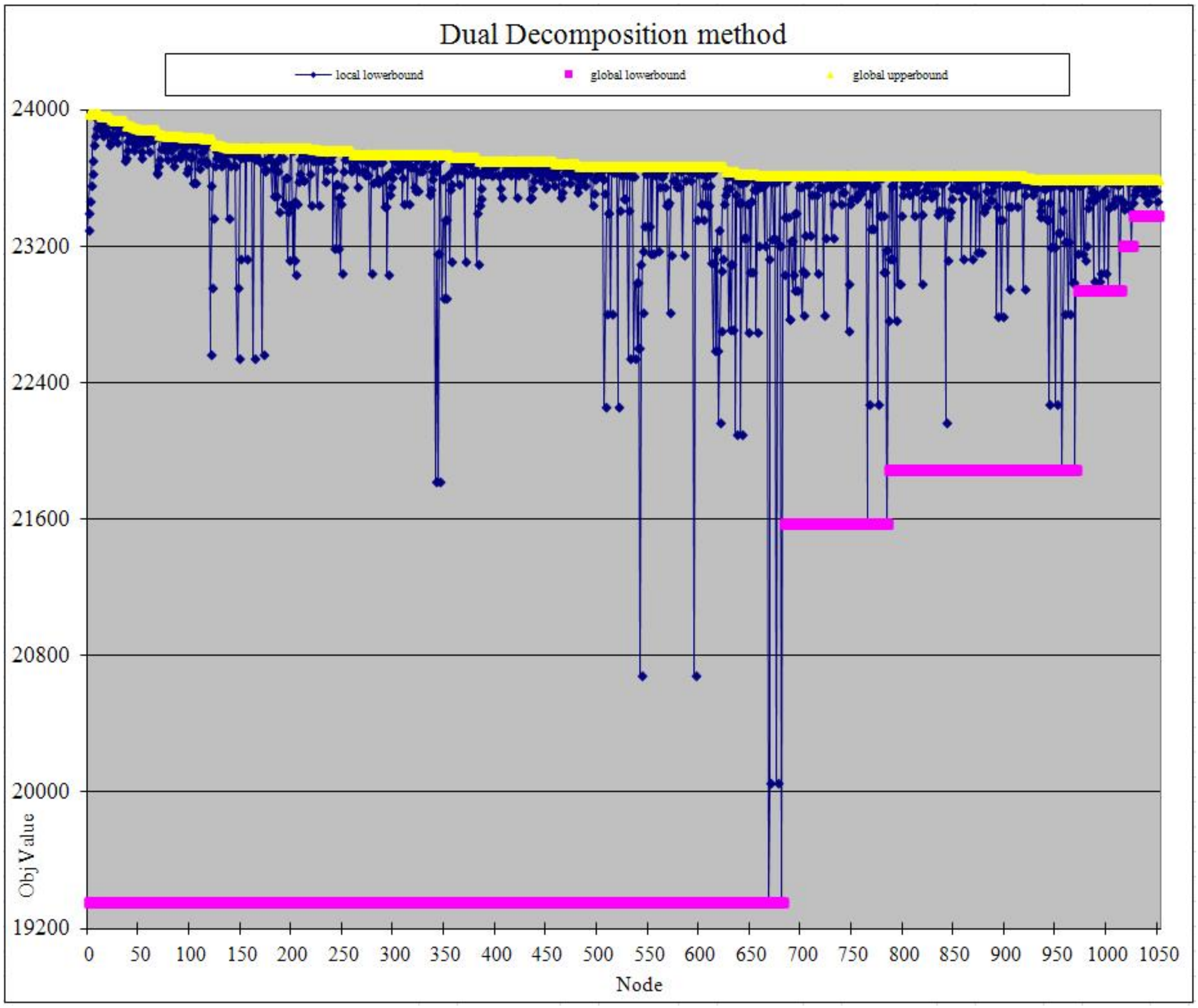

Figure 3.1: Dual decomposition method

but only $|S|$ floating point numbers are required in the "$H$ constant" approach. We will perform additional computational tests comparing equivalent "$H$ matrix" and "$H$ constant" representations on two-stage SUFLP instances generated in the next section.

The "$H$ matrix" representation offers tighter bounds than the "$H$ constant" one, but comes at the expense of larger storage requirements and processing due to the increased number of variables in $H$. Consequently, $D^3$ with "$H$ matrix" may not perform as well as the case with "$H$ constant" on SUFLP instances with many scenarios and with no integer

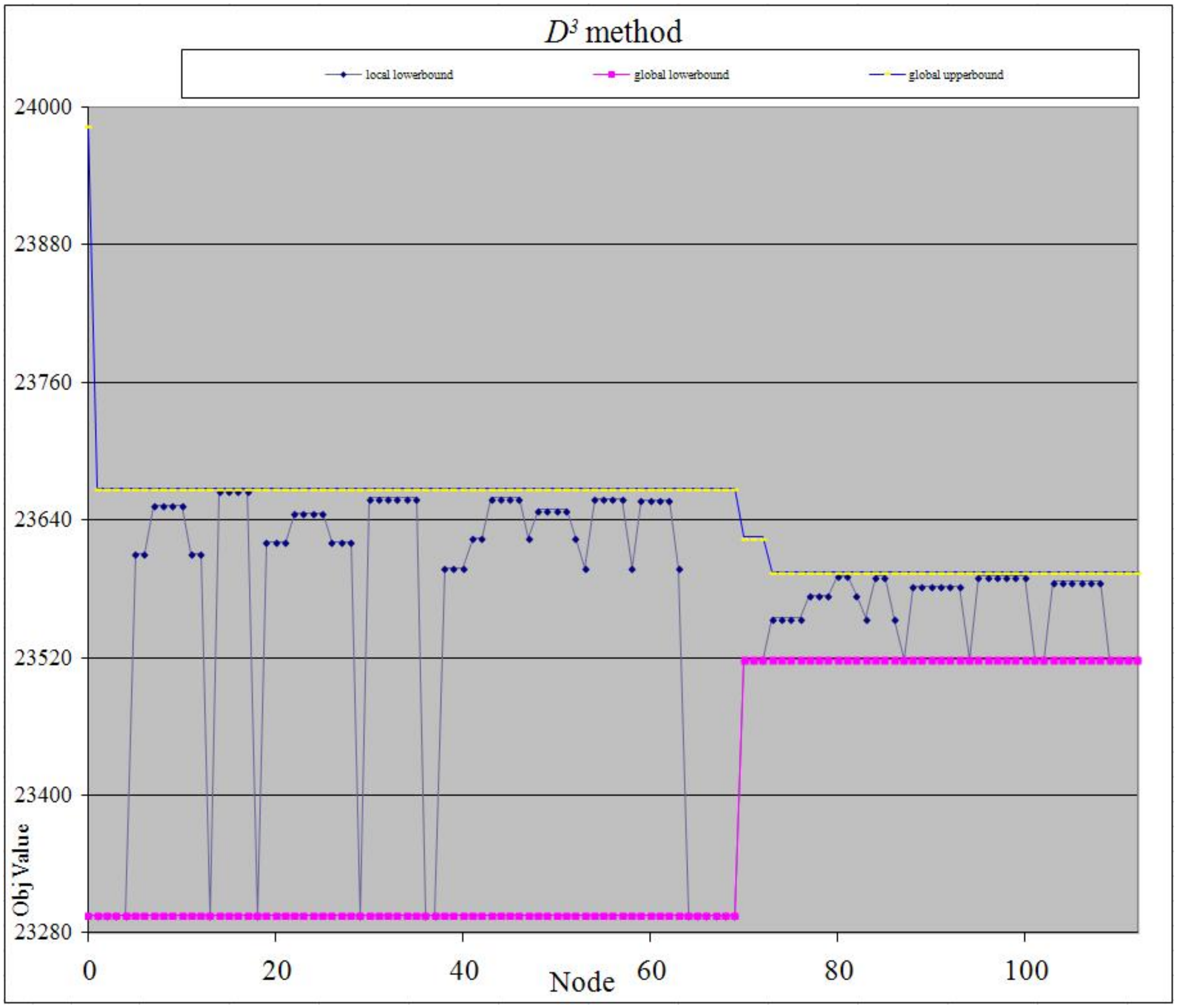

Figure 3.2: $D^3$ method

recourse.

## 3.3.3 Three-stage SUFLP Computational Results

In this section, we report the results of three-stage SUFLP experiments. All three-stage problems were solved under $D^3$ by branching on first stage decisions and treating the recourse in the second and third stages as two-stage DEP scenario subproblems.

| Experiment | | $D^3$ | | | CPLEX MIP | | |
|---|---|---|---|---|---|---|---|
| | Problem | Time | Solution | Nodes | Time | Solution | Gap |
| 1 | 3F_4P_10S_4R | 1.8 | 5727.5 | 11 | 0.2 | 5727.5 | |
| 2 | 3F_6P_10S_10R | 4.5 | 8273.7 | 15 | 2.7 | 8273.7 | |
| 3 | 3F_6P_15S_10R | 9.4 | 8226.9 | 15 | 34.7 | 8226.9 | |
| 4 | 3F_6P_20S_10R | 7.9 | 8142.9 | 13 | 38.8 | 8142.9 | |
| 5 | 3F_6P_30S_2R | 9.2 | 7453.5 | 15 | 620.5 | 7453.5 | |
| 6 | 3F_6P_30S_4R | 10.3 | 7828.1 | 15 | 1193.3 | 7828.1 | |
| 7 | 3F_6P_30S_10R | 15.9 | 8084.8 | 15 | 334.6* | 8087.5 | 5.63% |
| 8 | 5F_10P_20S_5R | 22.3 | 11870.9 | 35 | 356.2* | 11877.0 | 5.20% |
| 9 | 5F_10P_20S_10R | 31.8 | 12072.5 | 31 | 366.6* | 12072.5 | 7.32% |
| 10 | 5F_10P_20S_15R | 40.2 | 12169.9 | 31 | 345.6* | 12175.2 | 11.70% |
| 11 | 5F_10P_20S_20R | 48.7 | 12226.5 | 29 | 362.3* | 12226.5 | 11.07% |
| 12 | 5F_10P_20S_25R | 60.0 | 12234.6 | 29 | 348.1* | 12237.0 | 11.71% |
| 13 | 10F_10P_20S_10R | 66.7 | 11206.3 | 51 | 423.1* | 11231.2 | 9.23% |
| 14 | 10F_20P_20S_10R | 211.7 | 20256.4 | 77 | 438.3* | 20404.8 | 11.69% |
| 15 | 20F_20P_40S_10R | 1775.1 | 19336.0 | 187 | 687.9* | 19760.6 | 15.17% |

Table 3.4: Computational results of three-stage SUFLP with integer serving variables

We generated the first stage setup costs uniformly in the interval [370, 730]. The extra costs in the second stage and the third stage were generated for different scenarios uniformly in the range [30, 70] and [50, 100], respectively. Similarly, the serving costs were randomly generated for different scenarios in the range [0.2, 2.0] and [0.4, 2.5] in the second stage and the third stage, respectively. The demands were in the range [500, 1500] and [600, 2000] in the second stage and the third stage, respectively. The sizes of the instances are shown in Table 2 in the Appendix A.

| Experiment | | $D^3$ | | | CPLEX MIP | |
|---|---|---|---|---|---|---|
| | Problem | Time | Solution | Nodes | Time | Solution |
| 1 | 10F_10P_200S_4R | 275.9 | 10784.1 | 41 | 172.8 | 10784.1 |
| 2 | 10F_10P_300S_4R | 413.7 | 10790.9 | 41 | 280.7 | 10790.9 |
| 3 | 10F_10P_400S_4R | 522.9 | 10857.8 | 37 | 913.4 | 10857.8 |
| 4 | 15F_15P_100S_5R | 485.6 | 14905.0 | 71 | 337.6 | 14905.0 |
| 5 | 20F_20P_40S_5R | 876.4 | 18897.9 | 191 | 5127.4 | 18897.9 |
| 6 | 20F_20P_40S_10R | 1154.3 | 19336.0 | 173 | 4325.8* | 19336.0 |
| 7 | 20F_20P_40S_20R | 2145.9 | 19501.6 | 191 | 40436.5 | 19501.6 |
| 8 | 20F_20P_40S_40R | 5048.8 | 19613.3 | 177 | 30714.2* | 19613.3 |
| 9 | 30F_20P_20S_3R | 683.5 | 17253.5 | 249 | 243.7 | 17253.5 |
| 10 | 30F_20P_40S_5R | 1670.3 | 18115.5 | 257 | 2330.5* | 18117.2 |
| 11 | 30F_30P_30S_30R | 21400.4 | 27199.1 | 500 | 2730.2* | 27271.8 |

Table 3.5: Computational results of three-stage SUFLP with continuous serving variables

In Table 3.4, we list the CPU times (in seconds) and the solutions from the $D^3$ method and from using CPLEX MIP solver version 12.4 to solve the DEP. Unlike two-stage instances, we observe that the gaps remain pretty large although the current feasible solution may be optimal or close to the optimal solution, when CPLEX MIP solver stops with memory errors. We list the percentages of gaps between the incumbent feasible solution and the global relaxed bound in "Gap" column under "CPLEX MIP" in Table 3.4. One may note that the best feasible solution reported by CPLEX MIP solver is the same as the optimal value obtained by $D^3$ method in test #9 and #11. However the optimality gaps between the best feasible solutions and the lower bounds are 7.3% and 11.1% in the column of CPLEX MIP. Those large gaps are because CPLEX computes very weak lower bounds in B&B or B&C processing in solving the whole model and keeps more and more nodes still active until using up all memory.

Similar to the two-stage SUFLP, if $d$ and $e$ are integer, $y$ and $w$ will be integer. Relaxing the integrality of $y$ and $w$ gives us the semi-DEP formulation. In Table 3.5, we list the computational results for solving the large size semi-DEP instances with both the $D^3$ method and CPLEX MIP solver. Unlike the tests in Table 3.4, the gaps reported by the CPLEX solver in unsolved semi-DEP cases are not very large, all less than 1%, therefore the gaps are not listed in the table.

In Table 3.4, it is seen that the $D^3$ method performs better than the CPLEX MIP solver on DEP problems with integer serving variables. For small problem sizes (e.g., #3 - #6), the $D^3$ method was up to 116 times faster. For medium sizes (e.g., #7 - #14), CPLEX terminated without solving to optimality. The gaps of upper bounds and lower bounds reported by the CPLEX solver are on average 9.4%.

In Table 3.5, the $D^3$ method only performs better than the CPLEX solver for large instances of semi-DEP (e.g., #3, #5 and #7). In some cases (e.g., #6, #8, #10 and #11), CPLEX stops with memory error after a long time running. The number of integer variables are only a small portion of the three-stage semi-DEPs, where CPLEX MIP solver can take advantage of solving linear problems very efficiently. Generally, the $D^3$ method is better than the CPLEX solver in most instances of the three-stage semi-equivalent SUFLPs, which is similar to the two-stage instances.

## 3.3.4 Computational results of the developments in dual decomposition method

Since the $D^3$ method involves two main ideas (the "$H$ matrix" representation of non-anticipativity and dynamic Lagrangian dual calculation), we present results of experiments that aim to isolate their individual effects. As a result there are four cases to consider:

1) (HC) $H$ constant, no Dynamic (the Dual Decomposition method)

2) (HC+D) $H$ constant, Dynamic

3) (HM) $H$ matrix, no Dynamic

4) (HM+D) $H$ matrix, Dynamic (the $D^3$ method)

| Experiment | HC | | HC+D | | HM | | HM+D | |
|---|---|---|---|---|---|---|---|---|
| | Time | Nodes | T | N | T | N | T | N |
| 4F_5P_20S | 5.8 | 13 | 4.8 | 15 | 2.6 | 9 | 2.0 | 9 |
| 5F_10P_50S | 46.6 | 33 | 24.6 | 23 | 15.5 | 11 | 7.5 | 11 |
| 8F_30P_50S | 1,848.1 | 331 | 635.2 | 191 | 265.4 | 53 | 124.1 | 45 |
| 8F_30P_100S | 2,977.3 | 289 | 1,964.9 | 207 | 353.2 | 35 | 206.5 | 41 |
| 10F_20P_50S | 224.9 | 47 | 236.2 | 101 | 88.2 | 21 | 48.2 | 27 |
| 10F_25P_100S | 1,633.1 | 169 | 903.8 | 147 | 199.6 | 21 | 170.1 | 41 |
| 10F_40P_100S | 3,647.2 | 401 | 4,759.0 | 804 | 336.9 | 27 | 199.3 | 51 |
| 10F_50P_50S | 6,376.8 | 1053 | 3,899.1 | 1175 | 728.6 | 81 | 282.1 | 115 |
| 12F_40P_50S | 4,520.4 | 777 | 1,837.9 | 531 | 192.3 | 29 | 191.6 | 71 |
| 18F_25P_50S | 857.8 | 99 | 879.5 | 273 | 464.3 | 37 | 174.1 | 67 |
| 20F_30P_50S | 1,820.8 | 149 | 2,105.9 | 457 | 569.1 | 41 | 257.1 | 81 |
| 20F_30P_100S | 6,148.9 | 185 | 7,193.8 | 433 | 1,364.2 | 41 | 552.4 | 81 |

Table 3.6: Computational results of developments in four variants

The detailed computational results are in Table 3.6. The times and nodes of each case indicate the CPU time and the total nodes being searched while solving the problems.

From the Table 3.6, the $H$ matrix strategy and dynamic bounding strategy both contribute to significant savings in CPU time. The $D^3$ method in many cases requires only 4% to 35% of the CPU time of the standard dual decomposition method to solve the same SUFLP models.

We also summarize the comparison of the four variants in Table 3.7. In the entry of different methods, it shows the average and range of magnitudes of CPU times saved over all tests. We put 1 as benchmark in the top-left for "$H$ constant, no Dynamic"(HC). Then

| | No Dynamic | Dynamic |
|---|---|---|
| $H$ constant | 1 | 1.49 [0.77 , 2.91] |
| $H$ matrix | 7.01 [1.85 , 23.51] | 11.69 [2.91 , 23.59] |

Table 3.7: CPU times saved of developed methods on average and in a range

we calculate the ratio of times in column "HC" over times in column "HC+D" in Table 3.6 and put the average and range in the top-right for "$H$ constant, Dynamic." Similarly, the average and range of the ratio of times in column "HC" over times in column"HM" in Table 3.6 are put in the bottom-left for "$H$ matrix, No Dynamic." Finally, the ratio of times in column "HC" over times in column "HM+D" in the bottom-right for "$H$ matrix, Dynamic." For example, the values in the bottom-left indicate that the method with "$H$ matrix, No Dynamic" runs on average 7.01 times faster than the dual decomposition method (HC) of solving all tests, with the best case being 23.51 times faster and the worst case being 1.85 times faster. In the bottom-right, the $D^3$ method ("$H$ matrix, Dynamic") runs on average 11.69 times faster than the dual decomposition method with the best case being 23.59 times faster and the worst case being 2.91 times faster.

## 3.4 Conclusions

In this chapter, we consider a dynamic dual decomposition-based branch and bound strategy, $D^3$, to solve stochastic uncapacitated facility location problems (SUFLP). Our approach is an adaptation of the framework of Carøe and Schultz (1999) for stochastic integer programs where Lagrangian dual bounds are used in a branch and bound approach. The approach in $D^3$ does not compute Lagrangian bounds at every node, but instead dynamically determines whether this bound will be computed. In addition, a proposed matrix representation for non-anticipativity is seen to improve the strength of the Lagrangian bounds significantly. The matrix representation provides a stronger bound than the one of a single constant equivalent. The $D^3$ method efficiently solved moderate and large sized instances of SUFLP whose DEP could not be solved or solved much slower by the CPLEX MIP solver. Even if the integer restrictions on serving variables are removed, CPLEX MIP solver still could not solve large-size problems. The three-stage SUFLP model is also studied and solved by the $D^3$ method, which is found to be effective. Further research will consider the inclusion of more stages in SUFLP problems where ideas from branch and fix (see Alonso-Ayuso *et al.* (2003)) and dual decomposition may be combined.

# Chapter 4

# $D^3$+Column Generation on SSPP

## 4.1 Introduction to Column Generation and SSPP

The Set Packing Problem (SPP) has numerous applications and its polyhedral structure has been well studied over the last century (see Garey and Johnson (1979)). However, comparing what has been extensively studied on the SPP, only a few attempts to solve a Stochastic version of SPP (SSPP) can be found in the literature. The SSPP is an exciting problem motivated by new applications in industry. The uncertainty often leads it impossible to choose assuring scenarios where the problem will occur. Some theoretical studies on approximate algorithms to solve SSPP have appeared recently (Dean *et al.* (2005), Dean *et al.* (2008) and Escudero *et al.* (2011)). Using an exact approach to solve SSPP is still rare. It is worth noting that an $O(\sqrt{|K|})$-approximate greedy algorithm to solving the Stochastic Set Packing Problem has been presented in Dean *et al.* (2005). Here $|K|$ is the number of total ground elements.

In Escudero *et al.* (2011), the SSPP with a penalty associated with scenarios is decomposed into deterministic SPPs using a Lagrangian decomposition method. Computational results on the large-scale SSPPs show that the proposed heuristic algorithm provides very good lower bounds and upper bounds within reasonable CPU times. The proposed SSPP is only a one-stage problem with simple recourse and the proposed algorithm is an approximate method, not an exact algorithm for a global optimal solution.

In Dean *et al.* (2005) and Dean *et al.* (2008), a variant of the SSPP, the stochastic knapsack problem, was solved by tailored algorithms with and without dynamic choices of placing items in the knapsack. The items sizes are assumed to follow known, arbitrary distributions but the values of items are deterministic. The model can also solve stochastic machine scheduling problems with uncertain job durations. Although both approximate algorithms are polynomial-time, the gap of both best solutions are still large.

Shiina and Birge (2004) proposed a new algorithm for the stochastic unit commitment problem which is based on a column generation approach. The 0-1 variables represent the status of generating units. The unit commitment features are also restricted in their specified mathematical formulation. The data in the testing instances are only with two to four scenarios and 10 to 20 units. Hence, using column generation to solve the deterministic model may be more efficient than using some scenario decomposition method. The problem can be modeled using the same formulation as our SSPP after adjusting the unit commitment constraints, and can be solved by our proposed method. Unlike our algorithm, Shiina and Birge use the Dantzig-Wolfe reformulation and column generation method to decompose the whole model directly.

For stochastic integer problems with a large number of scenarios, a decomposition strategy has been shown to be one of the more effective tools. We have shown that the original Dual Decomposition method (Carøe and Schultz (1999)) could not solve even moderate size of Stochastic Uncapacitated Facility Location Problem (SUFLP). The Dynamic Dual Decomposition ($D^3$) method is proposed and implemented to solve various sizes of SUFLPs. This refined Dual Decomposition method provides a platform to combine other methods for various stochastic integer problems. This chapter aims to solve the Stochastic Set Packing Problem (SSPP) using an exact method to obtain a global optimal solution which combines the $D^3$ approach presented in the previous chapter and the developed column generation method.

The chapter is organized as follows. The SSPP formulation is introduced in Section 4.1.1. The proposed method is described in Section 4.2 and a small illustrative instance is demonstrated in Section 4.3.1. Section 4.3.2 reports our computational experiences on large-scale problems. Section 4.4 gives concluding remarks.

### 4.1.1 SSPP formulation

The Set Packing Problem (SPP) involves the selection of an underlying set of patterns such that the rewards from the selected patterns is maximized. The knowledge of conventional SPP is summarized in Balas and Padberg (1976) and Kwon *et al.* (2008). The stochastic version of SPP, SSPP, involves choosing determined patterns in the first stage and the second stage over all scenarios by maximizing the expected reward. To model the uncertainties in SSPP, we define the patterns in the second stage which are independently various over scenarios with probability distributions. The differences will happen in the presence of elements in each pattern, the rewards of each pattern and the total number of available patterns over scenarios in the second stage. The probability of scenarios are known in the first stage.

The mathematical expression of the SSPP model is:

Index and Set:

$K$ = the set of ground elements, $K = \{k \mid k = 1, 2, ...|K|\}$;

$N$ = the set of available patterns in the first stage, $N = \{i \mid i = 1, 2, ...|N|\}$;

$S$ = the set of scenarios, $S = \{s \mid s = 1, 2, ...|S|\}$;

$M^s$ = the set of available patterns in scenario $s$ in the second stage, $M^s = \{j^s \mid j^s = 1, 2, ...|M^s|\}$.

Parameter:

$c_i =$ the reward of pattern $i$ in the first stage;

$d^s_j =$ the reward of pattern $j$ in scenario $s$ in the second stage;

$a_{k,i} = 1$ if the element $k$ is packed in pattern $i$, 0 otherwise;

$A_i = [a_{1,i}\ a_{2,i}\ \cdots\ a_{|K|,i}]^T$ the column vector of pattern $i$;

$A = [A_1\ A_2\ \cdots\ A_{|N|}]$ the matrix with all patterns in the first stage, the size is $|K| \times |N|$;

$b^s_{k,j} = 1$ if the element $k$ is packed in pattern $j^s$, 0 otherwise;

$B^s_j = [b^s_{1,i}\ b^s_{2,i}\ \cdots\ b^s_{|K|,i}]^T$ the column vector of pattern $j$ in scenario $s$;

$B^s = [B^s_1\ B^s_2\ \cdots\ B^s_{|M^s|}]$ the matrix with all patterns in the second stage, the size is $|K| \times |M|$;

$p_s =$ the probability of the scenario $s$.

