DECOMPOSITION ALGORITHMS FOR STOCHASTIC COMBINATORIAL OPTIMIZATION: COMPUTATIONAL EXPERIMENTS AND EXTENSIONS

by

Lewis Ntaimo

A Dissertation Submitted to the Faculty of the
DEPARTMENT OF SYSTEMS AND INDUSTRIAL ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of
DOCTOR OF PHILOSOPHY
In the Graduate College
THE UNIVERSITY OF ARIZONA

2004

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

The University of Arizona ®
Graduate College

As members of the Final Examination Committee, we certify that we have read the dissertation prepared by Lewis Ntaimo

entitled Decomposition Algorithms for Stochastic Combinatorial Optimization: Computational Experiments and Extensions

and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor Philosophy

Suvrajeet Sen	7/15/04 date
Julia L. Higle	7/15/04 date
J. Cole Smith	7/15/04 date
	date
	date

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Suvrajeet Sen 8/11/04
Dissertation Director: date

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the copyright holder.

SIGNED: LEWIS NTAIMO

ACKNOWLEDGEMENTS

First and foremost, I thank the Lord God Almighty for allowing and enabling me to complete my Ph.D. studies. I am greatly thankful to my advisor Dr. Suvrajeet Sen for his vision, direction, advice and financial support throughout my research.

I thank Dr. Julia L. Higle, Dr. Jonathan C. Smith, Dr. Salim A. Hariri and Dr. Bernard P. Zeigler, for their willingness to serve on my committee. I extend my heartfelt thanks to Dr. Fei-Yue Wang who taught me courses in robotics and automation and intelligent control systems and applications during my M.S. studies in Mining and Geological Engineering. He is the one who persuaded me to consider SIE for my Ph.D. studies after I completed my M.S studies.

I give special thanks to the department head Dr. Ronald G. Askin for all his support and all the members of staff in the SIE department, especially Myra Forchione, Linda Cramer, Meg Rosenquist and Bill Ganoe, for his valuable assistance with any issues I had with the computers in the department. I also acknowledge my fellow colleagues for their help and encouragement: Yijia Xu, Lihua Yu, Guglielmo Lulli, Enock C. Mofya, Josephat M. Zimba, and Bwalya Malama.

Finally, I would like to thank Dr. Bob Bixby for confirming our experience with CPLEX and guiding the choice of parameter settings; Dr. Bob Doverspike (AT&T Research), Dr. Madhav Marathe (Los Alamos National Labs), Dr. David Morton (Univ. of Texas, Austin) and Dr. Maarten van der Vlerk (Univ. of Groningen, Netherlands) for their useful comments on an earlier version of Chapter 5; Dr. Antonio Alonso-Ayuso and Dr. L. F. Escudero for providing the stochastic SSCh problem instances and Nan Kong and Dr. Andrew J. Schaefer for providing the stochastic matching problem instances, and for confirming our computational results.

This research was funded by grants from the OR Program (DMII-9978780) and the Next Generation Software Program (CISE-9975050) of NSF.

DEDICATION

To my wife Chloe Agnes Ntaimo, and our lovely children Joseph Mulenga Ntaimo and Claire Mumba Ntaimo, both who were born during my graduate studies, and Caelin Joshua Balfour. And to my father Laso Ntaimo, my mother-in-law Claire Lewis Balfour, my sister-in-law Charlene Bernedette Balfour, and all our family.

Your love, understanding, support and prayers propelled me to complete this long march.

TABLE OF CONTENTS

TABLE OF CONTENTS — *Continued*

TABLE OF CONTENTS — *Continued*

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Some of the most important and challenging problems in computer science and operations research are stochastic combinatorial optimization (SCO) problems. SCO deals with a class of combinatorial optimization models and algorithms in which some of the data are subject to significant uncertainty and evolve over time, and often discrete decisions need to be made before observing complete future data. Therefore, under such circumstances it becomes necessary to develop models and algorithms in which plans are evaluated against possible future scenarios that represent alternative outcomes of data. Consequently, SCO models are characterized by a large number of scenarios, discrete decision variables and constraints.