Variable:

$x_i = 1$ if pattern $i$ is selected, 0 otherwise;

$y^s_j = 1$ if pattern $j$ is selected in scenario $s$, 0 otherwise.

Without loss of generality, we set $|M^s|$ identical over all scenarios as the number $M$. Otherwise we may choose $M$ to be the largest number of $|M^s|$ over all scenarios and make up the number of the patterns to $M$ by adding dummy patterns which have zero reward.

The goal of SSPP is to obtain the maximal benefit over all scenarios by picking the patterns packing these elements. The Deterministic Equivalent Problem (DEP) of the two-stage SSPP is:

$$\max \quad \sum_{i\in N} c_i x_i + \sum_{s\in S} p^s \Big(\sum_{j\in M^s} d^s_j y^s_j\Big) \tag{4.1}$$

$$\text{s.t.} \quad \sum_{i\in N} a_{k,i} x_i + \sum_{j\in M^s} b^s_{k,j} y^s_j \le 1 \qquad \forall k \in K, \forall s \in S \tag{4.2}$$

$$x_i, y^s_j \quad \text{binary} \qquad \forall i \in N, \forall j \in M^s, \forall s \in S \tag{4.3}$$

In the above DEP, there are $|N|+|S|\times|M|$ binary variables and $|K|\times|S|$ constraints. Here all patterns in the first stage and in the second stage are explicitly given. Therefore, we can model the DEP in AMPL and solve it using CPLEX MIP solver.

## 4.2 The $D^3$+Column Generation Method

The $D^3$ method introduced in the previous chapter can be used to decompose the DEP of SSPP (4.1 - 4.3). A scenario subproblem is defined as:

$$\max \sum_{i=1}^{n} c_i x_i^s + \sum_{j=1}^{M} d_j^s y_j^s \tag{4.4}$$

$$\text{s.t.} \sum_{i=1}^{n} a_{k,i} x_i^s + \sum_{j=1}^{M} b_{k,j}^s y_j^s \le 1 \qquad \forall k \in K \tag{4.5}$$

$$x_i^s, y_j^s \ \text{ binary} \qquad \forall i \in N, \forall j \in M^s \tag{4.6}$$

In order to use the Column Generation method to solve the subproblem (4.4 - 4.6), we will start to select all first stage patterns to form the follow problem for scenario $s$:

$$\max \ \Delta_s = \sum_{i=1}^{n} c_i x_i^s \tag{4.7}$$

$$\text{s.t.} \ \sum_{i=1}^{n} x_i^s A_i \le \mathbf{1} \tag{4.8}$$

$$0 \le x_i^s \le 1 \qquad \forall i \in N \tag{4.9}$$

where the vector $\mathbf{1}$ is the column vector of all 1's. Then, we will iteratively add more patterns from the second stage by the Column Generation method for solving:

$$\max \ \Delta_s = \sum_{i=1}^{n} c_i x_i^s + \sum_{j \in S'} d_j^s y_j^s \tag{4.10}$$

$$\text{s.t.} \ \sum_{i=1}^{n} x_i^s A_i + \sum_{j \in S'} y_j^s B_j' \le \mathbf{1} \tag{4.11}$$

$$0 \le x_i^s \le 1, \ 0 \le y_j^s \le 1 \qquad \forall i \in N, \forall j \in S' \tag{4.12}$$

where $S' \subset S$. The original sub-problem (4.4 - 4.6) of each scenario will be solved as follows. We will use $\Delta_s^*$ and $\Delta_s'$ to indicate the optimal value of the current subproblem and the best value in the series of subproblems generated by Column Generation.

```
Step 1: Set $\Delta_s' = 0$.
Step 2: Solving LP (4.7 - 4.9) for the optimal value $\Delta_s^*$. Assume the reduced
cost of each constraint is $\pi_k$.
    If $\Delta_s^* > \Delta_s'$,
        $\Delta_s' = \Delta_s^*$. Go to Step 3.
    Else
        Go to Step 4
    End if
Step 3: For all unselected patterns in the second stage,
    If there exist any pattern(s) $B_j^s$ that $\sum_k \pi_k b_{k,j}^s \leq d_j^s$,
        Add pattern(s) $B_j^s$ into the LP (4.10 - 4.12).
        Go to Step 2
    Else
        Go to Step 4
    End if
Step 4: Solve LP (4.10 - 4.12) with the constraints that all variables are binary.
```

The solution obtained in Step 4 may not be an optimal solution of the original IP (4.4 - 4.6). In later computational experiments, we may observe that the aforementioned algorithm can obtain very good feasible solutions for small scenario sub-problems which are decomposed from large-size SSPPs. In order to obtain an optimal solution of the original IP (4.4 - 4.6), after Step 4 we have to repeat column generation at every node of the B&B tree to solve the IP, which defines a Branch-and-Price algorithm, as in Vance *et al.* (1993). For the sake of simplicity of the proposed $D^3$+CG method, we invoke the CPLEX IP solver directly to solve the LP in (4.10 - 4.12) with integrality constraints in Step 4 for a good feasible solution.

If the purpose is only solving the subproblem (4.4 - 4.6) in each scenario by the column generation, we do not need to select all patterns from the first stage. We may start from some part of patterns in order to obtain a smaller column matrix in case of a large number of patterns in the first stage. However, selecting all patterns in the first stage will be more efficient to implement the Branch-and-Bound strategy.

As we mentioned previously, if the patterns in the second stage are not explicitly given, the proposed method can still work.

We also think that Column Generation with deleting and adding patterns may be more efficient in Step 3:

```
Step 3: First for all patterns in the second stage in (4.8),
    if $\sum_k \pi_k b^s_{k,j} > d^s_j$, then delete the pattern from (4.8).
    For all unselected patterns in the second stage,
    if $\sum_k \pi_k b^s_{k,j} \leq d^s_j$, then select the pattern and add it into (4.8).
```

Deleting some patterns may normally be done once in a while when the matrix of patterns has grown very large. Using this strategy, we may keep a smaller constraint matrix of LP (4.8) and we can obtain the optimal value of LP (4.7 - 4.9). However, we notice that solving the LP with the smaller constraint matrix may miss some optimal solution(s) if the LP (4.7 - 4.9) without deleting patterns has more than one optimal solution.

**Definition**: Pattern $i$ *conflicts* with pattern $j$, if both patterns $i$ and $j$ select the same element. That means there exist $k$, $\theta_{k,i} = \theta_{k,j} = 1$, where $\theta_{k,i}$ and $\theta_{k,j}$ are an element in $\mathbf{A_i}$ and $\mathbf{B^s_j}$ respectively.

At the non-root nodes of B&B tree, we can pre-check the pattern conflict to fathom some node. If we branch on the node with $x_i \geq 1$ and $x_j \geq 1$, that means the solution of this node must include pattern $i$ and $j$. However, if pattern $i$ and $j$ conflict with each other, this node must violate the constraints. Using this way, we can pre-check the conflict patterns to fathom some nodes.

In Step 2, after we obtain the feasible solution of $\mathbf{x}$, we will plug in $\bar{\mathbf{x}}$ and solve the set of subproblems. In solution $\bar{\mathbf{x}}$, the component $\bar{x}_i$ with value 1 means the corresponding first stage pattern $i$ is picked in this solution. Then any pattern in second stage that conflicts with this pattern will never be able to enter the column generation LP (4.7 - 4.9). Therefore, using the pre-checking strategy, we may reduce the number of patterns in the second stage significantly in the column generation procedure.

## 4.3 Computational Experiments

### 4.3.1 A small instance

| | First Stage $A$ | | | |
|---|---|---|---|---|
| reward | 25.0 | 20.38 | 38.15 | 21.85 |
| item 1 | 0 | 1 | 0 | 0 |
| item 2 | 0 | 0 | 1 | 1 |
| item 3 | 1 | 1 | 1 | 0 |
| item 4 | 0 | 0 | 0 | 1 |
| item 5 | 1 | 0 | 1 | 0 |

Table 4.1: Patterns in the first stage of a small instance of SSPP

In order to illustrate the proposed method, we solve a small example in more detail. There are 5 items. In the first stage, there are 4 patterns. In the second stage, there are 4 scenarios with 6 patterns in each.

| | $s=1, p^1=0.25$ | | | | | | $s=2, p^2=0.25$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reward | 12.9 | 12.4 | 21.8 | 24.5 | 24.1 | 13.7 | 30.9 | 14.3 | 33.6 | 7.6 | 31.5 | 30.1 |
| item 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| item 2 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| item 3 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| item 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| item 5 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| | $s=3, p^3=0.25$ | | | | | | $s=4, p^4=0.25$ | | | | | |
| reward | 30.5 | 31.8 | 20.7 | 30.8 | 34.9 | 22.2 | 30.6 | 30.0 | 40.0 | 21.9 | 12.7 | 22.9 |
| item 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| item 2 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| item 3 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| item 4 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| item 5 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |

Table 4.2: Patterns in the second stage of a small instance of SSPP

We have the following vectors corresponding to the ones in (4.1 - 4.3). Please notice that we express the coefficient vectors, $\mathbf{c}, \mathbf{d}^1, \cdots$, and $\mathbf{d}^4$, as row vectors.

$\mathbf{c} = [25.0, 20.38, 38.15, 21.85]$,

$\mathbf{d}^1 = [12.9, 12.4, 21.8, 24.5, 24.1, 13.7], \cdots$ ,

$\mathbf{d}^4 = [30.6, 30.0, 40.0, 21.9, 12.7, 22.9]$.

$$\mathbf{A} = [a_{k,i}] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{B^1} = [b^1_{k,j}] = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}, \cdots, \mathbf{B^4} = [b^4_{k,j}] = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

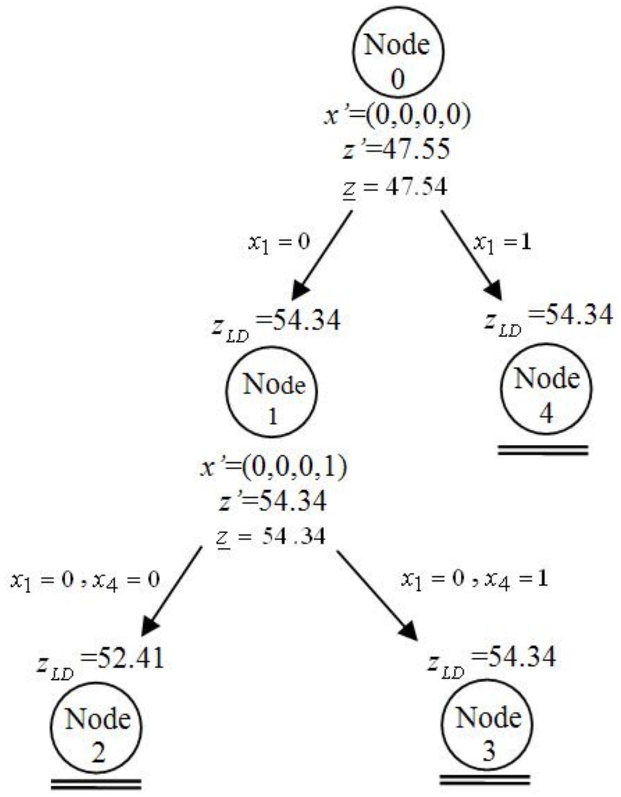

Figure 4.1: Branch&Bound tree on small SSPP

The DEP of the two-stage linear SSPP is

$$\begin{array}{rllllll} \max & \mathbf{cx} + p^1\mathbf{d^1y^1} + p^2\mathbf{d^2y^2} + p^3\mathbf{d^3y^3} + p^4\mathbf{d^4y^4} & & & & \\ \text{s.t.} & \mathbf{Ax} + & \mathbf{B^1y^1} & & & & \leq \mathbf{1} \\ & \mathbf{Ax} + & & \mathbf{B^2y^2} & & & \leq \mathbf{1} \\ & \mathbf{Ax} + & & & \mathbf{B^3y^3} & & \leq \mathbf{1} \\ & \mathbf{Ax} + & & & & \mathbf{B^4y^4} & \leq \mathbf{1} \end{array} \tag{4.13}$$

At the root node, we will solve the subproblem for each scenario s,

$$\begin{array}{rl} \max & \mathbf{cx^s} + \mathbf{d^sy^s} \\ \text{s.t.} & \mathbf{Ax^s} + \mathbf{B^sy^s} \leq \mathbf{1} \end{array}$$

In order to use the column generation method, first solve the restricted LP of

$$\begin{array}{rllllll} \max & \Delta = 25.0x_1 + 20.38x_2 + 38.15x_3 + 21.85x_4 & & & & \\ \text{s.t.} & & x_2 & & & \leq 1 \\ & & & x_3 + & x_4 & \leq 1 \\ & x_1 + & x_2 + & x_3 & & \leq 1 \\ & & & & x_4 & \leq 1 \\ & x_1 + & & x_3 & & \leq 1 \\ & 0 \leq x_1, x_2, x_3, x_4 \leq 1 & & & & \end{array} \tag{4.14}$$

The optimal solution is $x^* = [1, 0, 0, 1]$ with $\Delta^* = 46.85$. The dual values of the constraints are $\pi = [0, 21.85, 20.38, 0, 4.62]$. In scenario 1, the first pattern in the second stage is $b^1_{k,1} = [1\ 0\ 0\ 0\ 1]^T$. Then $\sum_k \pi_k b^1_{k,1} = 0 + 4.62 \leq d^1_1 = 12.92$. We add this pattern to the LP. All other patterns in scenario 1 could not be added into the LP. We solve the LP with one more pattern.

$$\begin{array}{llllllll}
\max & \Delta = 25.0x_1 + 20.38x_2 + 38.15x_3 + 21.85x_4 + 12.92y_1 & & & & & & (4.15)\\
\text{s.t.} & & x_2 + & & & y_1 & \leq 1 & \\
& & & x_3 + & x_4 & & \leq 1 & \\
& x_1 + & x_2 + & x_3 & & & \leq 1 & \\
& & & & x_4 & & \leq 1 & \\
& x_1 + & & x_3 + & & y_1 & \leq 1 & \\
& 0 \leq x_1, x_2, x_3, x_4, y_1 \leq 1 & & & & & &
\end{array}$$

Then the optimal solution is $[x\ y]^* = [0.5, 0.5, 0, 1, 0]$ with $\Delta^* = 51.00$. The dual values of constraints are $\pi = [4.15, 21.85, 16.23, 0, 8.77]$. When we calculate $\sum_k \pi_k b^1_{k,1}$ of the five remaining patterns, none could be added in the LP. We stop the column generation method. Now the optimal solution is not integer, we solve the last LP with integrality restrictions. The optimal integer solution of the first scenario subproblem is $x^* = [1, 0, 0, 1]$ with $Z_1^* = 46.85$. Similarly, we solve the subproblem of the other 3 scenarios by Column Generation: For scenario 2, $Z_2^* = 58.55$ with first pattern in the first stage and the third pattern in the second stage are picked. For scenario 3, $Z_3^* = 57.16$ with no pattern in the first stage and the fifth and sixth pattern in the second stage are picked. For scenario 4, $Z_4^* = 61.92$ with no pattern in the first stage and the third and fourth pattern in the second stage are picked. The scenario solution of the first stage variables are

Then we calculate the dispersion of the first stage variable: $\tilde{x} = \sum_{s=1}^{S} p^s x^s = [0.5, 0, 0, 0.25]$. Round the dispersion and get the feasible solution of $\bar{x} = [0, 0, 0, 0]$. The rounding rule is:

$x' = 1$, if $\tilde{x} > 0.5$; otherwise $x' = 0$.

Plug $\bar{x}$ in the restricted LP and use the column generation to solve the feasible solution

| | | | | |
|---|---|---|---|---|
| $x^{s=1}$ | 1 | 0 | 0 | 1 |
| $x^{s=2}$ | 1 | 0 | 0 | 0 |
| $x^{s=3}$ | 0 | 0 | 0 | 0 |
| $x^{s=4}$ | 0 | 0 | 0 | 0 |

of original problem with the objective value $z' = 47.5475$, which is saved as the current global lower bound $\underline{z} = 47.5475$.

Now we branch on the root node on the first stage variable $x_1$ which has the largest dispersion and generate two child nodes with the region $X^1 = \{x_1 = 0, 0 \leq x_2 \leq 1, 0 \leq x_3 \leq 1, 0 \leq x_4 \leq 1\}$ and $X^2 = \{x_1 = 1, 0 \leq x_2 \leq 1, 0 \leq x_3 \leq 1, 0 \leq x_4 \leq 1\}$ respectively.

At node 1 with $x_1 = 0$, we solve the Lagrangian relaxation of the subproblems in each scenario with the initial $\lambda = \vec{0}$ :

$$\begin{aligned} \max \quad & (p^s c + \lambda H^s)x^s + p^s d^s y^s \\ \text{s.t.} \quad & Ax^s + B^s y^s \leq 1 \\ & x^s \in X^1 \end{aligned} \tag{4.16}$$

Similarly, we use the column generation method to solve it. Then use the sub-gradient method to solve $z_{LD} = \min_\lambda\{\sum_s L_s(x^s, y^s, \lambda)\}$. After two iterations, the Lagrangian relaxation is $z_{LD} = 54.34$. Rounding the dispersion obtained in solving the Lagrangian relaxation, we get a value $x' = [0, 0, 0, 1]$. Plug in and get the feasible solution with value $z' = 54.34$. The global lower bound is updated as $\underline{z} = \max\{\underline{z}, z'\} = 54.34$.

Now we branch on $x_4$ with the largest dispersion and generate 2 child nodes of the current node with the region $X^{11} = \{x_1 = 0, 0 \leq x_2 \leq 1, 0 \leq x_3 \leq 1, x_4 = 0\}$ and $X^{12} = \{x_1 = 0, 0 \leq x_2 \leq 1, 0 \leq x_3 \leq 1, x_4 = 1\}$ respectively.

At node 2 with $x_1 = x_4 = 0$, we solve the Lagrangian relaxation. After 1 iteration, the Lagrangian relaxation $z_{LD} = 52.41 < \underline{z}$. We may fathom this node.

Now consider node 3 with the region $X^{12} = \{x_1 = 0, 0 \leq x_2 \leq 1, 0 \leq x_3 \leq 1, x_4 = 1\}$. Fixing $x_4 = 1$ means the fourth pattern has been picked in the subproblems. Then all patterns which conflict with this pattern will not be considered in the column generation method. For example, the fourth pattern includes item 2 and 4. In the second stage, pattern 2 in scenario 1 includes item 2 which cause a conflict. In this scenario, patterns 2, 3, 4 and 6 all conflict with this pattern. So using the conflict strategy, a bunch of patterns are eliminated in the column generation method in the subproblem.

We solve the Lagrangian relaxation at node 3. After 1 iteration, the Lagrangian relaxation $z_{LD} = 54.34 = \underline{z}$. We may fathom this node.

In B&B tree, there is a node with the region $X^2 = \{x_1 = 1, 0 \leq x_2 \leq 1, 0 \leq x_3 \leq 1, 0 \leq x_4 \leq 1\}$ left. We solve the Lagrangian relaxation. Similarly, we may eliminate the patterns in the second stage which conflicts with the first pattern ($x_1 = 1$). After 1 iteration, the Lagrangian relaxation $z_{LD} = 49.9473 < \underline{z}$. We may fathom this node.

There is no node left in B&B tree. After searching 5 nodes (including the root node), the optimal solution is $z^* = 54.34$ with the solution of the first stage variable as $x = [0, 0, 0, 1]$.

### 4.3.2 Computational Experiments on Large-scale Problems

To illustrate the efficiency of the proposed method to solve large SSPP models, we compare it with CPLEX 12.4 (CPLEX Optimization) to solve the DEP. In our experiments, we will show that comparing total CPU time, the proposed method beats CPLEX. The total CPU time of the proposed method includes searching the whole B&B tree to approximate a good feasible solution, since the proposed method is a heuristic. Time is the total CPU time in seconds until the optimal solution is obtained or the solver stops. All experiments are tested on a PC running Window XP with Pentium 4 CPU 3.2GHz and 1.00GB of RAM.

We generate data for the SSPP instances similar to Kwon *et al.* (2008) and Escudero *et al.* (2011). The rewards of the patterns in the first stage, $c_i, i = 1, \dots, n$, are generated in the range [12 , 18] with a uniform distribution. The rewards of the patterns in the second stage $d_j^s$ are generated in the range [10.0 , 15.0] with a uniform distribution. We set two thresholds, $t_1$ and $t_2$, for generating the patterns in the first and the second stage, respectively. If the random number is greater than the threshold, the entry in the pattern is one, otherwise it is zero. To reach the similar density of the constraint matrix in Kwon *et al.* (2008), we set $t_1 = 0.6$ and $t_2 = 0.55$ for the instances in Table 4.3 to follow the rule that practically the patterns in the first stage should have a higher reward and a sparser pattern than in the second stage. Also other threshold values are tested in a sensitivity analysis in Table 4.4 and Table 4.5.

In Table 4.3, the names of all examples have the same sense that the number of ground elements, patterns in the first stage, patterns in the second stage, and the number of scenarios are indicated as $K$, $n$, $m$ and $S$, respectively. For example, test #1 is a moderate size problem with 10 ground elements, 20 patterns in the first stage, 100 patterns in the second stage and 100 scenarios. V. and C. indicate the number of binary variables and constraints in each subproblem for the proposed method or indicate the number of variables and constraints in the DEPs which are modeled in AMPL and solved by CPLEX 12.4. Since we use column generation and the conflict eliminating strategy in the subproblems of the proposed method, the number of variables listed under "Subproblem" is the maximal number under the proposed method. In most cases, when we get the optimal solution of subproblem (4.7 - 4.9), the variables in the final LP are much less than V. which is $|N| + |M|$. The nonzero column (N.Z.) lists the number of nonzeros (all ones) in the constraints in DEP.

We also try to solve the DEP (4.1 - 4.3) by the column generation method directly. We give the explicit formula as follows to fit the column generation method before we

| | | | Subproblem | | DEP | | |
|---|---|---|---|---|---|---|---|
| | $K\_n\_m\_S$ | $t_1 - t_2$ | V. | C. | V. | C. | N.Z. |
| 1 | 10_20_100_100 | $0.6 - 0.55$ | 120 | 10 | 10,020 | 1,000 | 55,936 |
| 2 | 10_20_100_125 | $0.6 - 0.55$ | 120 | 10 | 12,520 | 1,250 | 66,072 |
| 3 | 10_20_100_160 | $0.6 - 0.55$ | 120 | 10 | 16,020 | 1,600 | 83,924 |
| 4 | 10_20_100_200 | $0.6 - 0.55$ | 120 | 10 | 20,020 | 2,000 | 105,633 |
| 5 | 10_20_150_100 | $0.6 - 0.55$ | 170 | 10 | 15,020 | 1,000 | 76,780 |
| 6 | 10_20_150_125 | $0.6 - 0.55$ | 170 | 10 | 18,770 | 1,250 | 94,364 |
| 7 | 20_20_50_100 | $0.6 - 0.55$ | 70 | 20 | 5,020 | 2,000 | 60,199 |
| 8 | 20_20_80_100 | $0.6 - 0.55$ | 100 | 20 | 8,020 | 2,000 | 87,670 |
| 9 | 20_10_100_100 | $0.6 - 0.55$ | 110 | 20 | 10,010 | 2,000 | 98,866 |
| 10 | 20_10_200_100 | $0.7 - 0.65$ | 210 | 20 | 20,010 | 2,000 | 145796 |
| 11 | 20_10_200_100 | $0.55 - 0.6$ | 210 | 20 | 20,010 | 2,000 | 168,660 |
| 12 | 20_20_200_200 | $0.6 - 0.55$ | 220 | 20 | 40,020 | 4,000 | 391,078 |
| 13 | 20_20_400_100 | $0.6 - 0.55$ | 420 | 20 | 40,020 | 2,000 | 375,678 |
| 14 | 30_30_300_300 | $0.6 - 0.55$ | 330 | 30 | 90,030 | 9,000 | 1,324,966 |

Table 4.3: The size of tested examples of two-stage SSPP

use it.

$$\begin{array}{rcccccc}
\text{Variables:} & x_1 \dots x_n & y_1^1 \dots y_m^1 & y_1^2 \dots y_m^2 & \cdots & y_1^S \dots y_m^S & \\
\text{reward:} & c_1 \dots c_n & p^1 b_1^1 \dots p^1 b_m^1 & p^2 b_1^2 \dots p^2 b_m^2 & \cdots & p^S b_1^S \dots p^S b_m^S & \\
\text{Patterns:} & \mathbf{A} & \mathbf{B^1} & \mathbf{0} & \cdots & \mathbf{0} & \leq \mathbf{1} \\
 & \mathbf{A} & \mathbf{0} & \mathbf{B^2} & \cdots & \mathbf{0} & \leq \mathbf{1} \\
 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & \mathbf{A} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{B^S} & \leq \mathbf{1}
\end{array} \tag{4.17}$$

There are $|K| \times |S|$ rows and $|N| + (|M| \times |S|)$ columns (variables are all binary). After using the conventional column generation method to eliminate some columns, the IP is still too large to be solved in moderate size models. In the small size models, using column generation to solve (4.17) was not faster than solving the original DEP modeled in AMPL and invoking the CPLEX IP solver.

In order to demonstrate the efficiency of the proposed method, we solve the instances by three methods: the proposed $D^3$ method with refined Column Generation method ("$D^3$+CG"), CPLEX MIP solver solving the whole DEP directly ("CPLEX MIP"), and solving the DEP directly by conventional Column Generation ("CG DEP") method. In Table 4.4, we list the CPU times and the solutions from these three methods. The "N." under the $D^3$+Column Generation is the total number of nodes in the B&B tree searched. The "T." columns list the total CPU time in seconds.

In some experiments, CPLEX IP solver failed to find the optimal solution and stopped with a feasible solution reporting "unrecoverable failure with integer solution." From the output, it was found that CPLEX created about a half million nodes in the B&B tree, most of which were still active since the gap between the relaxed bound and the feasible bound was very large. These cases are shown with the times marked *. These cases also happen when solving the IP after the DEP is simplified by the conventional column

| | | | $D^3$+CG | | | CPLEX MIP | | CG DEP | |
|---|---|---|---|---|---|---|---|---|---|
| | $K\_n\_m\_S$ | $t_1 - t_2$ | T. | S. | N. | T. | S. | T. | S. |
| 1 | 10_20_100_160 | 0.60-0.55 | 187.0 | 70.1780 | 33 | 3.1 | 70.1780 | 19.5 | 70.1780 |
| 2 | 10_20_100_200 | 0.60-0.55 | 254.7 | 63.7402 | 45 | 9.6 | 63.7402 | 29.3 | 63.7402 |
| 3 | 10_20_150_100 | 0.60-0.55 | 75.2 | 71.4770 | 35 | 4.1 | 71.4770 | 21.9 | 71.4770 |
| 4 | 10_20_150_200 | 0.60-0.55 | 80.3 | 75.4836 | 15 | 14.1 | 75.4836 | 18.5 | 75.4836 |
| 5 | 20_10_100_100 | 0.60-0.55 | 41.5 | 48.1652 | 19 | 32.6 | 48.1652 | 45.3 | 48.1652 |
| 6 | 20_10_200_100 | 0.60-0.55 | 128.7 | 42.0320 | 19 | 242.2 | 42.0320 | 233.4 | 42.0320 |
| 7 | 20_20_50_100 | 0.60-0.55 | 75.1 | 31.4499 | 25 | 11.5 | 31.4499 | 33.4 | 31.4499 |
| 8 | 20_20_80_100 | 0.60-0.55 | 141.6 | 32.3868 | 49 | 719.2 | 32.3868 | 3031.8 | 32.3868 |
| 9 | 20_20_100_100 | 0.60-0.55 | 122.3 | 34.4981 | 43 | 60.2 | 34.4981 | 78.5 | 34.4981 |
| 10 | 20_20_100_150 | 0.60-0.55 | 320.3 | 34.6262 | 31 | 433.0 | 34.6262 | # | # |
| 11 | 20_20_100_200 | 0.60-0.55 | 344.9 | 34.7332 | 45 | 12009.7* | 34.7332 | 8071.9 | 34.7332 |
| 12 | 20_20_120_100 | 0.60-0.55 | 125.6 | 36.2190 | 39 | 67.8 | 36.2190 | 93.4 | 36.2190 |
| 13 | 20_20_120_150 | 0.60-0.55 | 245.8 | 36.3170 | 43 | 1630.9 | 36.3170 | 7543.7 | 36.3170 |
| 14 | 20_20_120_200 | 0.60-0.55 | 152.0 | 42.3973 | 27 | 106.5 | 42.3973 | 164.2 | 42.3973 |
| 15 | 20_20_150_100 | 0.60-0.55 | 81.8 | 39.0146 | 25 | 336.9 | 39.0146 | 197.8 | 39.0146 |
| 16 | 20_20_150_150 | 0.60-0.55 | 238.9 | 38.7392 | 43 | 42180.3* | 38.7392 | 69046.1* | 38.7153 |
| 17 | 20_20_150_200 | 0.60-0.55 | 397.3 | 39.1593 | 31 | 29196.1* | 39.1593 | 4894.8* | 38.8394 |
| 18 | 20_20_200_100 | 0.60-0.55 | 150.5 | 42.2403 | 35 | 4677.8 | 42.2403 | 6074.6 | 42.2403 |
| 19 | 20_20_200_150 | 0.60-0.55 | 270.4 | 41.5834 | 31 | 54160.5* | 41.5834 | 31626.9* | 40.9746 |
| 20 | 20_20_200_200 | 0.60-0.55 | 318.9 | 41.8038 | 35 | 52203.6* | 41.2170 | 11505.6* | 39.8217 |
| 21 | 20_20_200_400 | 0.60-0.55 | 667.1 | 41.7817 | 31 | 66769.0* | 41.0833 | # | # |
| 22 | 20_20_400_100 | 0.60-0.55 | 148.8 | 49.2570 | 31 | 8813.5* | 49.2099 | 45824.6* | 46.3242 |
| 23 | 20_20_400_200 | 0.60-0.55 | 268.9 | 49.1585 | 25 | 78089* | 48.7788 | # | # |
| 24 | 20_20_400_400 | 0.60-0.55 | 306.2 | 53.0860 | 15 | 21101.9* | 53.0796 | # | # |
| 25 | 30_30_150_150 | 0.60-0.55 | 153.6 | 30.8000 | 23 | 5969.5 | 30.8000 | 12091.6* | 30.8000 |
| 26 | 30_30_200_200 | 0.60-0.55 | 142.1 | 31.7200 | 15 | 36611.4 | 31.7200 | # | # |
| 27 | 30_30_300_300 | 0.60-0.55 | 1287.7 | 30.4874 | 43 | 78277.8* | 29.7188 | # | # |
| 28 | 30_30_400_400 | 0.60-0.55 | 6156.8 | 31.7620 | 43 | 10589.7* | 29.8069 | # | # |

* Unrecoverable failure with integer solution

# Fail to populate problem data arrays due to the memory limit

Table 4.4: Computational results of two-stage SSPP by three methods

generation method. In the large-size test 30_30_300_300, when using AMPL to build the model (4.17), there are 90,030 binary variables, 9,000 constraints and 1,324,966 nonzero items in the constraint matrix. After the pre-processing by AMPL/CPLEX to eliminate redundant variables or constraints, there are still 89,961 columns, 9,000 constraints and 1,323,731 nonzero items in the constraint matrix. Since AMPL could not efficiently reduce the large-size IP, CPLEX eventually collapsed with a 20.01% gap after a long time (78,277.8 seconds).

| 20_10_200_100 | | | | $D^3$+CG | | | CG DEP | | CPLEX MIP | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $t_1 - t_2$ | N.Z. | D. | T. | S. | N. | T. | S. | T. | S. |
| 1 | 0.80 – 0.75 | 103,888 | 24.74 % | 128.5 | 133.3596 | 35 | 32.7 | 133.3596 | 29.0 | 133.3596 |
| 2 | 0.70 – 0.65 | 145,796 | 34.71 % | 90.4 | 69.6405 | 17 | 9351.7* | 66.5067 | 13294.9 | 69.6405 |
| 3 | 0.65 – 0.60 | 167,260 | 39.82 % | 88.0 | 53.0357 | 23 | 33480.5* | 52.2343 | 21515.5 | 53.0357 |
| 4 | 0.60 – 0.55 | 188,163 | 44.80 % | 128.7 | 42.0320 | 19 | 233.4 | 42.0320 | 242.2 | 42.0320 |
| 5 | 0.55 – 0.50 | 208,835 | 49.72 % | 75.4 | 33.0511 | 19 | 127.0 | 33.0511 | 143.0 | 33.0511 |
| 6 | 0.50 – 0.45 | 230,096 | 54.78 % | 57.7 | 28.1652 | 23 | 34.9 | 28.1652 | 17.2 | 28.1652 |
| 7 | 0.45 – 0.40 | 250,963 | 59.75 % | 55.9 | 24.1922 | 23 | 40.3 | 24.1922 | 316.1 | 24.1922 |
| 8 | 0.40 – 0.35 | 272,395 | 64.86 % | 69.8 | 16.9738 | 25 | 41.9 | 16.9738 | 28.0 | 16.9738 |
| 9 | 0.35 – 0.30 | 293,999 | 70.00 % | 23.1 | 17.5736 | 11 | 373.1 | 17.5736 | 801.3 | 17.5736 |
| 10 | 0.30 – 0.25 | 314,730 | 74.94 % | 7.3 | 16.8621 | 3 | 1229.7 | 16.8621 | 1567.3 | 16.8621 |
| 11 | 0.25 – 0.20 | 334,971 | 79.75 % | 6.6 | 17.3368 | 1 | 2158.8 | 17.3368 | 3397.4 | 17.3368 |
| 12 | 0.20 – 0.15 | 355,590 | 84.66 % | 7.3 | 17.9124 | 3 | 1277.1 | 17.9124 | 963.8 | 17.9124 |

Table 4.5: Sensitivity experiments over various density of patterns for same model with size 20_10_200_100

We also implement the conventional Column Generation method to solve the DEP directly by programming in VC++. First the program populates the data of a restricted

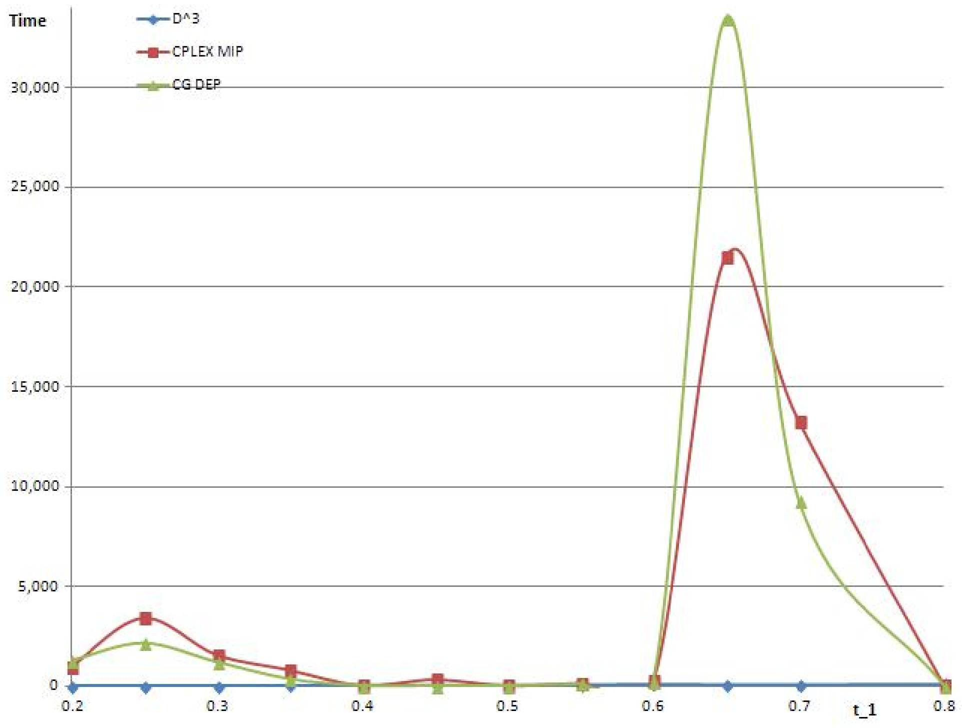

Figure 4.2: The chart of CPU time corresponding $t_1$ for the model 20_10_200_100

LP and solves it, then using the column generation method will add more columns to LP until no column can be added. As more columns are added, the size of LP becomes larger. Without any efficient pre-processing on reducing redundant constraints or variables like AMPL does, the number of constraints and variables explodes very quickly. Then the Visual C++ compiler reports error that "total image size exceeds max image and may not run" and stops running, when the program populates the LP data and invokes CPLEX to solve it in large size of models (e.g., #10, #21, #23, #24 and #26 – #28). In those cases, VC++ cannot even populate the data to process conventional CG method (denoted by a "#" sign in the time ("T.") column under the header "CG DEP"). When comparing the CPU times of the $D^3$+CG method to CPLEX solver, one may note that AMPL has conducted various advanced techniques to pre-process data and pre-solve the problem before CPLEX solves the whole model. However, our proposed method does not have the advantage of pre-processing data or pre-solve the problem.

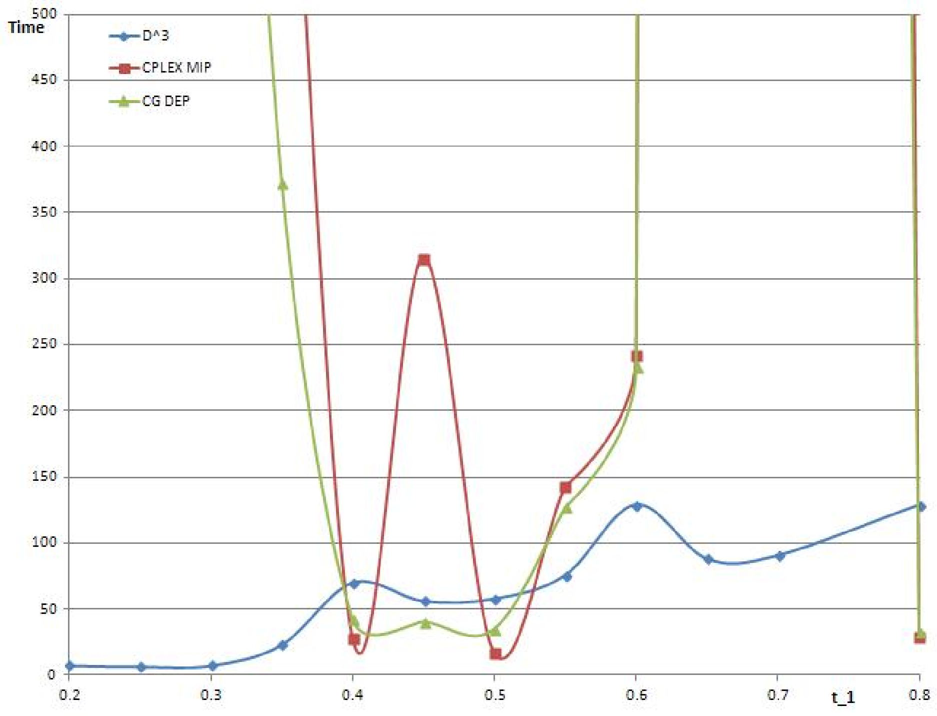

Figure 4.3: The zoom in chart of CPU time for the model 20_10_200_100

We find that in the experiments with the same size, the CPU times used by CPLEX to solve the DEP directly will vary significantly with varying densities of patterns. We choose test #6 (20_10_200_100) in the sensitivity experiments. We process more tests with various thresholds on test #6, which has a moderate size. In test #6, we have 20 nodes, 10 patterns in the first stage, 200 patterns in the second stage and 100 scenarios. In Table 4.5, we change $t_1$ from 0.8 to 0.2 and set $t_2$ 0.05 less than $t_1$. The numbers of non-zero items in the constraint matrix and the percentages of densities, which are the percentage of element ones (non-zero items) in the constraint matrix in (4.17) of DEP, are listed in the columns with headers "N.Z." and "D.", respectively. The CPU times of solving DEP by conventional CG method and CPLEX MIP solver vary significantly. The proposed method has much more stable solution times. The proposed method can solve the small-size model (20_10_200_100) of different densities within short CPU times from 6.6 seconds to 128.7 seconds. However CPLEX solves the same size model from 28 seconds to 21,515 seconds. The conventional CG also spends from 32 seconds to 33,480 seconds before the program of the conventional CG collapses. Figure 4.2 shows the CPU times corresponding to various $t_1$ on test #1 – #12 in Table 4.5. Figure 4.3 shows a magnified view of Figure 4.2.

We choose another larger size model to do more sensitivity analyses of various densities, shown in Table 4.6. Since there are some failures which are marked with "*" on times under the header "T." in both "CPLEX MIP" and "CG DEP," we also list the gaps between the best feasible solutions and the optimal solutions under the headers "Gap" for both methods. Without pre-eliminating redundancy and efficient pre-solve processing, the conventional CG, which is implemented in VC++, collapses on most tests with large gaps from 12.4% to 47.4%. CPLEX MIP solver can either solve to optimality in some tests or stop with small gaps less than 4% except for the test #8 ($t_1$=0.40 and $t_2$=0.35) where the CPLEX solver stopped after 10 hours with a gap still more than 20%.

Since the $D^3$+CG decomposes the SSPP into small sized subproblem (4.4 - 4.6) for each scenario before solving it by developed column generation method, the various densities will not affect CPU times of solving each individual small sized subproblem of each scenario. But it seems the density considerably affects the CPU times on solving the whole model with various densities which are large-scale NP-hard IPs.

One may notice that the proposed method, the "$D^3$+refined CG" method, can efficiently solve the cases (tests #10 – #12 in both Table 4.5 and Table 4.6) with denser patterns in the constraint matrix by the refined CG method to pre-check the conflicting patterns in each iteration of non-root nodes, since increased density causes more patterns to conflict to each others.

| 20_20_200_200 | | | | $D^3$+CG | | | CG DEP | | CPLEX MIP | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $t_1 - t_2$ | N.Z. | D. | T. | S. | N. | T. | Gap | T. | Gap |
| 1 | 0.80-0.75 | 215,110 | 24.44% | 619.7 | 139.8457 | 73 | 576.7 | - | 116.1 | - |
| 2 | 0.70-0.65 | 302,560 | 34.38% | 731.9 | 68.7188 | 47 | 796.1 | - | 5540.7* | 3.46% |
| 3 | 0.65-0.60 | 347,132 | 39.45% | 302.0 | 52.7951 | 31 | 9942.1* | 12.36% | 73122.6* | - |
| 4 | 0.60-0.55 | 391,078 | 44.44% | 318.9 | 41.8038 | 35 | 11505.63* | 17.50% | 52203.6* | 1.73% |
| 5 | 0.55-0.50 | 437,334 | 49.70% | 277.8 | 32.8129 | 35 | 12095.2* | 24.99% | 109251.0* | 1.26% |
| 6 | 0.50-0.45 | 481,103 | 54.67% | 199.4 | 28.4392 | 31 | 1923.2* | 24.74% | 27,008.5 | - |
| 7 | 0.45-0.40 | 524,965 | 59.66% | 237.8 | 23.9990 | 35 | 2059.4* | 29.95% | 4,043.0 | - |
| 8 | 0.40-0.35 | 569,108 | 64.67% | 280.6 | 17.9265 | 41 | 2050.9* | 47.42% | 36498.9* | 21.22% |
| 9 | 0.35-0.30 | 612,476 | 69.61% | 54.4 | 17.7697 | 11 | 1644.9* | 39.11% | 60948.7* | 1.99% |
| 10 | 0.30-0.25 | 656,616 | 74.62% | 11.9 | 16.8187 | 3 | 1399.4* | 31.69% | 44959.4* | 1.67% |
| 11 | 0.25-0.20 | 700,632 | 79.62% | 11.1 | 16.7674 | 1 | 1647.7* | 21.89% | 59293.9* | 2.37% |
| 12 | 0.2-0.15 | 744,365 | 84.59% | 13.6 | 17.8558 | 1 | 1349.5* | 17.09% | 37115.6* | 1.52% |

* Unrecoverable failure with integer solution

Table 4.6: Sensitivity experiments over various density of patterns for same model with size 20_20_200_200

## 4.4 Conclusions

We combined the $D^3$ and a refined Column Generation approach for solving SSPP. The proposed method could solve moderate and large size instances of SSPP whose DEP could not be solved by the CPLEX IP solver. The sensitivity study on various densities of the patterns shows that the proposal method is a more robust method than solving the DEP by IP solver or by implementing a Column Generation method directly. That is, the CPU times of solving SSPPs directly with the CPLEX solver are very sensitive to the density, but using the proposed method, the solution times are more stable.

# Chapter 5

# $D^3$+Benders Decomposition on Stochastic Multi-plant Facility Location Problems

## 5.1 Introduction to Stochastic Multi-plant Facility Location Problems

The facility location problem is fundamental in logistics and supply chain management. The Uncapacitated Facility Location Problem (UFLP) and sole-sourcing (Silva (2004)) or $p$-median (Ntaimo and Sen (2008)) Capacitated Facility Location Problem (CFLP) have been well-studied in recent decades. The location decision variables of both problems are binary. The uncapacitated facility location problem can be extended to the Multi-plant Facility Location Problems (MpFLP) (recent studies are Ghiani *et al.* (2002a), Ghiani *et al.* (2002b) and Wu *et al.* (2006)).

The MpFLP is one kind of CFLP where several plants can be built in each location. Therefore the facility location decision variables are general integer (as opposed to binary) variables. The MpFLP can be applied in the manufacturing sector where several groups of identical machines can be located in cells in a manufacturing plant. Also, in telecommunication network design (Andrade *et al.* (2005) and Ntaimo (2004)), identical switches can be stacked in each network hub or node in order to transmit data. Due to the uncertainty of customers' demands for high speed service, telecommunications providers often adopt a very conservative approach to capital investment, leading to potential losses in revenue with poor service.

In Ghiani *et al.* (2002a) and Ghiani *et al.* (2002b), the authors address the capacitated facility location problem in which several facilities (plants) can be opened in the same site and describe a tailored Lagrangian heuristic to solve the decomposed subproblems. Their computational results indicate that the proposed algorithm provides high quality lower and upper bounds. Several applications were modeled and a case study related to a polling station location problem in an Italian municipality is reported. However, in their model, binary variables represent the location decision for each polling station even though multiple identical polling stations will be set in one site. The polling stations problem in their study can be modeled and solved as our deterministic multi-plant model with integer location decision variables since the setup cost and the capacity of each polling station in a polling site are identical.

In Wu *et al.* (2006), the capacitated facility location problem with specific setup costs is modeled and then is solved by a Lagrangian Heuristic Algorithm (LHA). The setup costs, which are nonlinear functions of the number of plants, include two parts: a setup cost of opening a site and setup costs of building multiple plants in an open site. Correspondingly, the binary variables will decide to open a site or not and the integer variables will decide how many plants will be built in each open site. The model, with both binary and integer variables as the location decision variables, is similar to the deterministic model of MpFLP here, except that the setup costs of multiple plants in each location are linear to the number of plants and there are no binary variables of site decision in our model. In the computational tests, they also provide two equivalent, alternative MIP formulations by changing integer variables to be binary. The LHA is shown to be efficient in solving original models compared with CPLEX MIP solver on solving two alternative MIP models.

An industrial application in telecommunication networks, which has similar structure to the stochastic version of Multi-plant Facility Location Problem (SMpFLP), was studied in Andrade *et al.* (2005). The computational results using three stochastic algorithms for solving two stage Stochastic Linear Integer Problems (SLIP) are reported: an integer L-shaped (Benders) decomposition method, a branch-and-bound framework and a disjunctive cutting plane method. There are only integer variables in the first stage and only continuous variables in the second stage.

In this chapter, we will solve the stochastic version of the MpFLP. First, we will apply the $D^3$ method to decompose the relaxed SMpFLPs, then we will use the Benders Decomposition Method (BDM) to solve the scenario subproblems of the decomposed SMpFLP. However, our tests in Table 5.1 and Table 5.2 show that the conventional BDM cannot solve the subproblems efficiently. In Magnanti and Wong (1981), they proposed an algorithm to improve the performance of BDM when solving Uncapacitated Facility Location Problems (UFLP). When we studied their algorithm, we find that their method cannot be directly implemented on the multi-plant problem due to the different structures of the multi-plant and single-plant problems. In this chapter, we extend their method when solving multi-plant subproblems with a refined BDM. We provide a theoretical study on Pareto-cuts and the termination of Benders decomposition in our algorithm to solve SMpFLP.

Wentges (1996) presented modified Benders cuts for the Capacitated Facility Location Problem (CFLP) and showed their efficiency in computational tests. But only CFLPs with a simple plant were analyzed and solved.

An aggregation method and error bounds in stochastic linear programming (SLP) were analyzed in Birge (1985). By aggregating scenarios, the complexity of the original SLP can be reduced significantly and the optimal solution of the SLP can be bounded using the solution of the aggregated problem. We have decomposed the SMpFLP into a group of small linear subproblems which can be aggregated and solved by the scenario aggregation method. We conduct a theoretical analysis on the error bounds based on the specific structure of the decomposed multi-plant subproblems in SMpFLP.