This dissertation focuses on computational experimentation with practical decomposition algorithms for large-scale SCO. Stochastic mixed-integer programming (SMIP), the optimization branch concerned with models containing discrete decision variables and random parameters, provides one way for dealing with such decision-making problems under uncertainty. This dissertation studies decomposition algorithms, models and applications for large-scale two-stage SMIP. The theoretical underpinnings of the method are derived from the disjunctive decomposition (D^2) method. We study this class of methods through applications, computations and extensions.

With regard to applications, we first present a stochastic server location problem (SSLP) which arises in a variety of applications. These models give rise to SMIP problems in which all integer variables are binary. We study the performance

of the D^2 method with these problems. In order to carry out a more comprehensive study of SSLP problems, we also present certain other valid inequalities for SMIP problems.

Following our study with SSLP, we also discuss the implementation of the D^2 method, and also study its performance on problems in which the second-stage is mixed-integer (binary). The models for which we carry out this experimental study have appeared in the literature as stochastic matching problems, and stochastic strategic supply chain planning problems. Finally, in terms of extensions of the D^2 method, we also present a new procedure in which the first-stage model is allowed to include continuous variables. We conclude this dissertation with several ideas for future research.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Combinatorial optimization problems are among some of the most important and challenging problems in computer science and operations research (Cook et al., 1998). These optimization problems are characterized by discrete-choice variables. Stochastic combinatorial optimization (SCO) deals with a class of combinatorial optimization models and algorithms in which some of the data are subject to significant uncertainty. Such models are characterized by data that evolve over time and often decisions need to be made before observing complete future data. Therefore, under such circumstances it becomes necessary to develop models in which plans are evaluated against possible future scenarios that represent alternative outcomes of data. Consequently, SCO models are of a large-scale nature and are characterized by a large number of decision variables and constraints.

Stochastic programming (SP), the optimization branch concerned with models containing random parameters, provides one way for dealing with such decision-making problems under uncertainty. In particular, stochastic linear programming (SLP) deals with SP models with continuous decision variables, while stochastic mixed-integer programming (SMIP) deals with SCO models in which some of the decision variables are required to be continuous, discrete, or mixed. SMIP is a very young field which has a lot of applications in both science and engineering. SP models arise for example in telecommunication (Sen et al.,

1994), transportation (Powell, 1988), finance (Carinõ et al., 1994), electricity power generation (Carøe and Schultz, 1998), manufacturing (Eppen et al., 1989), and the military (Morton et al., 1996; Baker et al., 2002). Deterministic models often lead to myopic decisions under uncertainty that can result in significant losses. SP models on the other hand take into account the possible future outcomes and thus hedge against unforseen potential losses. Several application cases have been reported in the literature where SP models have indeed resulted in better decision making and significant savings in profits.

SMIP models generally arise whenever deterministic IP models result in inadequate models under uncertainty. In the general case integer decisions have to be made both before and after observing the outcomes of the random variables. For example, strategic decisions about production topology and plant sizing in strategic supply chain management under uncertainty have to be made prior to the final product price and demand realizations at different markets, while operational or tactical decisions such as scheduling, and raw material volume supply from vendors are generally made after price and demand realizations.

Despite the large number of applications that lead to SMIP models, very few practical algorithms have been developed to date. In fact, computational results and algorithms are fairly scant. One may attribute this to the fact that integer programming (IP) is generally NP-hard. But in addition to inheriting the properties of IP models, SMIP models are generally of a large-scale nature due to the uncertainty in the problem data. Therefore, SMIP models present formidable algorithmic challenges. Indeed, this calls for novel decomposition algorithms that are both scalable and practical.

The objective of this dissertation is to contribute to tackling the aforementioned challenges of SMIP through new models, algorithms, and computational experiments. This dissertation is devoted to all three aspects.