Two different methods of selecting the sample scenarios are proposed and tested in Jonsson and Silver (1996). However, both methods of selecting scenarios, random sampling and descriptive sampling, are generic – neither is dependent on the structure of the model. Instead, we will aggregate the original large-scale model by selecting scenarios with values of demands and serving costs which are within the pre-determined percentages of discrepancies with original data.

The chapter is organized as follows. The formulations of the deterministic and stochastic MpFLP are introduced in Section 5.2. In Section 5.3 we review how to apply the Conventional Benders Decomposition Method (CBDM) directly to a simple version of SMpFLP where only integer variables are in the first stage and continuous variables are in the second stage. Our $D^3$+Pareto-Benders Decomposition Method ($D^3$+PBDM) is proposed in Section 5.4. In order to create effective cuts for Benders method, the concept of a Pareto-optimal cut is introduced and refined in Section 5.4.2. An aggregation method is developed to aggregate and solve the large-scale SMpFLPs in Section 5.5. In Section 5.6, we report our computational experiments on both randomly generated data and benchmark data. Finally we give concluding remarks in Section 5.7.

## 5.2 The Formulation of Stochastic Multi-plant Facility Location Problems

For the rest of the chapter, let $[x]^+ = \max\{x, 0\}$. For the optimization problem, $\min\{z = f(\mathbf{x}) \mid \mathbf{x} \in X\}$, let $z^*$ and $\mathbf{x}^*$ indicate the optimal value and the optimal solution, respectively.

In order to describe the developed Benders Decomposition Method to solve subproblems, first we formulate a type of deterministic MpFLP in a manner similar to the single plant CFLP described in Magnanti and Wong (1981) and further studied in Wentges (1996).

Let

$N$ = the set of potential facility locations, $N = \{i \mid i = 1, 2, ...|N|\}$;

$D$ = the set of clients, $D = \{j \mid j = 1, 2, ...|D|\}$;

$f_i$ = the setup cost for locating a plant at location $i$;

$c_{i,j}$ = the cost of each unit of product being served to client $j$ from location $i$;

$d_j$ = the demand from client $j$;

$m_i$ = the maximum number of plants able to be built at location $i$;

$q_i$ = the capacity of each plant at location $i$;

$x_i$ = the number of plants to be built at location $i$;

$y_{i,j}$ = the portion of client $j$'s demand being served from facility $i$.

$$\min \sum_{i\in N} f_i x_i + \sum_{i\in N, j\in D} c_{i,j} d_j y_{i,j} \tag{5.1}$$

$$\text{s.t. } x_i \leq m_i \qquad \forall i \in N \tag{5.2}$$

$$\sum_{i\in N} y_{i,j} = 1 \qquad \forall j \in D \tag{5.3}$$

$$\sum_{j\in D} d_j y_{i,j} \leq q_i x_i \qquad \forall i \in N \tag{5.4}$$

$$x_i \geq 0 \text{ integer}, y_{i,j} \geq 0 \quad \forall i \in N, \forall j \in D \tag{5.5}$$

The objective function (5.1) is to minimize the total costs including the setup costs of building plants and serving costs of products from open plants to clients. Constraint (5.2) enforces that the number of plants being built cannot be larger than allowed at each location. Constraint (5.3) ensures that the demand will be satisfied for each client. Constraint (5.4) ensures that the products will only be served at location $i$ with open plants and the amount of products being shipped out to all clients are limited by the capacity of the plants at location $i$. We name Constraint (5.4) as *"aggre-shipping"* conditions, since it requires that all products being shipped out are limited by the capacity of multiplants at each location.

In the literature of the Capacitated Facility Location Problem (CFLP) where the facility variables are binary, some models of CFLP also introduced and included with the following constraints:

$$y_{i,j} \leq x_i \qquad \forall i \in N, \forall j \in D \tag{5.6}$$

We name Constraints (5.6) as *"pathway"* constraints, since the products being shipped out to a client is limited by the capacity of a pathway from location $i$ to client $j$. The condition $d_j y_{i,j} \leq q_i x_i$ can be derived from Constraint (5.6), if $\frac{d_j}{q_i} \leq 1$ which is usually the case in CFLP.

In the model of CFLP with binary facility variables, the formulation with only aggre-shipping constraints (5.4) is called the *weak* formulation and the formulation with both pathway constraints (5.6) and aggre-shipping constraints (5.4) is known as the *strong* formulation (see Wentges (1996)).

In Wentges (1996), as it is mentioned that without loss of generality, if $d_j \leq q_i \; \forall \; i, j$, sometimes the CFLP with binary facility variables is formulated without constraints (5.6) since constraints (5.6) result from (5.4) and (5.5). Therefore, the CFLP in Wentges (1996) with both pathway constraints (5.6) and aggre-shipping constraints (5.4) were chosen to create strong Benders cuts.

The formulation with only aggre-shipping conditions (5.4) will yield inferior Benders cut while solving the Benders subproblem, whereas the strong formulation has been shown to be very helpful for obtaining strong Benders cut. Pathway constraints (5.6) are also considered in the Mixed Plant Location Problem (MPLP) with binary variables in Guignard and Spielberg (1979). In the model of MPLP, some existing plants have capacity restrictions, but others, which are usually only potential plants to be explored for possible construction, have unlimited capacity. In the model, the capacity on the path from location $i$ to client $j$ is also given as $m_{i,j}$. Then a group of "disaggregate" constraints, e.g., (5.6), are included in the model. With the additional constraints over the integer variables, the proposed direct dual method provide a strong relaxed form of the problem.

Due to the observation in the MpFLP ((5.1) - (5.5)) that the total capacity of all plants must be able to satisfy all demands in feasible solutions, we may add the *"total-capacity"* condition of the multi-plant decision variables in MpFLP:

$$\sum_{i \in N} q_i x_i \geq \sum_{j \in D} d_j \tag{5.7}$$

The total-capacity condition can also be induced from the "aggre-shipping" conditions (5.4) and the constraints (5.3).

We use total-capacity constraints (5.7) to relax aggre-shipping constraints (5.4) in the aforementioned MpFLP as:

$$\min \sum_{i \in N} f_i x_i + \sum_{i \in N, j \in D} c_{i,j} d_j y_{i,j} \tag{5.8}$$

$$\text{s.t.} \sum_{i \in N} q_i x_i \geq \sum_{j \in D} d_j \tag{5.9}$$

$$x_i \leq m_i \qquad \forall i \in N \tag{5.10}$$

$$\sum_{i \in N} y_{i,j} = 1 \qquad \forall j \in D \tag{5.11}$$

$$d_j y_{i,j} \leq q_i x_i \qquad \forall i \in N, \forall j \in D \tag{5.12}$$

$$x_i \geq 0 \text{ integer}, y_{i,j} \geq 0 \quad \forall i \in N, \forall j \in D \tag{5.13}$$

We formulate a two-stage stochastic MpFLP, named as SMpFLP, as follows:

$$\min \sum_{i \in N} f_i x_i + \sum_{s \in S} p^s \left( \sum_{i \in N} h_i^s u_i^s + \sum_{i \in N, j \in D} c_{i,j}^s d_j^s y_{i,j}^s \right) \tag{5.14}$$

$$\text{s.t. } x_i + u_i^s \leq m_i \qquad \forall i \in N, \forall s \in S \tag{5.15}$$

$$\sum_{i \in N} y_{i,j}^s = 1 \qquad \forall j \in D, \forall s \in S \tag{5.16}$$

$$\sum_{j \in D} d_j^s y_{i,j}^s \leq q_i x_i + t_i^s u_i^s \qquad \forall i \in N, \forall s \in S \tag{5.17}$$

$$x_i, u_i^s \geq 0 \text{ integer} \qquad \forall i \in N, \forall s \in S \tag{5.18}$$

$$y_{i,j}^s \geq 0 \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{5.19}$$

In this two-stage SMpFLP model, there are $|S|$ scenarios in the second stage. There are a total of $|N||S| + |D||S| + |N||S|$ constraints and $|N| + |N||S| + |N||D||S|$ variables consisting of $|N|$ integer variables in the first stage, and $|N||S|$ integer variables and $|N||D||S|$ continuous variables in the second stage. The parameters $h_i^s$ and $t_i^s$ are the setup cost for building a plant and the capacity of each open plant, respectively, at location $i$ in scenario $s$ in the second stage. The parameter $p^s$ is the probability of scenario $s$. The variables $u_i^s$ and $y_{i,j}^s$ in the second stage are the number of plants built at location $i$ and the portion of client $j$'s demand being served from location $i$ in scenario $s$, respectively.

Objective function (5.14) minimizes the total costs including the setup costs of building plants in the first stage and the expected value under all scenarios of the recourse costs, which involve the setup costs of building more plants and costs of servicing clients in the second stage. Constraint (5.15) enforces that total number of plants built at location $i$ will not exceed the allowed upper bound. Constraint (5.16) enforces that the demand is satisfied for each client under each scenario in the second stage. Constraint (5.17) ensures that all products being served from facility $i$ will not exceed the capacity of plants built in either the first stage or the second stage. In this model, the recourse matrix includes items $t_i^s$ and $d_j^s$, which are both dependent on scenario $s$ and the parameters of the second stage variables are in the objective function.

Based on the aforementioned relaxed MpFLP, we also introduce the relaxed SMpFLP as follows and solve it instead of the original SMpFLP ((5.14) - (5.19)).

$$\min \sum_{i\in N} f_i x_i + \sum_{s\in S} p^s \left( \sum_{i\in N} h_i^s u_i^s + \sum_{i\in N, j\in D} c_{i,j}^s d_j^s y_{i,j}^s \right) \tag{5.20}$$

$$\text{s.t.} \sum_{i\in N} (q_i x_i + t_i^s u_i^s) \geq \sum_{j\in D} d_j^s \qquad \forall s \in S \tag{5.21}$$

$$x_i + u_i^s \leq m_i \qquad \forall i \in N, \forall s \in S \tag{5.22}$$

$$\sum_{i\in N} y_{i,j}^s = 1 \qquad \forall j \in D, \forall s \in S \tag{5.23}$$

$$d_j^s y_{i,j}^s \leq q_i x_i + t_i^s u_i^s \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{5.24}$$

$$x_i, u_i^s \geq 0 \text{ integer} \qquad \forall i \in N, \forall s \in S \tag{5.25}$$

$$y_{i,j}^s \geq 0 \qquad \forall i \in N, \forall j \in D, \forall s \in S \tag{5.26}$$

The Stochastic Uncapacitated Facility Location Problems (SUFLP) in Chapter 3 is a special case of SMpFLP where $m_i, d_j^s, q_i$ and $t_i^s$ are all equal to 1. Obviously, SMpFLP is an NP-hard problem since the SMpFLP and SUFLP are extended cases with uncertainty from the classical NP-hard problem UFLP.

## 5.3 Benders Decomposition Method

Benders Decomposition Method (BDM), developed by Benders (1962), was originally developed to solve linear mixed integer programming problems. BDM divides the linear mixed integer program into two problems: the Benders subproblem with only continuous variables, and the Benders master problem with the integer variables and an artificial continuous variable. Then BDM will iteratively solve a main problem and a subproblem and will eventually lead to an optimal solution of the original MIP.

If there are no facility decision variables $u_i^s$ in the second stage, we may directly use the BDM to solve two-stage stochastic problems by separating the Benders subproblems with second stage variables and the Benders master problem with first stage variables. In this case, the Benders Decomposition Subproblem (BDS) for each scenario given the fixed first stage variables $\bar{x}^k$ in iteration $k$ is

$$\min_{\mathbf{y}} \ \beta_s^k(\bar{\mathbf{x}}^k) = \sum_{i \in N, j \in D} c_{i,j}^s d_j^s y_{i,j}^s + \sum_{i \in N} f_i \bar{x}_i^k \tag{5.27}$$

$$\text{s.t.} \ \sum_{i \in N} y_{i,j}^s = 1 \qquad \forall j \in D \tag{5.28}$$

$$\sum_{j \in D} d_j^s y_{i,j}^s \leq q_i \bar{x}_i^k \qquad \forall i \in N \tag{5.29}$$

$$y_{i,j}^s \geq 0 \qquad \forall i \in N, \forall j \in D \tag{5.30}$$

The subproblem contains only continuous variables $y_{i,j}^s$ and it can be decomposed and solved independently for each scenario. The Benders Decomposition Main problem given the fixed second stage variables is

$$\min_{\mathbf{x}} \ \alpha^k(\bar{\mathbf{y}}^k) = z_s^k \tag{5.31}$$

$$\text{s.t. } z_s^k \geq \sum_{i \in N} f_i x_i^k + \sum_{i \in N, j \in D} c_{i,j}^s d_j^s \bar{y}_{i,j}^s \qquad \forall s \in S \tag{5.32}$$

$$x_i \leq m_i \qquad \forall i \in N \tag{5.33}$$

$$x_i \geq 0 \text{ integer} \qquad \forall i \in N \tag{5.34}$$

However, this method will not perform very well in solving the SMpFLP (5.14–5.19) since the discrete decision variables $u_i^s$ that exist in the second stage in (5.14–5.19) will make the decomposed subproblem as follows which is a mixed-integer problem.

$$\min_{\mathbf{y}} \ \beta_s^k(\bar{\mathbf{x}}^k) = \sum_{i \in N} h_i^s u_i^s + \sum_{i \in N, j \in D} c_{i,j}^s d_j^s y_{i,j}^s + \sum_{i \in N} f_i \bar{x}_i^k \tag{5.35}$$

$$\text{s.t. } u_i^s \leq m_i - \bar{x}_i^k \qquad \forall i \in N \tag{5.36}$$

$$\sum_{i \in N} y_{i,j}^s = 1 \qquad \forall j \in D \tag{5.37}$$

$$t_i^s u_i^s - \sum_{j \in D} d_j^s y_{i,j}^s \geq -q_i \bar{x}_i^k \qquad \forall i \in N \tag{5.38}$$

$$y_{i,j}^s \geq 0 \qquad \forall i \in N, \forall j \in D \tag{5.39}$$

$$u_i \geq 0 \text{ integer} \qquad \forall i \in N \tag{5.40}$$

## 5.4 $D^3$+Pareto-Benders Decomposition Method

### 5.4.1 Solving the Benders Client-Subproblems

In this section, we first use the $D^3$ method to decompose the relaxed SMpFLP ((5.20) - (5.26)) into subproblems for each scenario. Then we use the BDM to decompose the scenario subproblems into the main subproblem including all integer location decision variables and the dual subproblem with continuous serving variables. Since we relax aggre-shipping constraints (5.17), we can even further decompose the dual subproblem for each client and use a new heuristic method to solve the client subproblem. After we obtain a solution of each scenario subproblem with pathway constraints, we need to solve an LP if the solution of scenario subproblem does not satisfy the aggre-shipping conditions (5.17) in order to approximate a solution of original SMpFLP ((5.14) - (5.19).

In order to improve the performance of BDM, we conduct a further study on Pareto-cuts and use Pareto-cuts to solve the main subproblem efficiently in BDM. Following the aforementioned routine of the proposed $D^3$+PBDM, we solve the SMpFLP efficiently. We demonstrate the flow chart of the proposed method in Figure 5.1.

We have developed the $D^3$ method to decompose and efficiently solve stochastic MIPs. The details on the $D^3$ method are in Chapter 3. After using the $D^3$ method to decompose the relaxed SMpFLP (5.20–5.26), we get the following subproblem for each scenario $s$ which we call the Scenario-decomposed Subproblem (SS):

$$\text{(SS)}\ \min \sum_{i\in N} f_i'' x_i^s + \sum_{i\in N} h_i^s u_i^s + \sum_{i\in N, j\in D} c_{i,j}^s d_j^s y_{i,j}^s \tag{5.41}$$

$$\text{s.t.} \sum_{i\in N}(q_i x_i^s + t_i^s u_i^s) \geq \sum_{j\in D} d_j^s \tag{5.42}$$

$$x_i^s + u_i^s \leq m_i \qquad \forall i \in N \tag{5.43}$$

$$\sum_{i\in N} y_{i,j}^s = 1 \qquad \forall j \in D \tag{5.44}$$

$$d_j^s y_{i,j}^s \leq q_i x_i^s + t_i^s u_i^s \qquad \forall i \in N, \forall j \in D \tag{5.45}$$

$$x_i^s, u_i^s \geq 0 \text{ integer} \qquad \forall i \in N \tag{5.46}$$

$$y_{i,j}^s \geq 0 \qquad \forall i \in N, \forall j \in D \tag{5.47}$$

In this problem, the location decision variables $x_i^s, u_i^s$ are integer, and the serving variables $y_{i,j}^s$ are continuous. Therefore, it is a mixed integer programming (MIP) problem. The parameters $f_i''$ are different from the original $f_i$ in the Lagrangian relaxation process (see (2.17) in Chapter 2). But later we still use $f_i$ to indicate the coefficients of $x_i^s$ in SS

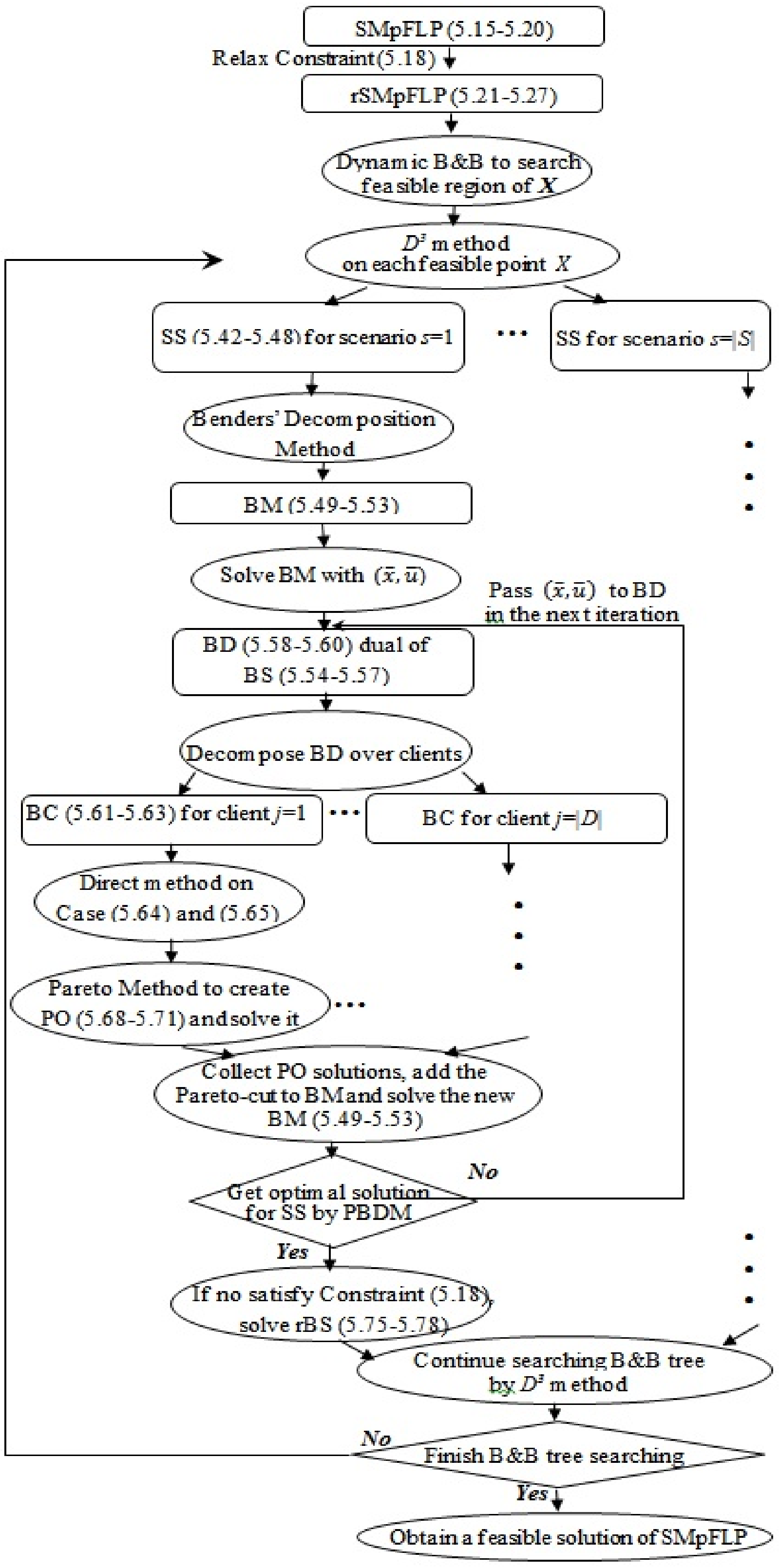
SMpFLP (5.15-5.20)
Relax Constraint (5.18)
rSMpFLP (5.21-5.27)
Dynamic B&B to search feasible region of X
D³ method on each feasible point X
SS (5.42-5.48) for scenario s=1
SS for scenario s=|S|
Benders' Decomposition Method
BM (5.49-5.53)
Solve BM with (x̄, ū)
Pass (x̄, ū) to BD in the next iteration
BD (5.58-5.60) dual of BS (5.54-5.57)
Decompose BD over clients
BC (5.61-5.63) for client j=1
BC for client j=|D|
Direct method on Case (5.64) and (5.65)
Pareto Method to create PO (5.68-5.71) and solve it
Collect PO solutions, add the Pareto-cut to BM and solve the new BM (5.49-5.53)
Get optimal solution for SS by PBDM
No
Yes
If no satisfy Constraint (5.18), solve rBS (5.75-5.78)
Continue searching B&B tree by D³ method
Finish B&B tree searching
No
Yes
Obtain a feasible solution of SMpFLP

Figure 5.1: The flow chart of $D^3$+PBDM on SMpFLP

for convenience.

Since we will use BDM to solve the above SS in each scenario, we may eliminate the index of $s$ on all variables and parameters for the sake of succinctness.

In the conventional BDM, the master subproblem includes integer variables and the fixed values of continuous variables. The fixed values of continuous variables are the solutions of Bender subproblem solved in the previous iteration of the BDM. The master subproblem of SS with integer facility decision variables, which is named Benders Main (BM) subproblem, is as follows:

$$\text{(BM)} \min_{\mathbf{x},\mathbf{u}} \ \beta = z \tag{5.48}$$

$$\text{s.t. } z \geq \sum_{i\in N}(f_i - \sum_{j\in D} q_i \bar{w}_{i,j})x_i + \sum_{i\in N}(h_i - \sum_{j\in D} t_i \bar{w}_{i,j})u_i + \sum_{j\in D}\bar{v}_j \tag{5.49}$$

$$\sum_{i\in N}(q_i x_i + t_i u_i) \geq \sum_{j\in D} d_j \tag{5.50}$$

$$x_i + u_i \leq m_i \qquad \forall i \in N \tag{5.51}$$

$$x_i, u_i \in \mathbf{Z}^+ \qquad \forall i \in N \tag{5.52}$$

There is one continuous variable $z$ and $2|N|$ integer variables in the master subproblem BM. Constraints (5.50–5.52) will exist in BM in all iterations of BDM. Additionally, one more constraint (5.49) with the updated $\bar{w}_{i,j}$ and $\bar{v}_j$ will be added in BM in each iteration of BDM.

In the proposed algorithm, we will invoke a commercial MIP solver to solve the BM directly. After obtaining a fixed feasible integer configuration of $\bar{x}_i$ and $\bar{u}_i$, the resulting subproblem to solve in BDM is called Benders Subproblem (BS):

$$\text{(BS)} \ \min_{\mathbf{y}} \ \alpha = \sum_{i\in N, j\in D} c_{i,j} d_j y_{i,j} + \sum_{i\in N} f_i \bar{x}_i + \sum_{i\in N} h_i \bar{u}_i \tag{5.53}$$

$$\text{s.t.} \ \sum_{i\in N} y_{i,j} = 1 \qquad \forall j \in D \tag{5.54}$$

$$- d_j y_{i,j} \geq -q_i \bar{x}_i - t_i \bar{u}_i \qquad \forall i \in N, \forall j \in D \tag{5.55}$$

$$y^s_{i,j} \geq 0 \qquad \forall i \in N, \forall j \in D \tag{5.56}$$

It is worthwhile to notice that there are only continuous variables in this formulation. We formulate the dual of BS as a Benders Dual subproblem (BD) for each scenario, neglecting the constant part $\sum_{i\in N} f_i \bar{x}_i + \sum_{i\in N} h_i \bar{u}_i$ in the objective function:

$$\text{(BD)} \ \max_{\mathbf{v},\mathbf{w}} \ \alpha' = \sum_{j\in D} v_j - \sum_{i\in N, j\in D} (q_i \bar{x}_i + t_i \bar{u}_i) w_{i,j} \tag{5.57}$$

$$\text{s.t.} \ v_j - d_j w_{i,j} \leq c_{i,j} d_j \qquad \forall i \in N, \forall j \in D \tag{5.58}$$

$$w_{i,j} \geq 0 \qquad \forall i \in N, \forall j \in D \tag{5.59}$$

where $v_j$ and $w_{i,j}$ are dual variables corresponding to constraints (5.54) and (5.55), respectively.

For a fixed solution $(\bar{x}, \bar{u})$, the BD may have a degenerate optimal basis. This implies that more than one Benders cut could be generated. Therefore, a smart choice of an optimal solution in the subproblem is very important to generate an efficient Benders cut to accelerate the BDM significantly.

In Magnanti and Wong (1981), they used a direct heuristic method to solve the Benders subproblem of the single-plant problem. However, due to the complicated structure of the multi-plant problem, we construct a tailored heuristic method.

Now, we may further decompose the dual subproblem BD (5.57–5.59) for each client

$j$, which we will call the Benders Client (BC) subproblem.

$$\text{(BC)} \max_{\mathbf{v},\mathbf{w}} \ \alpha_j = v_j - \sum_{i \in N} (q_i \bar{x}_i + t_i \bar{u}_i) w_{i,j} \tag{5.60}$$

$$\text{s.t.} \ v_j - d_j w_{i,j} \leq c_{i,j} d_j \qquad \forall i \in N \tag{5.61}$$

$$w_{i,j} \geq 0 \qquad \forall i \in N \tag{5.62}$$

For each fixed $j$, we are going to solve the above BC. For the trivial case where $d_j = 0$, the optimal value and the optimal solution is $\alpha^*_{k,j} = v^*_j = 0$.

For each fixed solution $(\bar{x}, \bar{u})$ on the integer variables, let us denote
$F = \{i | q_i \bar{x}_i + t_i \bar{u}_i \geq 1\}$ to be the set of locations with available products, and
$O = \{i | q_i \bar{x}_i + t_i \bar{u}_i = 0\}$ to be the set of locations without products.

For a given $j$, we sort the serving costs $c_{i,j}$ of all $i$ in $F$ in nondecreasing order

$$c_{1',j} \leq c_{2',j} \leq \cdots \leq c_{i'(j)-1,j} \leq c_{i'(j),j} \leq \cdots \leq c_{(|F|)',j}$$

where the sorting will create a new order of all elements in $F$ as $\{1', 2', \cdots, i'(j) - 1, i'(j), \cdots, (|F|')\} = F$.
Denote the serving capacity $C_{i'(j)} = \sum_{i=1'}^{i'(j)} (q_i \bar{x}_i + t_i \bar{u}_i)$, which is the sum of the capacities with the increasing serving cost. Since $(q_i \bar{x}_i + t_i \bar{u}_i)$ is positive for each $i \in F$, there will be in two cases:

$$1) \ C_{i'(j)-1} = d_j \text{ and } C_{i'(j)} > d_j. \tag{5.63}$$

$$2) \ C_{i'(j)-1} < d_j \text{ and } C_{i'(j)} > d_j. \tag{5.64}$$

We will analyze both cases and try to construct the solutions of each case. As the knapsack problem, we may directly get the unique optimal solution for the second case: $v^*_j = c_{i'(j),j} d_j$. One may notice that the direct heuristic method on the single-plant facility location problems in Magnanti and Wong (1981) is not same as our problem in the first case. We need to extend the method for our multi-plant problems. For the first case, the optimal solutions of BC (5.60–5.62) are not unique and actually contain infinite degenerate solutions. When $C_{i'(j)-1} = d_j$, then

$$C_{i'(j)-2} = C_{i'(j)-1} - (q_{i'(j)-1}\bar{x}_{i'(j)-1} + t_{i'(j)-1}\bar{u}_{i'(j)-1}) < d_j,$$

because the index $i'(j) - 1 \in F$. Hence, the degenerate solutions are in the interval:

$$c_{i'(j)-1,j}d_j \leq v_j^* \leq c_{i'(j),j}d_j.$$

The solution for $w_{i,j}$ in both cases is: $w_{i,j}^* = [(v_j^* - c_{i,j}d_j)/d_j]^+, \quad \text{for } i \in F.$

Any value in the interval set $[\ [(v_j^* - c_{i,j}d_j)/d_j]^+, +\infty)$ for $w_{i,j}, i \in O$ will not change the solution of BC (5.60–5.62). However, it will affect the solution in the later BM (5.48–5.52) where the cut will be formed as:

$$z \geq \sum_{i\in N}(f_i - \sum_{j\in D} q_i w_{i,j}^*)x_i + \sum_{i\in N}(h_i - \sum_{j\in D} t_j w_{i,j}^*)u_i + \sum_{j\in D} v_j^*.$$

In both cases, the optimal values are identical:

$$\alpha_j^* = v_j^* - \sum_{i\in F}(q_i\bar{x}_i + t_i\bar{u}_i)w_{i,j}^* - \sum_{i\in O}(q_i\bar{x}_i + t_i\bar{u}_i)w_{i,j}^* = c_{i'(j),j}d_j - \sum_{i=1'}^{i'(j)}(q_i\bar{x}_i + t_i\bar{u}_i)(c_{i'(j),j} - c_{i,j}).$$

It is easy to interpret this optimal dual solution in terms of the facility location problem. $v_j^*$ is the cost of servicing client $j$ corresponding to the solution $(\bar{x}, \bar{u})$. $w_{i,j}^*$ is the reduction in the cost of servicing customer $j$ from opening a new plant in location $i$.

So far, we have developed a heuristic method to solve BC (5.60–5.62). Following the conventional process of BDM, the optimal solution of BC (5.60–5.62) in the current iteration will be used to create the Benders cut (5.49). Then we add the cut to the Benders master subproblem and solve it in the next iteration. However, the Benders cut in the conventional BDM is not efficient for solving SS (5.41–5.47). In later Section 5.4.2, we will enhance the Pareto-cut and develop the refined BDM for solving SS more efficiently.

### 5.4.2 Pareto-Optimal Cut

As mentioned before, for a fixed solutions $(\bar{x}, \bar{u})$, the linear subproblem BC (5.60–5.62) may have a degenerate optimal basis. Therefore, it is possible that more than one Benders cut can be derived. This is the main reason that the CBDM has poor convergence when solving SS (5.41–5.47). We observed that the CBDM did not converge in many of our computational tests (see Table 5.2).

The concept of a *Pareto-optimal* cut was defined by Magnanti and Wong (1981).

**Definition 1.4.1.** The cut $z \geq g(\bar{w}^1) + xf(\bar{w}^1)$ *dominates* (or is strong than) the cut $z \geq g(\bar{w}^2) + xf(\bar{w}^2)$ if $g(\bar{w}^1) + xf(\bar{w}^1) \geq g(\bar{w}^2) + xf(\bar{w}^2)$ holds for all $x \in X$ with a strict inequality for at least one point $x' \in X$, where the set $X$ is the feasible region of a given problem.

**Definition 1.4.2.** A cut is *Pareto-optimal*, if no cut dominates it.

**Definition 1.4.3.** A cut is determined by the vector $\bar{w} \in W$. We say

a) $\bar{w}^1$ *dominates* $\bar{w}^2$, if the cut associated with $\bar{w}^1$ dominates the one with $\bar{w}^2$;

b) $\bar{w}^1$ is *Pareto-optimal*, if the corresponding cut is Pareto-optimal.

**Definition 1.4.4.** Let **R** denote the field of real numbers. The set of all $n$-tuples of real numbers forms an $n$-dimensional vector space over **R**, which is denoted as $\mathbf{R}^n$ and is called Euclidean space. The *affine hull*, **Aff(X)**, of a set $X$ in Euclidean space $\mathbf{R}^n$ is the set of all affine combinations of the elements of $S$. So

$$\mathbf{Aff}(\mathbf{X}) = \left\{ \sum_{i=1}^{k} \alpha_i x_i \mid x_i \in X, \alpha_i \text{ are scalar}, i = 1, 2, ..., k; \sum_{i=1}^{k} \alpha_i = 1; k = 1, 2, ... \right\}.$$

**Definition 1.4.5.** Let $X$ be a subset of the $n$-dimensional Euclidean space $\mathbf{R}^n$. The *relative interior* of $X$ is the interior of $X$ considered as a subset of its affine hull $\mathbf{Aff}(X)$, and is denoted by $\mathbf{ri}(X)$. Any point $x^o$ contained in $\mathbf{ri}(X^C)$, where $(X^C)$ is the convex hull of $X$ is called a *core point* of $X$.

**Proposition 4.** *(see [Magnanti and Wong (1981)], Theorem 1) Let $x^o$ be a core point of the set $X$. Let $U(\hat{x})$ for $\hat{x} \in X$ denote the set of optimal solutions to the optimization problem*

$$\max_{w \in W} \{f(w) + \hat{x} g(w)\} \tag{5.65}$$

*and let $w^o$ solve the problem*

$$\max_{w \in U(\hat{x})} \{f(w) + x^o g(w)\} \tag{5.66}$$

*Then $w^o$ is Pareto optimal.*

Based on the aforementioned proposition, solving the following subproblem for each $j$ will lead to a Pareto-optimal cut for the next iteration in CBDM:

$$\text{(PO)} \max_{v,w} \; v_j - \sum_{i \in N} (q_i x_i^o + t_i u_i^o) w_{i,j} \tag{5.67}$$

$$\text{s.t.} \; v_j - \sum_{i \in N} (q_i \bar{x}_i + t_i \bar{u}_i) w_{i,j} = \alpha_j^* \tag{5.68}$$

$$v_j - d_j w_{i,j} \leq c_{i,j} d_j \qquad \forall i \in N \tag{5.69}$$

$$w_{i,j} \geq 0 \qquad \forall i \in N \tag{5.70}$$

where $(\bar{x}, \bar{u})$ is the fixed solution of BM (5.48–5.52) from the previous iteration in CBDM, $\alpha_j^*$ is the optimal value in BC (5.60–5.62) corresponding to open plants $(\bar{x}, \bar{u})$, and $(x^o, u^o)$ is a core point of $X$, the feasible region of $(x, u)$ (see (**??**)) .

Substituting for $v_j$ in the objective function (5.67) from the equality constraint (5.68), the objective becomes

$$\max_{v,w} \sum_{i \in N} \left[q_i(\bar{x}_i - x_i^o) + t_i(\bar{u}_i - u_i^o)\right] w_{i,j} + \alpha_j^*$$

Since for any $i \in O$, $q_i \bar{x}_i + t_i \bar{u}_i = 0$ and $-q_i x_i^o - t_i u_i^o \leq 0$, the optimal solution is $w_{i,j} = [(v_j - c_{i,j} d_j)/d_j]^+$ for $i \in O$.

After we use the heuristic method to solve the BC (5.60–5.62) for both cases (5.63) and (5.64), we will solve PO (5.67–5.70) to obtain the Pareto-optimal cut. The linear problem PO (5.67–5.70) is piecewise linear as a function of $v_j$ and the optimal value of (5.60) can only be reached when the solutions of $v_j$ in PO (5.67–5.70) are in the interval $[c_{i'(j)-1,j}d_j, c_{i'(j),j}d_j]$. Hence,

$$\begin{aligned} &w_{i,j} = 0, \text{ for } i \in \{i'(j), \cdots, (|F|)'\} \text{ and} \\ &w_{i,j} = [(v_j - c_{i,j}d_j)/d_j]^+, \text{ for } i \in \{1', \cdots, i'(j)-1\} \end{aligned} \tag{5.71}$$

Denote $\pi_i = q_i(\bar{x}_i - x_i^o) + t_i(\bar{u}_i - u_i^o)$. With all above results, the objective function may be rewritten as a function of the variable $v_j$:

$$\begin{aligned} & v_j - \sum_{i\in N}(q_i x_i^o + t_i u_i^o) w_{i,j} \\ = {} & \alpha_j^* + \sum_{i\in N}[q_i(\bar{x}_i - x_i^o) + t_i(\bar{u}_i - u_i^o)]w_{i,j} \\ = {} & \alpha_j^* + \sum_{i=1'}^{i'(j)-1} \pi_i[(v_j - c_{i,j}d_j)/d_j]^+ + \sum_{i=C}\pi_i[(v_j - c_{i,j}d_j)/d_j]^+ \end{aligned} \tag{5.72}$$

To maximize the value of the above piecewise linear and concave function in $v_j$, we sort the serving cost $c_{i,j}, i \in C$, and compare with $c_{i'(j)-1,j}$ and $c_{i'(j),j}$ in (5.63):

$c_{\widetilde{1},j} \le \cdots \le c_{\widetilde{i}(p),j} \le c_{i'(j)-1,j} < c_{\widetilde{i}(p+1),j} \le \cdots \le c_{\widetilde{i}(q),j} < c_{i'(j),j} \le c_{\widetilde{i}(q+1),j} \le \cdots \le c_{(|C|),j}$

where $\{\widetilde{1}, \widetilde{2}, \ldots, \widetilde{i}(p), \widetilde{i}(p+1), \widetilde{i}(q), \widetilde{i}(q+1), \ldots, (|\widetilde{C}|)\} = C$.

First, set $v_j = c_{i'(j)-1,j}d_j$, calculate the gradient of function (5.72) as

$$s_{\widetilde{i}(p)} = \sum_{i=1'}^{i'(j)-1}\pi_i + \sum_{i=\{\widetilde{1},\ldots,\widetilde{i}(p)\}}\pi_i \tag{5.73}$$

If $s_{\widetilde{i}(p)} \le 0$, then $v_j = c_{i'(j)-1,j}d_j$ is optimal for PO(5.67–5.70). Otherwise, we iteratively calculate

$$s_{\widetilde{i}} = \sum_{i=1'}^{i'(j)-1}\pi_i + \sum_{i=\{\widetilde{1},\ldots,\widetilde{i}\}}\pi_i$$

with the upper bound of $\widetilde{i}$ increasing from $\widetilde{i}(p+1)$ to $\widetilde{i}(q)$ in set $C$, until $s_{\widetilde{i}} \leq 0$. Without loss of generality, assume for $\widetilde{i}(t) : s_{\widetilde{i}(t)-1} > 0$ and $s_{\widetilde{i}(t)} \leq 0$, then $v_j = c_{\widetilde{i}(t),j} d_j$ is optimal for PO(5.67–5.70).

If for all $\widetilde{i} \in C, s_{i'=(|\widetilde{C}|)} > 0$, then $v_j = c_{i'(j),j} d_j$ is optimal for PO (5.67–5.70). Once we decide the optimal value of $v_j$, $w_{i,j}$ can be set by (5.71).

### 5.4.3 A Solution of Scenario Subproblems with aggre-shipping Conditions

Until the BDM is convergent of solving scenario subproblems SS (5.41–5.47), we obtain fixed values of integer facility decision variables $(\bar{x}_i, \bar{u}_i)$, which is the solution of BM (5.48–5.52) and fixed values of continuous dual variables $(\bar{v}_j, \bar{w}_{i,j})$, which is the solution of BD. We did not solve BS in BDM for the continuous serving variables $y_{i,j}$, which are the dual variables of BD. Let us denote the fixed values of serving variables as $\bar{y}_{i,j}$ obtained from BD. The solution, $(\bar{x}_i, \bar{u}_i, \bar{y}_{i,j})$, is an optimal solution of SS (5.41–5.47), but it most likely does not satisfy the aggre-shipping conditions, since the continuous serving variables $y_{i,j}$ are solved from BD (5.57–5.59), which is the dual problem of BS (5.53–5.56) including pathway conditions (5.55). However, under the fixed values of facility decision variables, $(\bar{x}_i, \bar{u}_i)$, there may exist a feasible solution of serving variables with aggre-shipping conditions due to the total-capacity condition (5.50) added in BM.

Therefore, we will solve the restricted BS (rBS) with aggre-shipping constraints (5.76) instead of pathway constraints (5.55) at the end of BDM with the fixed values $(\bar{x}_i, \bar{u}_i)$.

$$\text{(rBS)} \ \min_{\mathbf{y}} \ \alpha = \sum_{i\in N, j\in D} c_{i,j} d_j y_{i,j} + \sum_{i\in N} f_i \bar{x}_i + \sum_{i\in N} h_i \bar{u}_i \tag{5.74}$$

$$\text{s.t.} \ \sum_{i\in N} y_{i,j} = 1 \qquad \forall j \in D \tag{5.75}$$

$$-\sum_{j\in D} d_j y_{i,j} \geq -q_i \bar{x}_i - t_i \bar{u}_i \qquad \forall i \in N \tag{5.76}$$

$$y^s_{i,j} \geq 0 \qquad \forall i \in N, \forall j \in D \tag{5.77}$$

The rBS is an LP with only continuous serving variables $y_{i,j}$ and can be quickly solved by invoking a commercial LP solver.

After we obtain solutions of rBS for each scenario, we finish solving all scenario subproblems of original SMpFLP at the current node and return a feasible solution back to the $D^3$ method. If it is better than the incumbent feasible solution, it will be saved as the new incumbent feasible solution in $D^3$ method. If a rBS in a scenario is infeasible, we consider that the scenario subproblem of the original SMpFLP is infeasible in the current node and let the $D^3$ method continue to search the next node in the B&B processing.

In this way, we can finally obtain a feasible solution of original SMpFLP at the end of $D^3$ method. The final feasible solution may not guarantee a global optimum of the original SMpFLP, although the proposed $D^3$+PBDM can solve a global optimum of the relaxed SMpFLP. One may notice in later computational tests in Table 5.3 and Table 5.4 that the final solutions are very good feasible solutions even if they are not global optimal solutions which are obtained by a commercial MIP solver.

If we only fixed the values ($\bar{x}_i$) of the first stage variables $x_i$, and solved the rBS with variables $u_i$ and $y_{i,j}$, the proposed $D^3$+PBDM would obtain a global optimum of the original SMpFLP since the $D^3$ method enumerated all feasible region of the first stage variables $x_i$. However, the rBS, which was an MIP with both integer variables $u_i$ and continuous variables $y_{i,j}$, would be challenging to be solved directly.

Now let us summarize the $D^3$+PBDM on solving SMpFLP (5.14–5.19).

```
Decompose relaxed SMpFLP (5.20–5.26) into SS (5.41–5.47) using $D^3$.
Loop to solve SS (5.41–5.47) for each scenario by refined BDM {
  Decompose SS into BM (5.48–5.52) and BD (5.57–5.59) by BDM.
  Solve BM (5.48, 5.51, 5.50, 5.52) without cuts and save solution as $(\bar{x}, \bar{u})$.
  While (stop condition of BDM) {
    Use the heuristic method to solve BC (5.60–5.62) with fixed value of $(\bar{x}, \bar{u})$.
    Solve PO (5.67–5.70) to create the Pareto-cut.
    Add the Pareto-cut (5.49) into BM (5.48–5.52).
    Solve BM (5.48–5.52) to update $(\bar{x}, \bar{u})$.
  } End of while
  Solve rBS (5.74–5.77) with incumbent fixed value $(\bar{x}, \bar{u})$.
  If rBS is infeasible {
    Stop Loop and report $D^3$ that original SMpFLP (5.14–5.19) is infeasible on current node.
  }
  Else { Save the solution and Continue Loop.}
} End of Loop
Updating the incumbent feasible solution of SMpFLP if it is better.
Calculating the lower and the upper bound of SMpFLP (5.14–5.19).
Continue $D^3$ on searching B&B tree.
Back to the top line for the next node in B&B tree.
```

By using the $D^3$ method, we can first decompose the relaxed SMpFLP (5.20–5.26) into SS (5.41–5.47) for each scenario. Then using the Benders Decomposition method, the SS (5.41–5.47) are decomposed into BD (5.57–5.59) including only continuous serving variables and BM (5.48–5.52) including only integer location variables. Based on the structure of BD (5.57–5.59), it can be decomposed into BC (5.60–5.62) for each client.

Each BC (5.60–5.62) is a linear programming problem with only continuous variables and can be solved directly by a heuristic method. By adding Pareto-cut into the BM (5.48–5.52), the SS (5.41–5.47) can be solved more efficiently by BDM. When the BDM is convergent but the optimal solution of SS does not satisfy the aggre-shipping conditions, we solve an LP rBS (5.74–5.77) with the incumbent fixed value of $(\bar{x}, \bar{u})$ for each scenario in order to solve a feasible solution of original SMpFLP at the current node.

## 5.5 Scenario Aggregation Method

By aggregating the rows and columns in Stochastic Linear Programming (SLP) problems we can significantly reduce the complexity of the original problem. The optimal value of the original SLP can be bounded using the solution of the aggregated problem within some error bounds. The aggregation method and error bounds in SLP were studied in [Birge 1985]. These bounds suggest that when aggregating the subproblems with small differences over some scenarios, we may solve the aggregated problem instead of solving the original subproblems. As the differences over scenarios vanish, the gap of the solutions from the aggregated problem and the original problem will vanish as well.

If $q_i^s, h_i^s$ and $t_i^s$ are identical in all scenarios, we may use the aggregation method to aggregate BC (5.60–5.62) over scenarios. Let us rewrite the (BC) problem for each scenario $s$:

$$\begin{aligned}
\text{(BC)} \ \max_{\mathbf{v},\mathbf{w}} \ & \alpha_j^s = v_j^s - \sum_{i\in N}(q_i\bar{x}_i + t_i\bar{u}_i)w_{i,j}^s \\
\text{s.t.} \ & v_j^s - d_j^s w_{i,j}^s \le c_{i,j}^s d_j^s \qquad && \forall i \in N \\
& w_{i,j}^s \ge 0 && \forall i \in N
\end{aligned}$$

Denote the optimal value $\alpha_j^{s*}$ and optimal solution $v_j^{s*}$ and $w_{i,j}^{s*}$ in (BC). The dual problem of (BC) called (DC) is:

$$
\begin{aligned}
\text{(DC)}\ \min_{\mathbf{y}}\ & \alpha_j^s = \sum_{i\in N} c_{i,j}^s d_j^s y_{i,j}^s \\
\text{s.t.}\ & \sum_{i\in N} y_{i,j}^s = 1 \\
& d_j^s y_{i,j}^s \le q_i \bar{x}_i + t_i \bar{u}_i \qquad \forall i \in N \qquad (5.78)\\
& y_{i,j}^s \ge 0 \qquad \forall i \in N
\end{aligned}
$$

With some value of $(\bar{x}, \bar{u})$, (DC) may be infeasible. As mentioned before, we can avoid the infeasible case by pre-checking when branching on each node in the B&B tree. Both (BC) and (DC) are linear problems and have the same optimal value if both are feasible and bounded. Denote the optimal solution $y_{i,j}^{s*}$ in (DC) with an optimal value $\alpha_j^{s*}$.

We place the following assumptions on the aforementioned (DC).

**Assumption 1:** Given small values $\epsilon_1$ and $\epsilon_2$, there are some scenarios $s \in \Psi \subseteq S$ such that:

(1) $\max_{s\in\Psi} d_j^s - \min_{s\in\Psi} d_j^s \le \epsilon_1$ and $\max_{s\in\Psi} c_{i,j}^s d_j^s - \min_{s\in\Psi} c_{i,j}^s d_j^s \le \epsilon_2$.

(2) there exists a bound $\bar{W}$, such that for optimal solutions $w_{i,j}^{s*} \le \bar{W}$ for $s \in \Psi$.

The part (1) in Assumption 1 indicates the scenarios with similar demands and similar total serving costs can be aggregated to solve. In the part (2), $w_{i,j}^s$ are marginal costs of the constraint (5.78). In the process of our algorithm, we will pre-check that the capacity of all opened multi-plants in each location will satisfy the demand of each client $j$ over all scenarios when branching on each node in the B&B tree. Therefore, optimal solutions $w_{i,j}^{s*}$ in (BC) are always bounded. Here, we denote the bound of all $w_{i,j}^{s*}$ as $\bar{W}$ for easy description in the late proof.

The aforementioned assumptions may restrict the problems to which we may apply our method to aggregate, but they are sufficiently general to include most realistic examples as we may pre-check the subproblems are bounded before we aggregate them.

Now we may aggregate (BC) over scenarios $s \in \Psi$ for given $\epsilon_1$ and $\epsilon_2$. First we

calculate probabilities of the aggregated scenarios and introduce the following symbols to simplify the presentation of the analysis on the aggregation bounds.
Let $p'_s = p_s / \sum_{s\in\Psi} p_s$, for $s \in \Psi$, then $\bar{d}_j = \sum_{s\in\Psi} p'_s d^s_j$, $\bar{c}_{i,j} = \sum_{s\in\Psi} p'_s c^s_{i,j} d^s_j$.
Then because of Assumption 1, $|d^s_j - \bar{d}_j| \le \epsilon_1$, and $|c^s_{i,j} d^s_j - \bar{c}_{i,j}| \le \epsilon_2$ for $s \in \Psi$.
The aggregated client problem of (BC) over scenario $s \in \Psi$ is:

$$\begin{aligned} \text{(AC)}\ \max_{\bar{\mathbf{v}},\bar{\mathbf{w}}}\ & \bar{\alpha}_j = \bar{v}_j - \sum_{i\in N}(q_i\bar{x}_i + t_i\bar{u}_i)\bar{w}_{i,j} \\ \text{s.t. } & \bar{v}_j - \bar{d}_j\bar{w}_{i,j} \le \bar{c}_{i,j} && \forall i \in N \\ & \bar{w}_{i,j} \ge 0 && \forall i \in N \end{aligned}$$

Denote the optimal value $\bar{\alpha}^*_j$ and optimal solution $\bar{v}^*_j$ and $\bar{w}^*_{i,j}$ in (AC). The dual problem of (AC) is

$$\begin{aligned} \text{(AD)}\ \min_{\bar{\mathbf{y}}}\ & \bar{\alpha}_j = \sum_{i\in N}\bar{c}_{i,j}\bar{y}_{i,j} \\ \text{s.t. } & \sum_{i\in N}\bar{y}_{i,j} = 1 \\ & \bar{d}_j\bar{y}_{i,j} \le q_i\bar{x}_i + t_i\bar{u}_i && \forall i \in N \\ & \bar{y}_{i,j} \ge 0 && \forall i \in N \end{aligned}$$

which has the same optimal value $\bar{\alpha}^*_j$ and the optimal solution is $\bar{y}^*_{i,j}$. Let $\bar{\bar{\alpha}}^*_j = \sum_{s\in\Psi} p^{s\prime}\alpha^{s*}_j$. If $\bar{\bar{\alpha}}^*_j$ can be bounded by $\bar{\alpha}^*_j$ with some error bounds, we may solve the aggregated client problem (AC) instead of solving $|\Psi|$ scenario problems BC (5.60–5.62).

**Proposition 5.** *Given Assumption 1, then we have $\bar{\alpha}_j^* - \epsilon_2 - \epsilon_1 \bar{W} \leq \bar{\bar{\alpha}}_j^* \leq \bar{\alpha}_j^* + \epsilon_2 + \epsilon_1 \bar{W}$.*

**Proof**: First we will prove the right inequality:

$$\bar{\bar{\alpha}}_j^* = \sum_{s \in \Psi} p^{s\prime} (v_j^{s*} - \sum_{i \in N} (q_i \bar{x}_i + t_i \bar{u}_i) w_{i,j}^{s*}) \tag{5.79}$$

$$\leq \sum_{s \in \Psi} p^{s\prime} \Bigg[ v_j^{s*} - \sum_{i \in N} (q_i \bar{x}_i + t_i \bar{u}_i) w_{i,j}^{s*} + \sum_{i \in N} (c_{i,j}^s d_j^s - v_j^{s*} + d_j^s w_{i,j}^{s*}) \bar{y}_{i,j}^* + \sum_{i \in N} (q_i \bar{x}_i + t_i \bar{u}_i - \bar{d} \bar{y}_{i,j}^*) w_{i,j}^{s*} \Bigg] \tag{5.80}$$

$$= \sum_{s \in \Psi} p^{s\prime} \Bigg[ \sum_{i \in N} c_{i,j}^s d_j^s \bar{y}_{i,j}^* + (1 - \sum_{i \in N} \bar{y}_{i,j}^*) v_j^{s*} + \sum_{i \in N} (d_j^s \bar{y}_{i,j}^* - q_i \bar{x}_i - t_i \bar{u}_i) w_{i,j}^{s*} + \sum_{i \in N} (q_i \bar{x}_i + t_i \bar{u}_i - \bar{d}_j \bar{y}_{i,j}^*) w_{i,j}^{s*} \Bigg] \tag{5.81}$$

$$= \sum_{s \in \Psi} p^{s\prime} \Bigg[ \sum_{i \in N} c_{i,j}^s d_j^s \bar{y}_{i,j}^* + \sum_{i \in N} (d_j^s - \bar{d}_j) \bar{y}_{i,j}^* w_{i,j}^{s*} \Bigg] \tag{5.82}$$

$$\leq \sum_{s \in \Psi} p^{s\prime} \Bigg[ \sum_{i \in N} (\bar{c}_{i,j} + \epsilon_2) \bar{y}_{i,j}^* + \sum_{i \in N} \epsilon_1 \bar{y}_{i,j}^* \bar{W} \Bigg] + (1 - \sum_{i \in N} \bar{y}_{i,j}^*) \epsilon_2 \tag{5.83}$$

$$= \sum_{i \in N} \bar{c}_{i,j} \bar{y}_{i,j}^* + \sum_{i \in N} \epsilon_2 \bar{y}_{i,j}^* + \epsilon_2 - \sum_{i \in N} \bar{y}_{i,j}^* \epsilon_2 + \epsilon_1 \bar{W} \sum_{i \in N} \bar{y}_{i,j}^* \tag{5.84}$$

$$= \bar{\alpha}_j^* + \epsilon_2 + \epsilon_1 \bar{W}. \tag{5.85}$$

We add two parts on the right side of (5.79) to get the inequality (5.80) which are both nonnegative due to feasibility of (BC) and (AD). Since $1 - \sum_{i \in N} \bar{y}_{i,j}^* = 0$ from the feasibility of (AD) and $w_{i,j}^{s*} \geq 0$ from the feasibility of (BC), we have (5.82). We get the inequality (5.83) because of the optimality of (AD) and definition of $\epsilon_2$.

Now we will prove the left inequality.

$$\begin{aligned}
\bar{\bar{\alpha}}_j^* &= \sum_{s\in\Psi} p^{s\prime} \sum_{i\in N} c_{i,j}^s d_j^s y_{i,j}^{s*} && (5.86)\\
&\geq \sum_{s\in\Psi} p^{s\prime} \left[ \sum_{i\in N} c_{i,j}^s d_j^s y_{i,j}^{s*} + (1-\sum_{i\in N} y_{i,j}^{s*})\bar{v}_j^* + \sum_{i\in N}(d_j^s y_{i,j}^{s*} - q_i\bar{x}_i - t_i\bar{u}_i)\bar{w}_{i,j}^* \right] && (5.87)\\
&= \sum_{s\in\Psi} p^{s\prime} \left[ (\bar{v}_j^* - \sum_{i\in N}(q_i\bar{x}_i + t_i\bar{u}_i)\bar{w}_{i,j}^*) + \sum_{i\in N}(c_{i,j}^s d_j^s - \bar{v}_j^* + d_j^s\bar{w}_{i,j}^*) y_{i,j}^{s*} \right] && (5.88)\\
&\geq \sum_{s\in\Psi} p^{s\prime} \left[ \bar{\alpha}_j^* + \sum_{i\in N}(\bar{c}_{i,j} - \epsilon_2 - \bar{v}_j^* + (\bar{d}_j - \epsilon_1)\bar{w}_{i,j}^*) y_{i,j}^{s*} \right] && (5.89)\\
&= \bar{\alpha}_j^* \sum_{s\in\Psi} p^{s\prime} + \sum_{s\in\Psi} p^{s\prime} \left[ \sum_{i\in N}(\bar{c}_{i,j} - \bar{v}_j^* + \bar{d}_j\bar{w}_{i,j}^*) y_{i,j}^{s*} - \sum_{i\in N} y_{i,j}^{s*}\epsilon_2 - \sum_{i\in N}\epsilon_1\bar{w}_{i,j}^* y_{i,j}^{s*} \right] && (5.90)\\
&\geq \bar{\alpha}_j^* + \sum_{s\in\Psi} p^{s\prime} \left[ -\epsilon_2 - \epsilon_1\bar{W}\sum_{i\in N} y_{i,j}^{s*} \right] && (5.91)\\
&= \bar{\alpha}_j^* - \epsilon_2 - \epsilon_1\bar{W}. && \square
\end{aligned}$$

Since the feasibility of (DC), where $1 - \sum_{i\in N} y_{i,j}^{s*} = 0$ and $\sum_{i\in N}(d_j^s y_{i,j}^{s*} - q_i\bar{x}_i - t_i\bar{u}_i) \leq 0, \bar{w}_{i,j}^* \geq 0$, we add two nonpositive parts on the right side of (5.86) to get the inequality (5.87). Since $(\bar{c}_{i,j} - \bar{v}_j^* + \bar{d}_j\bar{w}_{i,j}^*) \geq 0$ and $y_{i,j}^{s*} \geq 0$ from the optimality of (AC) and (DC), we can eliminate the nonnegative parts on the right side of (5.90) to get (5.91).

We may note that the error bounds depend only on $\epsilon_1, \epsilon_2$ and $\bar{W}$. As the differences of demands and serving costs over scenarios vanish, the aggregated error bounds of the original (BC) problems and the aggregated (AC) problem vanish.

If only the serving costs $c_{i,j}^s$ vary over scenarios, we may eliminate the restrictions of $\bar{W}$ and define the error bounds only using $\epsilon_2$ as in the following proposition.

**Corollary 6.** *If the demands $d_j^s$ are identical for $s \in \Psi$, then we have $\bar{\alpha}_j^* - \epsilon_2 \leq \bar{\bar{\alpha}}_j^* \leq \bar{\alpha}_j^* + \epsilon_2$.*

**Proof**: We can replace $\bar{d}_j$ with $d_j^s$ in the proof of Proposition 3 to get the results. The results can be viewed as a direct extension of Proposition 3.

We do not have to require $c^s_{i,j}$ and $d^s_j$ are within the given $\epsilon_1$ and $\epsilon_2$ for all scenarios. By pre-setting some acceptable $\epsilon_1$ and $\epsilon_2$, any group of scenarios whose $c^s_{i,j}$ and $d^s_j$ are within the corresponding $\epsilon_1$ and $\epsilon_2$ can be aggregated as one (AC) and the optimal solution of the original (BC) over these scenarios can be approximated by solving the aggregated (AC). The (BC) problems with $c^s_{i,j}$ or $d^s_j$ beyond the error bounds will not be aggregated but will be solved directly. When using this scenario aggregation method to solve (5.67–5.70) for a Pareto-optimal cut, the constraint (5.68) will be replaced by

$$v_j - \sum_{i \in N}(q_i\bar{x}_i + t_i\bar{u}_i)w_{i,j} = \bar{\alpha}^*_j. \tag{5.92}$$

## 5.6 Computational Results

### 5.6.1 PBDM vs. MIP solver vs. CBDM on MpFLP with benchmark data

In this section, we first show experiments of solving the relaxed MpFLP ((5.8) - (5.13)), in order to test the proposed PBDM without the $D^3$ framework. The relaxed deterministic MpFLP problem with pathway constraints has the same formulation as the decomposed sub-problem SS (5.41–5.47) except without integer multiplant variables in the second stage and the scenario index $s$. In order to illustrate how efficient the PBDM is on solving the relaxed MpFLP, we also use the Conventional Benders Decomposition Method (CBDM) and the direct method that just invokes the CPLEX 12.4 MIP solver to solve the whole MIP model, SS (5.41–5.47), directly. In the process of CBDM, the LP BC (5.60–5.62) and the MIP BM (5.48–5.52) are iteratively solved by CPLEX 12.4 LP solver and MIP solver with default configurations.

In the first group of tests, we use benchmark data of Capacity Facility Location Problems (CFLP) from [Beasley, 1990]. The benchmark data was also used to test the method in [Wu, et al 2006] except the setup costs were generated using non-linear functions of the size of the facility in the same site.

We downloaded the data in the text files for parameters: capacity $q'_i$, setup cost $f'_i$, demand $d'_j$ and the serving cost $c'_{i,j}$. The text files are available online at *http://people.brunel.ac.uk/mastjjb/jeb/orlib/capinfo.html*.

Since these data are for the single-plant CFLP, we adjust the parameters for MpFLP. In order to test different levels of multi-plants in each location, we generated the limits of multi-plants $m_i$ from three ranges: [8 12], [20 40] and [80 100], which are indicated under the column "M_range". Then we separate the capacity and setup cost from a single plant to multi-plants as $q_i = q'_i/m_i$, $f_i = f'_i/m_i$. The capacity and setup cost $t_i$ and $h_i$ for the second stage facility decision are generated based on $q_i$ and $f_i$ with a random number $r$ in the interval [0.8 1.2]: $t_i = q_i r$ and $h_i = f_i r$. We also scale down the demands for each client as $d_j = d'_j/N$ due to aggre-shipping conditions. The $c'_{i,j}$ is the total cost of serving all demand $d'_j$ for client $j$ in CFLP, but $c_{i,j}$ is the cost of each unit of product being served to client $j$ from facility $i$ in MpFLP. Therefore, we set $c_{i,j} = c'_{i,j}/d_j$.

In Table 5.1, we list the CPU times of the three methods. The total numbers of iterations ("Itr" in the table) are listed under PBDM and CBDM. Time is the total CPU time in seconds until the optimal solution is obtained or the solver stops. All three methods can solve all instances in Table 5.1 and obtain the optimal values, which are listed under the column "$z^*$". For example, the instance #1 is a small size problem with 50 potential facility locations and 50 clients. The limit of multi-plants $m_i$ is randomly generated from the range [8 12]. The benchmark data are from "cap134" data file. PBDM takes 2.03 seconds and 20 iterations to solve the problem, CPLEX MIP solver takes 2.09 seconds to solve it and CBDM take 10.91 seconds and 44 iterations to solve it. All experiments were tested on a Pentium 4 PC with a 3.2 GHz processor and 1 GB of RAM.

| | | | | | | PBDM | | MIP solver | CBDM | |
|---|---|---|---|---|---|---|---|---|---|---|
| | File | I | J | M_range | $z^*$ | Time | Itr | Time | Time | Itr |
| 1 | cap134 | 50 | 50 | [8 12] | 566134.2 | 2.03 | 20 | 2.09 | 10.91 | 44 |
| 2 | | | | [20 40] | 474310.3 | 0.50 | 10 | 0.84 | 2.86 | 33 |
| 3 | | | | [80 100] | 443484.2 | 0.03 | 2 | 0.36 | 2.11 | 27 |
| 4 | cap133 | 50 | 50 | [8 12] | 434234.1 | 0.58 | 11 | 0.17 | 9.47 | 36 |
| 5 | | | | [20 40] | 345471.7 | 0.31 | 10 | 1.25 | 2.97 | 34 |
| 6 | | | | [80 100] | 315239.9 | 0.06 | 4 | 0.34 | 2.52 | 33 |
| 7 | cap131 | 50 | 50 | [8 12] | 254564.1 | 14 | 44 | 0.92 | 39.36 | 78 |
| 8 | | | | [20 40] | 171443.8 | 1.26 | 17 | 0.70 | 3.69 | 41 |
| 9 | | | | [80 100] | 143655.0 | 0.36 | 7 | 0.33 | 2.72 | 36 |
| 10 | capc-7250 | 100 | 1000 | [8 12] | 3067316.4 | 45.39 | 53 | 208.66 | 373.19 | 153 |
| 11 | | | | [20 40] | 1684552.1 | 0.88 | 13 | 184.09 | 192.3 | 110 |
| 12 | | | | [80 100] | 1287827.3 | 2.47 | 23 | 155.36 | 271.63 | 134 |
| 13 | capc-5000 | 100 | 1000 | [8 12] | 3590776.0 | 2.08 | 21 | 102.42 | 224.81 | 116 |
| 14 | | | | [20 40] | 2202876.5 | 1.24 | 17 | 143.77 | 196.75 | 110 |
| 15 | | | | [80 100] | 1745874.2 | 1.30 | 7 | 156.77 | 177.78 | 103 |
| 16 | capb-5000 | 100 | 1000 | [8 12] | 4107824.6 | 9.33 | 45 | 153.3 | 306.64 | 139 |
| 17 | | | | [20 40] | 2582080.5 | 0.58 | 12 | 131.95 | 173.3 | 106 |
| 18 | | | | [80 100] | 2037781.3 | 0.06 | 2 | 34.2 | 155.16 | 98 |
| 19 | capb-8000 | 100 | 1000 | [8 12] | 3295064.5 | 61.50 | 52 | 578.19 | 501.84 | 152 |
| 20 | | | | [20 40] | 1872446.6 | 1.23 | 19 | 124.36 | 205.59 | 118 |
| 21 | | | | [80 100] | 1333730.3 | 0.23 | 6 | 81.59 | 166.5 | 102 |
| 22 | capa-8000 | 100 | 1000 | [8 12] | 5046384.5 | 0.52 | 10 | 129.78 | 211.02 | 109 |
| 23 | | | | [20 40] | 3496387.1 | 0.22 | 6 | 132.59 | 155.52 | 100 |
| 24 | | | | [80 100] | 2985724.3 | 0.08 | 2 | 60.08 | 144.09 | 99 |
| 25 | capa-14000 | 100 | 1000 | [8 12] | 3807123.9 | 0.80 | 15 | 201.75 | 222.86 | 111 |
| 26 | | | | [20 40] | 2300232.6 | 0.67 | 14 | 114.19 | 231.09 | 111 |
| 27 | | | | [80 100] | 1778291.8 | 0.38 | 3 | 266.89 | 176.77 | 100 |

Table 5.1: Computational results on solving relaxed MpFLP with benchmark data by three methods

Results from this group of tests are shown in Table 5.1. The PBDM method performs better than the CPLEX MIP solver in most cases. Even the CBDM is not very inferior to CPLEX MIP solver and it can solve all instances. PBDM was up to 751 times faster than

CPLEX MIP solver. On average over all instances, PBDM was 170 times faster than the CPLEX MIP solver. In tests #4, #7, #8 and #9, CPLEX MIP solver was faster. CBDM can also solve all instances with benchmark data. However, CBDM cannot solve most instances with random generated data in Table 5.2 in the next section. Therefore, we do additional computational tests on randomly generated data besides benchmark data.

### 5.6.2 PBDM vs. MIP solver vs. CBDM on MpFLP with random data

In the second group of tests, we randomly generate the data using uniform distributions. We generate the setup costs $f_i$ from the range [8000 10000], the demand $d_j$ from the range [800 1000], and the serving cost $c_{i,j}$ from the range [0.02 0.2]. In order to test different levels of multi-plants, we randomly generate the limit of multi-plants $m_i$ from 3 ranges: [8 12], [20 40] and [80 100], as we did in Table 5.1 in the previous section. Similarly as in Chapter 3, the ranges of data are adapted from facility distributions in Management Information Systems of Municipal and Provincial Power Bureaus in Yantai city, Qingdao city and Tianjin city in 1998 - 2001.

Another reason for computational tests on random data is that even CBDM performs not very bad on solving some MpFLPs generated from benchmark data in the previous section but it cannot solve most tests on random data in Table 5.2.

In this group of tests in Table 5.2, PBDM performs much better than the CPLEX MIP solver. PBDM was up to 491 times faster than the CPLEX MIP solver. On average over all instances, PBDM was 53 times faster than the CPLEX MIP solver. The CPLEX MIP solver was faster only in Test #2, since the overhead of processing PBDM takes more time than CPU time on solving small-size MIPs. The results suggest that the Pareto-cuts accelerate the convergence in PBDM for solving multi-plant facility location problems. The optimal solutions of each test, which can be solved by both PBDM and MIP solver, are listed under the column "$z^*$".

| | | | | | PBDM | | MIP solver | CBDM | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | I | J | M_range | $z^*$ | Time | Itr | Time | Time | Itr | $z'$ | Gap(%) |
| 1 | 10 | 80 | [8 12] | 197600.8 | 0.02 | 2 | 0.23 | 0.48 | 12 | 197600.8 | - |
| 2 | 10 | 80 | [20 40] | 405114.2 | 0.10 | 2 | 0.08 | 335.83 | 2000 | 405114.2 | 1.78 |
| 3 | 10 | 80 | [80 100] | 1164339.3 | 0.10 | 2 | 0.11 | 290.14 | 2000 | 1164339.3 | 0.66 |
| 4 | 20 | 50 | [8 12] | 132840.0 | 0.05 | 2 | 0.14 | 1.86 | 20 | 132840.0 | - |
| 5 | 20 | 50 | [20 40] | 391918.1 | 0.03 | 2 | 0.16 | 550.61 | 2000 | 391918.1 | 1.15 |
| 6 | 20 | 50 | [80 100] | 1124443.8 | 0.02 | 2 | 0.08 | 539.61 | 2000 | 1124443.8 | 0.42 |
| 7 | 30 | 100 | [8 12] | 132892.2 | 0.05 | 2 | 0.53 | 885.30 | 2000 | 132892.2 | 6.54 |
| 8 | 30 | 100 | [20 40] | 386046.6 | 0.05 | 5 | 0.25 | 866.28 | 2000 | 386081.0 | 2.51 |
| 9 | 30 | 100 | [80 100] | 1119595.4 | 0.03 | 3 | 0.31 | 807.44 | 2000 | 1119595.4 | 0.78 |
| 10 | 60 | 150 | [8 12] | 144559.2 | 1.08 | 9 | 10.47 | 1800.81 | 1609 | 144559.2 | 10.23 |
| 11 | 60 | 150 | [20 40] | 396240.3 | 0.25 | 2 | 7.52 | 1801.52 | 1761 | 396240.3 | 3.66 |
| 12 | 60 | 150 | [80 100] | 1147679.3 | 0.16 | 5 | 1.14 | 1800.70 | 2000 | 1147806.2 | 1.32 |
| 13 | 100 | 500 | [8 12] | 172109.8 | 4.16 | 21 | 114.97 | 2207.38 | 80 | 172371.5 | 29.88 |
| 14 | 100 | 500 | [20 40] | 420727.0 | 0.47 | 8 | 29.63 | 211.64 | 112 | 420727.0 | - |
| 15 | 100 | 500 | [80 100] | 1155310.9 | 0.22 | 7 | 32.50 | 931.31 | 107 | 1155310.9 | - |
| 16 | 100 | 1000 | [8 12] | 218600.8 | 42.19 | 24 | 3509.81 | 2617.16 | 81 | 219358.2 | 52.91 |
| 17 | 100 | 1000 | [20 40] | 467497.7 | 0.58 | 7 | 46.08 | 257.56 | 109 | 467497.7 | - |
| 18 | 100 | 1000 | [80 100] | 1202109.7 | 0.20 | 5 | 98.33 | 3373.25 | 96 | 1203399.4 | 8.13 |
| 19 | 200 | 2000 | [8 12] | 311942.1 | 1364.14 | 268 | 9197.94 | 11547.55 | 492 | 311942.1 | 0.01 |
| 20 | 200 | 2000 | [20 40] | 557796.5 | 371.73 | 45 | 2656.97 | 8763.41 | 41 | 559457.6 | 40.73 |
| 21 | 200 | 2000 | [80 100] | 1326455.5 | 8.19 | 46 | 808.86 | 23469.92 | 57 | 1330617.3 | 15.51 |

Table 5.2: Computational results on solving relaxed MpFLP with random data by three methods

Without efficient cuts, conventional BDM may not even converge to optimal solutions in solving MpFLPs with randomly generated data in Table 5.2. The "Time" column under the CBDM column indicates either the total computer time if the optimal solution is obtained or the stop time. The final solutions obtained by the CBDM are listed under the "$z'$" column. The "Gap (%)" column indicates the percentage of gap between the best feasible solution and the best relaxed bound if the algorithm is stopped at a time or an iteration limit. Sometimes the CBDM already obtained the optimal solution in some instances (#2, #3, #5, #6, #7, #9, #10, #11, and #19), but the gap is still positive since the gap is not tight enough to stop the algorithm.

As we hypothesized, the degenerate cuts in the second case (5.64) will lead to a slow convergence rate in CBDM and therefore CBDM performs very poorly and cannot even solve small size problems except for the small-size instances #1, #4, #14, #15 and #17. However CBDM is still much slower than PBDM and CPLEX MIP solver in all instances. We set the half-hour (1800 seconds) time limit or 2000 iterations for CBDM in all tests except the last three large-size tests. For example, CBDM is stopped after 2000 iterations for 335.8 seconds and the optimality gap is 1.78% for instance #2 in Table 5.2. For the instance #10, CBDM stopped at the time-limit, approximately 1800 seconds, with 1609 iterations and a gap of 10.23%. The experiments in Table 5.2 show that the developed PBDM can solve MpFLPs with random data very efficiently and it takes far fewer iterations in Benders Decomposition process to get a global optimum. But the CBDM without pursuing Pareto-cuts, may repeatedly create the degenerate cuts and cannot converge to a global optimum within 2000 iteration limit (see instances #2, #3, #5 - #9, and #12).

As we have shown, the proposed $D^3$ method can decompose the whole model of stochastic MIPs into scenario sub-problems, which inherit the same structure of the original problem. Then we may invoke either the commercial MIP solver or conventional or tailored methods to solve the sub-problems, as we did in Chapter 3 and Chapter 4, respectively. However, the aforementioned experiments in Table (5.1) and Table (5.2) have shown that even the commercial MIP solver or the conventional Benders Decomposition method cannot solve the deterministic MpFLP efficiently, due to the general integer decision variables in the first stage and mixed integer variables in the second stage. Therefore, we have developed Pareto-Benders Decomposition method in Section

5.4 to avoid degenerate cuts in CBDM and we can combine the $D^3$ method with PBDM to solve large-size SMpFLP more efficiently.

### 5.6.3 $D^3$+PBDM vs. MIP solver on SMpFLP with benchmark data

In the section, we will test the $D^3$+PBDM method on solving the SMpFLP (5.14–5.19) with benchmark data. We also invoked CPLEX MIP solver to solve the whole SMpFLP (5.14–5.19) directly. We note that the MIP solver could not get the optimal solution in some instances. We mark with * on the instances when the CPLEX solver stops with "out of memory" errors. The final solutions solved by the $D^3$+PBDM are listed under the column "$z'$" since the solution may not be a global optimum for the original SMpFLP. The final solutions obtained by CPLEX MIP solver are listed under the column "$z''$". If CPLEX MIP solver can solve the SMpFLP without errors, the solution is a global optimum. The numbers in column "N" under the $D^3$+PBDM indicate the total number of nodes in the B&B tree that were searched by the $D^3$ framework.

We use the benchmark data of CFLP from Beasley (1990) to test the $D^3$+PBDM vs.the MIP solver to solve the SMpFLP. After processing the same way in Section 5.6.1 to read the benchmark data, we generate the uncertainty parameters in scenarios, $d_j^s$ and $c_{i,j}^s$, in the range between the minimal value and the maximal value of all $d_j$ and $c_{i,j}$ from the benchmark.

In Table 5.3, we note that CPLEX MIP solver could not solve the whole MIP of SMpFLP on benchmark data within the memory limit in some instances (#5, #6, #10 – #12, #21, #22, #24–#26, and #28 – #33), which are marked with a * on CPU time. CPLEX MIP solver can solve some small-size instances (#1 – #3, #8, #14, #15, #17 and #27) faster than the $D^3$+PBDM. However, in the cases (#4, #7, #13, #16 and #18 – #20) where both the MIP solver and the $D^3$+PBDM solved the problems, the

$D^3$+PBDM was up to 199 times faster than MIP solver. One may notice that in some instances (#9, #24, #30 and #33), the $D^3$+PBDM solved the original SMpFLP faster than the CPLEX MIP solver, but the feasible solutions are inferior to the ones obtained by the CPLEX MIP solver. We mark the solutions obtained by the $D^3$+PBDM in pink color.

### 5.6.4 $D^3$+PBDM vs. MIP solver on SMpFLP with random data

In this section, we will test the $D^3$+PBDM method on solving the SMpFLP with random data. We use the same method as in Section 5.6.2 to generate test data sets. For example, the demands were generated in the range [800 1000] with a uniform distribution in two dimensions: $s$ and $j$.

In Table 5.4, one may notice that the CPLEX MIP solver can obtain an optimal solution faster than the $D^3$+PBDM in some small instances (#3 and #23 ) where both can solve problems in short times. However, the $D^3$+PBDM was up to 4.2 times faster than the CPLEX solver in the cases (#4, #6, #8 and #13) where the CPLEX solver could eventually obtain the optimum but the $D^3$+PBDM outperformed the CPLEX solver. In many cases, the CPLEX solver cannot solve the SMpFLP and is stopped with errors. Only in Test #23, the feasible solution in pink color obtained by the $D^3$+PBDM is inferior to the optima solution obtained by the CPLEX solver.

In Table 5.1 to 5.4, we tested three different levels of multi-plants in each group of problems with same numbers of $I$, $J$, and $S$. The limits of multi-plants, $m_i$, are generated from three range: [8 12], [20 40] and [80 100]. In the formulation of relaxed MpFLP (5.8-5.13), the constraint (5.10) indicates that the feasible regions of integer variables $x_i$ in [8 12] range is smaller than the ones in [80 100] range. When reviewing all tests in the first two tables (Table 5.1 and 5.2), one may notice that CPU times of solving MpFLPs with a smaller multi-plant range, [8 12], are longer than the other two larger ranges by all

| | | | | | | $D^3$+PBDM | | | MIP solver | |
|---|---|---|---|---|---|---|---|---|---|---|
| # | File | I | J | S | M_range | Time | N | $z'$ | Time | $z''$ |
| 1 | cap41 | 3 | 10 | 10 | [8 12] | 7.6 | 24 | 5178.9 | 7.1 | 5178.9 |
| 2 | | | | | [20 40] | 75.7 | 166 | 5195.1 | 14.2 | 5195.1 |
| 3 | | | | | [80 100] | 74.9 | 280 | 5251.2 | 15.8 | 5251.2 |
| 4 | cap81 | 3 | 10 | 20 | [8 12] | 31.0 | 32 | 5246.2 | 6179.0 | 5246.2 |
| 5 | | | | | [20 40] | 202.8 | 210 | 5261.0 | 11714.8* | 5261.0 |
| 6 | | | | | [80 100] | 362.7 | 698 | 5319.6 | 3197.3* | 5321.1 |
| 7 | cap41 | 4 | 20 | 5 | [8 12] | 46.8 | 162 | 14133.1 | 91.3 | 14133.1 |
| 8 | | | | | [20 40] | 460.1 | 1918 | 14202.8 | 203.3 | 14202.8 |
| 9 | | | | | [80 100] | 29.3 | 230 | 14365.6 | 804.3 | 14365.3 |
| 10 | cap81 | 4 | 30 | 20 | [8 12] | 337.0 | 264 | 15969.2 | 2853.1* | 16025.2 |
| 11 | | | | | [20 40] | 2149.7 | 2228 | 16051.5 | 2402.3* | 16072.1 |
| 12 | | | | | [80 100] | 646.4 | 1204 | 16255.5 | 1381.4* | 16256.7 |
| 13 | cap81 | 5 | 10 | 5 | [8 12] | 21.0 | 60 | 7544.3 | 409.7 | 7544.3 |
| 14 | | | | | [20 40] | 50.1 | 196 | 7560.3 | 12.1 | 7560.3 |
| 15 | | | | | [80 100] | 14.5 | 86 | 7586.9 | 8.7 | 7586.9 |
| 16 | cap41 | 5 | 12 | 6 | [8 12] | 16.9 | 42 | 17642.4 | 210.5 | 17642.4 |
| 17 | | | | | [20 40] | 141.7 | 504 | 17739.6 | 30.0 | 17739.6 |
| 18 | | | | | [80 100] | 235.5 | 1640 | 17950.6 | 2708.0 | 17950.6 |
| 19 | cap81 | 5 | 12 | 8 | [8 12] | 22.9 | 36 | 18941.1 | 3299.1 | 18941.1 |
| 20 | | | | | [20 40] | 208.0 | 524 | 19054.8 | 1236.1 | 19054.8 |
| 21 | | | | | [80 100] | 553.0 | 2476 | 19315.7 | 3465.6* | 19315.8 |
| 22 | cap64 | 5 | 40 | 8 | [8 12] | 62.4 | 76 | 48015.3 | 4599.9* | 48057.5 |
| 23 | | | | | [20 40] | 166.5 | 298 | 48141.0 | 3297.7 | 48141.0 |
| 24 | | | | | [80 100] | 992.3 | 5550 | 48786.6 | 1748.4* | 48785.7 |
| 25 | cap81 | 6 | 30 | 10 | [8 12] | 308.2 | 388 | 23090.8 | 2827.6* | 23101.4 |
| 26 | | | | | [20 40] | 173.9 | 546 | 23239.6 | 1993.3* | 23246.3 |
| 27 | | | | | [80 100] | 994.0 | 4466 | 23301.2 | 387.8 | 23301.5 |
| 28 | cap64 | 8 | 50 | 10 | [8 12] | 280.2 | 350 | 78744.8 | 2902.7* | 78776.7 |
| 29 | | | | | [20 40] | 959.5 | 2250 | 78887.9 | 656.0* | 78946.2 |
| 30 | | | | | [80 100] | 2733.4 | 12874 | 79487.5 | 1699.3* | 79487.1 |
| 31 | cap64 | 16 | 50 | 8 | [8 12] | 29923.9 | 101026 | 156183.0 | 1853.8* | 156218.6 |
| 32 | | | | | [20 40] | 26414.7 | 92704 | 156199.9 | 2299.9* | 156202.5 |
| 33 | | | | | [80 100] | 23191.7 | 51176 | 158941.6 | 2223.8* | 158940.6 |

Table 5.3: Computational results on solving Stochastic MpFLP with benchmark data by two methods

| | | | | | $D^3$+PBDM | | | MIP solver | |
|---|---|---|---|---|---|---|---|---|---|
| # | I | J | S | M_range | Time | N | $z'$ | Time | $z''$ |
| 1 | 3 | 10 | 100 | [8 12] | 119.2 | 42 | 56017.5 | 2131.2* | 56017.5 |
| 2 | 3 | 10 | 400 | [8 12] | 2219.9 | 288 | 184984.5 | 2355.4* | 185048.8 |
| 3 | 5 | 40 | 5 | [8 12] | 9.84 | 42 | 81803.5 | 1.2 | 81803.5 |
| 4 | 5 | 40 | 10 | [8 12] | 26.3 | 94 | 83534.5 | 111.3 | 83534.5 |
| 5 | 5 | 40 | 20 | [8 12] | 37.8 | 64 | 84448.2 | 2618.3* | 84539.0 |
| 6 | 5 | 40 | 8 | [8 12] | 666.9 | 3108 | 326613.3 | 799.3 | 326613.3 |
| 7 | 5 | 40 | 8 | [20 40] | 191.2 | 208 | 234622.8 | 6787.9* | 234663.5 |
| 8 | 5 | 40 | 8 | [80 100] | 330.4 | 204 | 643343.5 | 809.0 | 643343.5 |
| 9 | 5 | 40 | 80 | [80 100] | 2054.0 | 2822 | 3896621.9 | 3482.2* | 3896621.9 |
| 10 | 5 | 40 | 40 | [8 12] | 57.3 | 64 | 84001.4 | 1954.6* | 84089.8 |
| 11 | 5 | 40 | 40 | [20 40] | 2002.4 | 388 | 242189.5 | 1704.6* | 242667.0 |
| 12 | 5 | 40 | 40 | [80 100] | 959.1 | 128 | 644961.7 | 2798.3* | 644961.7 |
| 13 | 6 | 36 | 8 | [8 12] | 121.1 | 78 | 101659.1 | 330.7 | 101659.1 |
| 14 | 6 | 36 | 8 | [20 40] | 186.9 | 128 | 287771.8 | 2946.3* | 287771.8 |
| 15 | 6 | 36 | 8 | [80 100] | 623.1 | 526 | 794675.8 | 6467.4* | 794719.1 |
| 16 | 6 | 36 | 10 | [8 12] | 31.8 | 92 | 101527.9 | 4724.8* | 101688.1 |
| 17 | 6 | 36 | 10 | [20 40] | 435.1 | 462 | 292239.8 | 2719.6* | 292710.2 |
| 18 | 6 | 36 | 10 | [80 100] | 316.9 | 1158 | 792887.4 | 2466.3* | 792916.3 |
| 19 | 6 | 36 | 20 | [8 12] | 398.3 | 90 | 102087.6 | 2503.9* | 102905.8 |
| 20 | 6 | 36 | 20 | [20 40] | 642.9 | 192 | 287568.6 | 2321.3* | 287600.2 |
| 21 | 6 | 36 | 20 | [80 100] | 3428.2 | 804 | 793806.6 | 1781.2* | 793924.3 |
| 22 | 6 | 50 | 20 | [8 12] | 58.4 | 94 | 100819.5 | 2694.3* | 100965.5 |
| 23 | 8 | 40 | 5 | [8 12] | 385.0 | 362 | 134690.3 | 256.1 | 134690.1 |
| 24 | 8 | 80 | 20 | [8 12] | 577.8 | 356 | 132938.5 | 3599.6* | 133260.3 |
| 25 | 8 | 80 | 20 | [20 40] | 4055.1 | 4004 | 386647.5 | 2696.5* | 386647.5 |
| 26 | 8 | 80 | 20 | [80 100] | 19732.4 | 40874 | 1175042.6 | 1864.8* | 1175518.7 |
| 27 | 16 | 50 | 10 | [8 12] | 2528.1 | 4658 | 266201.9 | 2117.3* | 266724.4 |
| 28 | 16 | 50 | 10 | [20 40] | 12607.6 | 32994 | 791596.2 | 542.8* | 791692.6 |
| 29 | 16 | 50 | 10 | [80 100] | 34395.7 | 103024 | 2375455.0 | 554.1* | 2375889.9 |

Table 5.4: Computational results on solving Stochastic MpFLP with random data by two methods

three algorithms (PBDM, CPLEX MIP solver and CBDM), respectively. For instance, Test #10 in Table 5.1 is solved by three ways, for 45.39 seconds, 208.66 seconds and 373.19 seconds, respectively. The times are all longer than corresponding CPU times of Test #12 with larger range [80 100], which are 2.47 seconds, 155.36 seconds and 271.63 seconds. The longer CPU times were also observed in Test #19 of three solving methods versus times in Test #21.

One possible explanation is that both PBDM and CBDM iteratively add cuts to cut off worse feasible solutions in order to solve for a global optimum of deterministic MpFLPs, which is essential as the Branch-and-Cut (B&C) process. When solving large-scale MIPs with smaller feasible regions of integer variables (especially with binary variables), the cuts may switch the incumbent solution in Benders Decomposition iterations significantly and frequently. Therefore the BDM may be harder to converge to a global optimum.

The advancement of new versions of CPLEX MIP solver is first solving LP of large-scale MIPs by the advanced simplex method, then approaching to an integer global solution of original MIP by some approximate methods or B&C method. The CPLEX MIP solver will solve the whole model of deterministic MpFLPs or stochastic MpFLPs as one large-scale MIP, so it may take shorter times to solve instances with larger multi-plant ranges than ones with smaller range. Especially if the advanced simplex method obtained an integer optimal solution directly or a very good seed of integer optimal solution, then it takes much less time to solve the whole large-scale MIP even with larger feasible regions (e.g., Test #14, #15, #17 and #27 in Table 5.3).

On the contrary, it is intuitive that the Branch-and-Bound (B&B) method may take less time branching and searching on smaller feasible regions of integer variables to enumerate feasible solutions than the times of solving large regions. When solving stochastic version of MpFLPs, a CPU time of implementing the $D^3$ method, which is essentially a B&B process, dominates and takes much longer than a time of implementing the PBDM to solve decomposed small-size subproblems. Therefore, Table 5.3 and Table 5.4 illustrate that the $D^3$+PBDM solves a global optimum with large multi-plant ranges slower than tests with small ranges (e.g., Test #6, #18, #21, #24, #27 and #30 in Table 5.3, also Test #15, #21, #26 and #29 in Table 5.4). However the CPLEX MIP solver still follows the same pattern in some instances in Table 5.1 and 5.2, where it takes longer time of solving small ranges than large ranges. The aforementioned intuitive explanation of CPU times on small and large ranges is only based on the experiments in Table 5.1 to 5.4, which requires more theoretical study to validate it.

### 5.6.5 Aggregation method with the $D^3$+PBDM on SMpFLP with benchmark data

Based on the previous tests, one may notice that the $D^3$+PBDM takes a long time to solve SMpFLPs in some large instances. In this section we will test the proposed aggregation method to accelerate the $D^3$+PBDM to solve large-size problems.

First the SMpFLP problems are generated by using the benchmark data as in Section 5.6.3 and use the $D^3$+PBDM to solve it. The number of scenarios for those problems is indicated as "$S'$ " in Table 5.5. Then we generate data for un-aggregated, large-size problems with a large number of scenarios that are indicated by "$S$". We use "$z_\epsilon$" to represent the percentage of error of optimal values of the large problems relative to the aggregated problems, which is calculated as

$$z_\epsilon = \frac{(z^* - z_s^*)}{z^*} \times 100\%,$$

where $z_s^*$ and $z^*$ are the optimal value of aggregated problems and un-aggregated problems respectively.

In order to set the error values, $\epsilon_1$ and $\epsilon_2$, to bound $d_j^s$ and $c_{i,j}^s d_j^s$ over scenarios under Condition (1) in Assumption 1: $\max_{s\in S} d_j^s - \min_{s\in S} d_j^s \leq \epsilon_1$ and $\max_{s\in S} c_{i,j}^s d_j^s - \min_{s\in S} c_{i,j}^s d_j^s \leq \epsilon_2$.

| | | Large problems | | | $D^3$+PBDM | | Aggregated problems | | Aggregation method & $D^3$+PBDM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | File | I | J | S | Time | N | $S'$ | $\epsilon$% | Time | N | $z_\epsilon$ |
| 1 | cap41 | 3 | 10 | 100 | 181.3 | 38 | 10 | 10% | 18.4 | 24 | -0.14 % |
| 2 | | | | | | | 20 | 5% | 38.6 | 34 | 1.40 % |
| 3 | | | | | | | 5 | 40% | 10.5 | 32 | -2.83 % |
| 4 | cap43 | 3 | 12 | 200 | 14554.6 | 310 | 10 | 10% | 476.9 | 254 | -0.29 % |
| 5 | | | | | | | 20 | 5% | 1041.4 | 320 | 0.44 % |
| 6 | | | | | | | 5 | 40% | 107.7 | 118 | 2.83 % |
| 7 | cap81 | 4 | 20 | 120 | 9727.5 | 450 | 10 | 10% | 2266.7 | 980 | 1.52 % |
| 8 | | | | | | | 20 | 5% | 3368.5 | 1130 | 0.50 % |
| 9 | | | | | | | 5 | 40% | 680.3 | 604 | 1.54 % |
| 10 | cap64 | 5 | 40 | 80 | 1442.3 | 240 | 10 | 10% | 830.4 | 964 | -2.13 % |
| 11 | | | | | | | 20 | 5% | 1584.6 | 962 | -1.19 % |
| 12 | | | | | | | 5 | 40% | 384.5 | 986 | -1.76 % |
| 13 | cap83 | 6 | 30 | 60 | 45630.9 | 4508 | 10 | 10% | 2570.9 | 3862 | -1.96 % |
| 14 | | | | | | | 20 | 5% | 13892.3 | 10054 | -1.89 % |
| 15 | | | | | | | 5 | 40% | 15538.9 | 10502 | -1.05 % |

Table 5.5: Cooperation of aggregation method on m_range=[8 12] with benchmark data

We set various percentages and generate the coefficients of $d_j^s$ and $c_{i,j}^s$, for $s \in S$, as follows. For example, let $\epsilon = 10\%, 20\%$ or $40\%$, then

$$\frac{2(\max\limits_{s\in S} d_j^s - \min\limits_{s\in S} d_j^s)}{\max\limits_{s\in S} d_j^s + \min\limits_{s\in S} d_j^s} \leq \epsilon \text{ and } \frac{2(\max\limits_{s\in S} c_{i,j}^s d_j^s - \min\limits_{s\in S} c_{i,j}^s d_j^s)}{\max\limits_{s\in S} c_{i,j}^s d_j^s + \min\limits_{s\in S} c_{i,j}^s d_j^s} \leq \epsilon.$$

That means

$$\epsilon_1 = \frac{\epsilon\ (\max\limits_{s\in S} d_j^s + \min\limits_{s\in S} d_j^s)}{2} \text{ and } \epsilon_2 = \frac{\epsilon\ (\max\limits_{s\in S} c_{i,j}^s d_j^s + \min\limits_{s\in S} c_{i,j}^s d_j^s)}{2}$$

We use the $D^3$+PBDM to solve both the large problems and the aggregated problems. Here we test on two different ranges: [8 12] (Table 5.5) and [80 100] (Table 5.6), from which the limits of the multi-plants are generated.

The sizes of the large problems, “I”, “J” and “S” are listed on the left side of Table 5.5 and Table 5.6 followed by the CPU time, “Time”, and the number of B&B nodes, “N”, being searched by D3+PBDM. On the right side of tables, the number of aggregated scenarios, $S'$, and the various error percentages, $\epsilon$%, are listed, following by the CPU time, “Time”, and the number of B&B nodes, “N”, being searched by D3+PBDM while solving the corresponding aggregated problems. The last column, “$z_\epsilon$”, is the percentage of difference between both optimal solutions, as aforementioned.

| | | Large problems | | | $D^3$+PBDM | | Aggregated problems | | Aggregation method & $D^3$+PBDM | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | File | I | J | S | Time | N | $S'$ | $\epsilon$% | Time | N | $z_\epsilon$ |
| 1 | cap43 | 3 | 6 | 200 | 2056.0 | 316 | 10 | 10% | 25.7 | 78 | 2.99 % |
| 2 | | | | | | | 20 | 5% | 87.2 | 156 | 0.02 % |
| 3 | | | | | | | 5 | 40% | 19.1 | 162 | 15.24 % |
| 4 | cap41 | 3 | 10 | 100 | 967.3 | 330 | 10 | 10% | 48.3 | 146 | -0.29 % |
| 5 | | | | | | | 20 | 5% | 149.3 | 238 | 1.44 % |
| 6 | | | | | | | 5 | 40% | 22.4 | 134 | -3.00 % |
| 7 | cap81 | 4 | 20 | 120 | 41958.3 | 11310 | 10 | 10% | 47.2 | 208 | 1.94 % |
| 8 | | | | | | | 20 | 5% | 348.2 | 806 | 0.88 % |
| 9 | | | | | | | 5 | 40% | 35.7 | 316 | 1.94 % |
| 10 | cap64 | 5 | 30 | 150 | 2407.94 | 494 | 10 | 10% | 170.8 | 574 | 0.47 % |
| 11 | | | | | | | 20 | 5% | 86.4 | 136 | 1.22 % |
| 12 | | | | | | | 5 | 40% | 20.0 | 120 | 0.14 % |

Table 5.6: Cooperation of aggregation method on m_range=[80 100] with benchmark data

In Table 5.5, we notice that the CPU time can be saved up to 135 times and on average 17.5 times by using the aggregation method on the group of test problems with m_range in [8 12] and with the error percentage no more than 2.83%. In Table 5.6, we notice that the CPU times can be saved up to 1176 times and on average 219 times by using the aggregation method on the group of test problems with m_range in [8 12] and with the error percentage no more than 3.0% except a special case in test #3 with a gap of 15.2%.

From these tests in Table 5.5 and 5.6, we notice that the proposed aggregation method with the $D^3$+PBDM can efficiently solve SMpFLP problems that may otherwise be too large to be solved. Furthermore this method closely produces solutions that approximate the optimal solution within acceptable CPU times and error percentages.

### 5.6.6 Aggregation method with $D^3$+PBDM on SMpFLP with random data

In this section, we will also test the aggregation method using random data. We generate the data of the aggregated SMpFLP problems and the large-size SMpFLP problems as we did in Section 5.6.4 and Section 5.6.5, respectively. The computational tests of multi-plants in two different ranges: [8 12] and [80 100], are listed in Table 5.7 and Table 5.8, respectively.

In Table 5.7, with the aggregation method, the CPU times of PBDM can be saved up to 868 times and on average 155 times with the error percentage no more than 5.1%. In two groups of tests #7 – #9 and #16 – #18, the un-aggregated large-size models cannot be solved by the $D^3$+PBDM method within the 60,000 seconds (16.7 hours) time limit. In Table 5.8, the CPU times can be saved up to 12947 times and on average 1639 times using the aggregation method on this group of test problems with the error percentage no more than 2.28%.

In Table 5.7 and Table 5.8, we may note that using the aggregation method and the $D^3$+PBDM cannot only save CPU time by solving the aggregated problems with a reduced number of scenarios but also approximate optimal solutions very closely.

| | Large problems | | | $D^3$+PBDM | | Aggregated problems | | Aggregation method & $D^3$+PBDM | | |
|---|---|---|---|---|---|---|---|---|---|---|
| # | I | J | S | Time | N | $s'$ | $\epsilon\%$ | Time | N | $z_\epsilon$ |
| 1 | 5 | 40 | 100 | 789.4 | 212 | 10 | 10% | 4.7 | 18 | -2.11 % |
| 2 | | | | | | 20 | 5% | 22.3 | 38 | -2.87 % |
| 3 | | | | | | 5 | 40% | 2.6 | 22 | -2.11 % |
| 4 | 8 | 80 | 60 | 9358.9 | 3658 | 10 | 10% | 221.5 | 582 | 0.76 % |
| 5 | | | | | | 20 | 5% | 498.8 | 668 | 0.62 % |
| 6 | | | | | | 5 | 40% | 48.8 | 288 | 1.06 % |
| 7 | 16 | 100 | 60 | 60607.6 | 13526 | 10 | 10% | 313.6 | 452 | 4.50 % |
| 8 | | | | | | 20 | 5% | 962.5 | 652 | 4.70 % |
| 9 | | | | | | 5 | 40% | 69.8 | 226 | 5.08 % |
| 10 | 20 | 80 | 60 | 149.7 | 52 | 10 | 10% | 32.8 | 48 | -0.18 % |
| 11 | | | | | | 20 | 5% | 79.3 | 58 | 0.01 % |
| 12 | | | | | | 5 | 40% | 23.1 | 52 | 0.01 % |
| 13 | 30 | 50 | 80 | 851.6 | 148 | 10 | 10% | 25.6 | 60 | -1.06 % |
| 14 | | | | | | 20 | 5% | 55.0 | 62 | -1.17 % |
| 15 | | | | | | 5 | 40% | 11.5 | 54 | -1.16 % |
| 16 | 50 | 100 | 80 | 60602.2 | 3798 | 10 | 10% | 491.16 | 696 | 2.48 % |
| 17 | | | | | | 20 | 5% | 872.98 | 516 | 2.73 % |
| 18 | | | | | | 5 | 40% | 104.47 | 208 | 2.51 % |

Table 5.7: Cooperation of aggregation method on m_plant=[8 12] with random data

## 5.7 Conclusion

In this chapter, we consider the stochastic version of multi-plant facility location problems. In the formulation, the location decision variables in both the first and second stage are general integer (as opposed to binary), which makes the problems even more difficult to solve.

In the original SMpFLP, we relax aggre-shipping constraints by pathway constraints and total-capacity constraints. Then the relaxed SMpFLP is decomposed by using our $D^3$ method as described in Chapter 3. Each scenario subproblem is a mixed integer program and is difficult to be solved by an commercial MIP solver directly or by the conventional BDM. We follow the framework of BDM to decompose the scenario sub-

| | Large problems | | | $D^3$+PBDM | | Aggregated problems | | Aggregation method & $D^3$+PBDM | | |
|---|---|---|---|---|---|---|---|---|---|---|
| # | I | J | S | Time | N | $S'$ | $\epsilon\%$ | Time | N | $z_\epsilon$ |
| 1 | 4 | 50 | 120 | 16961.0 | 2484 | 10 | 10% | 2.1 | 14 | -0.90 % |
| 2 | | | | | | 20 | 5% | 8.5 | 30 | -0.52 % |
| 3 | | | | | | 5 | 40% | 1.3 | 16 | -0.37 % |
| 4 | 5 | 40 | 100 | 2293.1 | 1260 | 10 | 10% | 37.8 | 158 | 0.18 % |
| 5 | | | | | | 20 | 5% | 73.7 | 148 | 0.11 % |
| 6 | | | | | | 5 | 40% | 21.3 | 160 | 0.20 % |
| 7 | 8 | 80 | 60 | 41400.2 | 19777 | 10 | 10% | 255.47 | 1508 | -0.89 % |
| 8 | | | | | | 20 | 5% | 232.8 | 744 | -0.95 % |
| 9 | | | | | | 5 | 40% | 107.17 | 1376 | -1.00 % |
| 10 | 10 | 100 | 60 | 25800.8 | 22038 | 10 | 10% | 162.0 | 852 | 0.84 % |
| 11 | | | | | | 20 | 5% | 1503.0 | 5064 | 0.69 % |
| 12 | | | | | | 5 | 40% | 74.0 | 920 | 0.98 % |
| 13 | 20 | 30 | 60 | 38468.6 | 19070 | 10 | 10% | 5114.8 | 32958 | 0.97 % |
| 14 | | | | | | 20 | 5% | 2886.3 | 9392 | 2.28 % |
| 15 | | | | | | 5 | 40% | 316.4 | 3910 | 1.34 % |

Table 5.8: Cooperation of Aggregation method on m_plant=[80 100] with random data

problem into a Benders Dual subproblem with only continuous serving variables and a Benders Master Problem with all integer location variables. The Benders Dual subproblem is further decomposed for each client as the Benders Client subproblem due to the structure of the relaxed SMpFLP. We develop a heuristic algorithm to solve the Client subproblem directly. In order to avoid degenerate cuts created after solving the Benders Dual subproblem in the CBDM and to accelerate the computation in solving the Benders Master problem, we advance the algorithm by generating Pareto-optimal cut. After the developed PBDM has converged, we solve an LP if the optimal solution of the scenario subproblem does not satisfy the replaced aggre-shipping constraints, in order that the proposed $D^3$+PBDM can obtain a good feasible solution of the original SMpFLP. Also the aggregation method is also proposed and implemented to aggregate scenarios in large-scale problems when solving Benders Dual subproblems.

The proposed methods are tested on both benchmark data and randomly generated data. The $D^3$+PBDM solves moderate and large sized instances of SMpFLP, which can not be solved or solved much slower by the CPLEX MIP solver. The computational comparisons also show that the aggregation method can reduce the number of scenarios in large problems and obtain approximately optimal solutions very quickly.

# Chapter 6

# Conclusion and Future Research

## 6.1 Conclusion

In the thesis, we have developed a scenario-wise decomposition method, Dynamic Dual Decomposition method ($D^3$ method), which is based on the Dual Decomposition method (Carøe and Schultz (1999)) of Stochastic Mixed Integer Programming. When we implement the Dual Decomposition method in solving the large-scale Stochastic Uncapacitated Facility Location Problems (SUFLP), we observe that the algorithm cannot efficiently converge in the process of branch-and-bound. After examining different representations of non-anticipativity and incorporating other methods into the whole algorithm, we constructed an exact method, $D^3$ method, to efficiently solve moderate and large sized instances of SUFLP, which cannot be solved or solved much slower by the state-of-the-art commercial MIP solver. We also modeled the three-stage SUFLP and solved it by the proposed method.

Using the developed $D^3$ method as the framework, we modeled the Stochastic Set Packing Problem and decomposed it into scenario subproblems that are similar to the deterministic set packing problem. Then we used a refined column generation method to solve the decomposed scenario subproblems. Our method performed better than solving the whole problem using the state-of-the-art commercial solver or by implementing the conventional column generation method directly in most cases. A sensitivity analysis on various densities of patterns showed that the proposed method is more robust than CPLEX IP solver.

Lastly, we combined our $D^3$ method and Benders Decomposition to solve the Stochastic version of the Multi-plant Facility Location Problem (SMpFLP). In our model, the location decision variables in both the first and the second stage are general integer (as opposed to binary), which causes the problem to be difficult to solve. After decomposing the whole model by using the $D^3$ method, we further decomposed the scenario subproblems into two parts: the Benders Dual subproblem with only continuous serving variables and the Benders Master problem with all integer facility location variables. In order to avoid degenerate cuts which are often encountered in solving the master problem by the conventional Benders Decomposition method, we developed an algorithm of generating Pareto-optimal cuts that was developed for single-plant problems to multi-plant location problems. We also studied and implemented an aggregation method to aggregate scenarios when solving Benders Dual subproblems. Computational tests on both benchmark data and our randomly generated data showed that the proposed method can solve large-scale SMpFLP problems efficiently, which cannot be directly solved or solved slower by the state-of-the-art commercial solver. The computational comparisons also showed that the aggregation method can reduce the number of scenarios in large-scale problems and obtain approximately optimal solutions efficiently.

Overall, in this thesis, we have demonstrated a decomposition-based method which can solve a broad group of stochastic integer problems more efficiently. Many industrial applications with uncertainty can be modeled as two-stage stochastic recourse models with first-stage integer variables. But large-scale models of this type of problems are hard to solve for a global optimum due to both stochastic and integer natures. The proposed method provides a framework to decompose the whole model scenario-wise into subproblems and combine other tailored methods to solve scenario subproblems. The computational experiments showed our methods can solve large-scale models more efficiently, which cannot be directly solved by the state-of-the-art commercial solver.

## 6.2 Future Research

In Chapter 2, we briefly noted some aspects of this research that remain for future work. Some of this work has been started, but not reported in this thesis.

### 6.2.1 Multi-stage SMIP

Although some decomposition methods on two-stage SMIP can be theoretically extended to the multi-stage cases, scalability is a potential issue. In Chapter 3, we applied the proposed $D^3$ method to decompose the three-stage SUFLP since the proposed method can provide a framework to combine other specific methods to solve scenario subproblems. However we did not conduct a deep study or develop a specific method for more general multi-stage stochastic programming like the branch-and-fix method. Therefore, future work should consider the inclusion of more stages combining ideas from the branch-and-fix method and other decomposition methods.

### 6.2.2 Stochastic Scheduling Programming

Constraint Programming (CP) is an alternative and popular method to solve complex scheduling problems. We have studied and modeled the Stochastic Job-shop Problem as a two-stage stochastic scheduling problem with recourse. The first stage variables and constraints are modeled in the formulation of a mathematical program but the second stage parts are modeled as a CP. Future work should consider using the $D^3$ method to decompose the model into scenario subproblems and invoking both a MIP solver and a CP solver to iteratively solve the first stage and the second stage subproblems, respectively.

### 6.2.3 Parallel Implementation

With advances in coordination of multi-core systems and synchronization of parallel computational power, the limits of serial processing on solving large-scale problems have been relieved tremendously.

When we were studying and implementing the Dual Decomposition method, we planned to advance the algorithm to a parallel computing framework as one of possible research directions since we would have a multi-core parallel system available. Some computing designs on how to distribute the algorithms and coordinate the computations on the parallel multi-core system have been drafted in pseudo-code. After we found the inefficiency in solving large-scale SUFLP and developed the algorithm in several aspects that have been presented in Chapter 3, we also further studied and designed on how to implement the proposed $D^3$ method in the parallel computer system. Based on our preliminary study on parallel computing, the proposed methods and developed algorithms in this thesis could also be used as the basis for future implementation on parallel or distributed computing platforms or in a cloud computing environment.

# Appendix A

# Some Tables of Model Sizes

| Experiment | | $D^3$ | | | | DEP | | | |
|---|---|---|---|---|---|---|---|---|---|
| # | Problems | Variables | | | Constraint | Variables | | | Constraint |
| | | Total | Binary | Integer | | Total | Binary | Integer | |
| 1 | 4F_5P_40S | 28 | 8 | 20 | 25 | 964 | 164 | 800 | 1000 |
| 2 | 4F_5P_80S | 28 | 8 | 20 | 25 | 1924 | 324 | 1600 | 2000 |
| 3 | 10F_50P_50S | 520 | 20 | 500 | 550 | 25510 | 510 | 25000 | 27500 |
| 4 | 10F_50P_100S | 520 | 20 | 500 | 550 | 51010 | 1010 | 50000 | 55000 |
| 5 | 15F_45P_15S | 705 | 30 | 675 | 720 | 10365 | 240 | 10125 | 10800 |
| 6 | 15F_45P_20S | 705 | 30 | 675 | 720 | 13815 | 315 | 13500 | 14400 |
| 7 | 15F_45P_25S | 705 | 30 | 675 | 720 | 17265 | 390 | 16875 | 18000 |
| 8 | 15F_45P_30S | 705 | 30 | 675 | 720 | 20715 | 465 | 20250 | 21600 |
| 9 | 15F_45P_50S | 705 | 30 | 675 | 720 | 34515 | 765 | 33750 | 36000 |
| 10 | 15F_45P_100P | 705 | 30 | 675 | 720 | 69015 | 1515 | 67500 | 72000 |
| 11 | 15F_50P_50P | 780 | 30 | 750 | 800 | 38265 | 765 | 37500 | 40000 |
| 12 | 15F_100P_100S | 1530 | 30 | 1500 | 1600 | 151515 | 1515 | 150000 | 160000 |
| 13 | 20F_50P_50S | 1040 | 40 | 1000 | 1050 | 51020 | 1020 | 50000 | 52500 |
| 14 | 20F_50P_100S | 1040 | 40 | 1000 | 1050 | 102020 | 2020 | 100000 | 105000 |
| 15 | 30F_50P_50S | 1560 | 60 | 1500 | 1550 | 76530 | 1530 | 75000 | 77500 |

Table A.1: Sizes of the test instances

For example, experiment 1 is a moderate sized problem with 4 potential facility locations (4F), 5 clients (5P) and 40 scenarios (40S).

| Experiment | | $D^3$ | | | | DEP | | | |
|---|---|---|---|---|---|---|---|---|---|
| # | Problems | Variables | | | Constraint | Variables | | | Constraint |
| | | Total | Binary | Integer | | Total | Binary | Integer | |
| 1 | 3F_4P_10S_2R | 48 | 12 | 36 | 48 | 453 | 93 | 360 | 480 |
| 2 | 3F_4P_10S_4R | 78 | 18 | 60 | 80 | 753 | 153 | 600 | 800 |
| 3 | 3F_4P_10S_10R | 168 | 36 | 132 | 176 | 1653 | 333 | 1320 | 1760 |
| 4 | 4F_8P_40S_5R | 220 | 28 | 192 | 240 | 8644 | 964 | 7680 | 9600 |
| 5 | 4F_8P_40S_10R | 400 | 48 | 352 | 440 | 15844 | 1764 | 14080 | 17600 |
| 6 | 3F_4P_50S_20R | 318 | 66 | 252 | 336 | 15753 | 3153 | 12600 | 16800 |
| 7 | 5F_10P_40S_4R | 610 | 60 | 550 | 660 | 24205 | 2205 | 22000 | 26400 |
| 8 | 5F_10P_20S_5R | 335 | 35 | 300 | 360 | 6605 | 605 | 6000 | 7200 |
| 9 | 5F_10P_20S_10R | 610 | 60 | 550 | 660 | 12105 | 1105 | 11000 | 13200 |
| 10 | 10F_10P_20S_5R | 670 | 70 | 600 | 660 | 13210 | 1210 | 12000 | 13210 |
| 11 | 5F_20P_20S_10R | 1160 | 60 | 1100 | 1320 | 23105 | 1105 | 22000 | 26400 |
| 12 | 5F_10P_50S_40R | 2260 | 210 | 2050 | 2460 | 112755 | 10255 | 102500 | 123000 |
| 13 | 3F_5P_100S_50R | 921 | 156 | 765 | 1020 | 91803 | 15303 | 76500 | 102000 |
| 14 | 5F_20P_100S_20R | 2210 | 110 | 2100 | 2520 | 220505 | 10505 | 210000 | 252000 |
| 15 | 5F_20P_200S_40R | 4310 | 210 | 4100 | 4920 | 861005 | 41005 | 820000 | 984000 |
| 16 | 10F_10P_400S_4R | 560 | 60 | 500 | 550 | 220010 | 20010 | 200000 | 220000 |
| 17 | 3F_10P_400S_20R | 696 | 66 | 630 | 840 | 277203 | 25203 | 252000 | 336000 |
| 18 | 3F_20P_500S_10R | 696 | 36 | 660 | 880 | 346503 | 16503 | 330000 | 440000 |
| 19 | 5F_20P_400S_4R | 530 | 30 | 500 | 600 | 210005 | 10005 | 200000 | 240000 |
| 20 | 5F_10P_500S_5R | 335 | 35 | 300 | 360 | 165005 | 15005 | 150000 | 180000 |
| 21 | 20F_20P_100S_5R | 2540 | 140 | 2400 | 2520 | 252020 | 12020 | 240000 | 252000 |

Table A.2: Sizes of three-stage SUFLP instances

# Bibliography

S. Ahmed, M. Tawarmalani, N. V. Sahinidis (2004): A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100, 355-377.

S. Ahmed, A. Shapiro (2002): The Sample Average Approximation Method for Stochastic Programs with Integer Recourse. *ISyE Technical Report*.

S. Ahmed (2013): A scenario decomposition algorithm for 0-1 stochastic programs. *Operations Research Letters*, 41, 565-569.

A. Alonso-Ayuso, L. F. Escudero, M. T. Ortuno (2003): BFC, A branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0-1 programs. *European Journal of Operational Research*, 151, 503-519.

R. Andrade, A. Lisser, N. Maculan, G. Plateau (2004): Telecommunication Network Capacity Design for Uncertain Demand. *Computational Optimization and Applications*, 29(2), 127-146.

R. Andrade, A. Lisser, N. Maculan, G. Plateau (2005): B&B Frameworks for the Capacity Expansion of High Speed Telecommunication Networks Under Uncertainty. *Annals of Operations Research*, 140, 49-65.

E. Balas, E. S. Ceria, G. Cornuejols (1993): A lift-and-project cutting plane algorithm for mixed 0-1 integer programs. *Mathematical Programming*, 58, 295-324.

E. Balas, M. W. Padberg (1976): Set partitioning: A survey. *SIAM Review*, 18(4), 710-760. Springer-Verlag, New York-Berlin.

J. Balachandran, S. Jain (1976): Optimal facility location under random demand with general cost structure. *Naval Research Logistics Quarterly*, 23, 421-436.

J. E. Beasley (1990): OR-Library: Distributing Test Problems by Electronic Mail. *Journal of the Operational Research Society*, 41(11), 1069-1072.

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

J. Bidot, T. Vidal, P. Laborie, J.C. Beck (2009): A Theoretic and Practical Framework for Scheduling in a Stochastic Environment, *Journal of Scheduling*, 12, 315-344.

J. R. Birge (1985): Aggregation Bounds in Stochastic Linear Programming. *Mathematical Programming*, 31(1), 25-41.

J. R. Birge, F. Louveaux (1997): *Introduction to Stochastic Programming.* Springer-Verlag, New York-Berlin.

T. B. Boffey (1989): Location Problems Arising in Computer Networks. *Journal of the Operational Research Society*, 40(4), 347-354.

C. C. Carøe, R. Schultz (1999): Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24, 37-45.

P. C. Chu, J. E. Beasley (1997): A Genetic Algorithm for the Generalized Assignment Problem. *Computers and operations Research*, 24, 17-23.

T. G. Crainic, X. Fu, M. Gendreau, W. Rei, W. Stein (2011): Progressive hedging-based metaheuristics for stochastic network design. *Networks*, 58, 114-124.

C. B. Dean (2005): Approximation algorithms for stochastic scheduling problems. *Ph.D. Thesis, Miassachusetts Institute of Technology, Mass, USA.*

C. B. Dean, M. X. Goemans, J. Vondrak (2005): Adaptivity and approximation for stochastic packing problems. *Proceedings of the 16th annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 395-404.

C. B. Dean, M. X. Goemans, J. Vondrak (2008): Approximating the Stochastic Knapsack Problem: The Benefit of Adaptivity. *Mathematics of Operations Research*, 33, 945-964.

S. Engell, A. Markert, G. Sand, R. Schultz (2004): Aggregated Scheduling of a Multi-product Batch Plant by Two-Stage Stochastic Integer Programming. *Optimization and Engineering*, 5, 335-359.

L.F. Escudero, M. A. Garin, G Perez, A. Unzueta (2013): Scenario Cluster Decomposition of the Lagrangian dual in two-stage stochastic mixed 0-1 optimization. *Computers & Operations Research*, 40, 362-377.

L.F. Escudero, M. Landete, A.M. Rodriguez-Chia (2011): Stochastic Set Packing Problem. *European Journal of Operational Research*, 211, 232-240.

M.M. Fazel-Zarandi, O. Berman, J.C. Beck (2012): Solving a Stochastic Facility LocationFleet Management Problem with Logic-Based Benders Decomposition, *IIE Transactions*, 45(8), 896-911.

M.M. Fazel-Zarandi, J.C. Beck (2012): Using Logic-Based Benders Decomposition to Solve the Capacity and Distance Constrained Plant Location Problem, *INFORMS Journal on Computing*, 24, 399-415.

L. M. Fisher (1981): The Lagrangian Relaxation Method for Solving Integer Programming. *Management Science*, 27, 1-18.

R. M. Freund (2004): Benders' Decomposition Methods for Structured Optimization, including Stochastic Optimization. *Lecture note*, MIT.

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

G. Ghiani, F. Guerriero, R. Musmanno (2002): The capacitated plant location problem with multiple facilities in the same site. *Computers and Operations Research*, 29, 1903-1912.

G. Ghiani, L. Grandinetti, F. Guerriero, R. Musmanno (2002): A lagrangean heurisic for the plant location problem with multiple facilities in the same site. *Optimization Methods and Software*, Vol 17, No. 6, 1059-1076.

P. C. Gilmore, R. E. Gomory (1961): A Linear Programming Approach to the Cutting Stock Problem. *Operations Research*, 9, 849-859 Sons, Chichester.

M. Guignard, K. Spielberg (1979): A Direct Dual Method for the Mixed Plant Location Problem with Some Side Constraints. *Mathematical Programming*, 17 198-228.

J.L. Higle, B. Rayco, S. Sen. Stochastic scenario decomposition for multi-stage stochastic programs. www.sie.arizona.edu/faculty/higle, 2004.

K. Huang (2005): Multi-stage Stochastic Programming Models in Production Planning. *Ph.D Thesis at the Georgia Institute of Technology.*

H. Jonsson, E. A. Silver (1996): Some insights regarding selecting sets of scenarios in combinatorial stochastic problems. *International Journal of Production Economics*, 45, 463-472.

K. C. Kiwiel (1990): Proximity Control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, 46, 105-122.

P. Kall, S. Wallace (1994): *Stochastic Programming.* John Wiley and Sons, Chichester.

W. K. Klein Haneveld, M. H. van der Vlerk (1999): Stochastic integer programming: General models and algorithms. *Annal of Operations Research*, 85, 39-57.

A. Klose, A. Drexl (2005): Facility location models for distribution system design. *European Journal of Operational Research*, 162, 4-29.

R. H. Kwon, G. V. Dalakouras, C. Wang (2008): On a posterior evaluation of a simple greedy method for set packing. *Optimization Letters*, 2, 587-597.

A. Lokketangen, D. L. Woodruff (1996): Progressive hedging and tabu search applied to mixed integer 0-1 multistage stochastic programming. *Journal of Heuristics*, 2, 111-128.

F. V. Louveaux, D. Peeters (1992): A dual-based procedure for stochastic facility location. *Operations Research*, 40(3), 564-573.

G. Laporte, F. V. Louveaux (1993): The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3), 133-142.

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

J. M. Mulvey, A. Ruszczynski (1995): A new scenario decomposition method for large-scale stochastic optimization. *Operations Research*, 43, 477-490.

A. Nemirovski, A. Shapiro (2006): Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17:969-996.

M. Novak, R. Schultz (2005): A Stochastic Integer Programming Model for Incorporating Day-Ahead Trading of Electricity into Hydro-Thermal Unit Commitment. *Optimization and Engineering*, 6, 163-176.

L. Ntaimo (2004): Decomposition Algorithms for Stochastic Combinatorial Optimization: Computational Experiments and Extensions. *Ph.D Thesis at the University of Arizona.*

L. Ntaimo, S. Sen (2008): A comparative study of decomposition algorithms for stochastic combinatorial optimization. *Computational Optimization and Applications*, 40, 299-319.

S. Owens, M. Daskin (1998): Strategic facility location: A review. *European Journal of Operational Research*, 111, 423-447.

A. Prekopa (2003): Probabilistic programming. In A. Ruszczynski and A. Shapiro, editors, *Stochastic Programming*, volume 10 of Handbooks in Operations Research and Management Science, 267-351.

R. Ravi, A. Sinha (2006): Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Mathematical Programming A*, 108, 97-114.

R. T. Rockafellar, R. J.-B. Wets (1991): Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16, 119-147.

N. V. Sahinidis (2004): Optimization under uncertainty: state-of-the-art and opportunities. *Computers and Chemical Engineering*, 28, 971-983.

T. Santoso ,S. Ahmed , M. Goetschalckx, A. Shapiro (2005): A stochastic programming approach for supply chain network design under uncertainty. *European Journal of Operational Research*, 167, 96-115.

P. Schütz, L. Stougie, A. Tomasgard (2008): Stochastic facility location with general long-run costs and convex short-run costs. *Computers and Operations Research*, 35, 2988-3000.

R. Schultz (2003): Stochastic programming with integer variables. *Mathematical Programming*, 97, 285-309.

R. Schultz, L. Stougie, M. H. vander Vlerk (1998): Solving stochastic programs with integer recourse by enumeration: a framework using Grobner basis reductions. *Math. Programming*, 83, 229-252.

S. Sen, J. L. Higle (2005): The $C^3$ Theorem and a $D^2$ Algorithm for Large Scale Stochastic Mixed-Integer Programming: Set Convexification. *Mathematical Programming*, 104(1), 1-20.

S. Sen, H. D. Sherali (2006): Decomposition with Branch-and-Cut Approaches for Two-Stage Stochastic Mixed-Integer Programming. *Mathematical Programming*, 106, 203-223.

H. Sherali, B. Fraticelli (2002): A modification of Benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22, 319-3472.

T. Shiina, J. R. Birge (2004): Stochastic unit commitment problem. *International Transactions in Operational Research*, 11 19-32.

A. Sinha (2004): *Location, Location, Location and Location-Facility location incorporating demand uncertainty, logistic network design, product heterogeneity and competition.* Ph.D dissertation, Carnegie Mellon University.

E. F. Silva (2004): *Improving Branch-and-price Algorithms and Applying Them to Stochastic Programs.* Ph.D dissertation, Naval Postgraduate School.

L. V. Snyder (2005): Facility Location Under Uncertainty: A Review. *Technical Report*, 04T-015, Lehigh University, PA, USA.

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

P. H. Vance, C. Barnhart, E. L. Johnson, G. L. Nemhauser (1993): Solving Binary Cutting Stock Problems by Column Generation and Branch-and-Bound. *Computational Optimization and Applications*, 3, 111-130.

J. -P. Watson, D. L. Woodruff (2010): Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 1-16.

P. Wentges (1996): Accelerating Benders' Decomposition for the Capacitated Facility Location Problem. *Mathematical Methods of Operations Research*, 44 : 267-290.

L. Wu, X. Zhang, J. Zhang (2006): Capacitated facility location problem with general setup cost. *Computers and Operations Research*, 33, 1226-1241.

X. M. Zhu (2006): Discrete Two-Stage Stochastic Mixed-Integer Programs with Applications to Airline Fleet Assignment and Workforce Planning Problems. *Ph.D Thesis at the Virginia Polytechnic Institute and State University.*