1.2 Problem Statement

Throughout this dissertation we consider the following general two-stage SMIP problem:

$$\underset{x \in X \cap \mathbf{X}}{\text{Min}}\ c^\top x + E[f(x, \tilde{\omega})], \tag{1.1}$$

where c is a known vector in $\Re^{n_1}$, $X \subseteq \Re^{n_1}$ is a set of feasible first-stage decisions and $\mathbf{X}$ define restrictions requiring some first-stage decision variables to be integer, $E[.]$ is the usual mathematical expectation operator with

$$E[f(x, \tilde{\omega})] = \sum_{\omega \in \Omega} p_\omega f(x, \omega),$$

$\tilde{\omega}$ is a multi-variate discrete random variable with a realization (scenario) ω with probability p_ω and sample space Ω, and for any ω

$$f(x, \omega) = \text{Min}\ q(\omega)^\top y, \tag{1.2a}$$

$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x, \tag{1.2b}$$

$$y \geq 0,\ y_j \text{ integer},\ j \in J_2. \tag{1.2c}$$

In problem formulation (1.2), $q(\omega)$ is the cost vector in $\Re^{n_2}$ for scenario $\omega \in \Omega$ and J_2 is an index set that may include some or all the variables listed in $y \in \Re^{n_2}$. Although the second-stage (recourse) variable y depends on the outcome ω, this dependence is not explicitly indicated here. This is because the subproblem for each outcome ω is decoupled from all other outcomes once a vector x is given. Thus this formulation emphasizes the loosely coupled nature of two-stage SMIP problems.

In this dissertation we address instances of problem (1.1-1.2) under the following assumptions:

(A1) Ω is a finite set.

(A2) X is a closed set and is defined as $X = \{x \in \Re^{n_1}_+ \mid Ax \geq b\}$.

(A3) $f(x,\omega) < \infty$ for all $(x,\omega) \in X \times \Omega$.

Assumption (A3) requires that the subproblem (1.2) remain feasible for all $(x,\omega) \in X \times \Omega$ and is referred to as relatively complete (integer) recourse (Wets, 1974).

Since we assume that the problem data is governed by discrete random variables, the formulation (1.1-1.2) can also be written as the so called *deterministic equivalent problem* (DEP) formulation or *extensive form* as follows:

$$\underset{x \in X \cap \mathbf{X}}{\text{Min}}\ c^\top x + \sum_{\omega \in \Omega} p_\omega q(\omega)^\top y^\omega \tag{1.3a}$$

$$\text{s.t. } T(\omega)x + Wy^\omega \geq r(\omega),\ \forall \omega \in \Omega, \tag{1.3b}$$

$$y^\omega \geq 0,\ y_j^\omega \text{ integer},\ j \in J_2,\ \forall \omega \in \Omega. \tag{1.3c}$$

Problem (1.3) is a large-scale MIP formulation and potentially can be solved by an MIP solver directly. However, in order to adequately capture the uncertainty in the problem, the number of scenarios $|\Omega|$ is generally large. Therefore, problem (1.3) may become intractable even for the state-of-the-art commercial MIP solvers. Also note that the dependence of the second-stage decision on the scenario is now explicitly made in the DEP formulation.

1.3 Research Scope and Approach

This dissertation deals with decomposition algorithms for SCO. The starting point for this dissertation is the Disjunctive Decomposition (D^2) algorithm proposed in (Sen and Higle, 2000). The aim of the dissertation is to investigate the potential of the D^2 method through computational studies, and extend its realm of applications by studying a wider class of problems. Accordingly, we begin by reviewing the underlying theory, and illustrate the concepts through a simple example problem. Computer implementation of the D^2 method is discussed and the associated

efficiency and convergence issues discussed. A computational study of the method is carried out through application to different large-scale SCO models and finally, extensions to the current D^2 theory are made.

This dissertation is oriented towards modeling and computation and its contributions will be in the following aspects:

1. Computer implementation of the D^2 algorithm and the identification of the issues associated such an implementation. These issues will potentially translate to future algorithms that follow a similar approach.

2. Proposing a new model for server location under uncertainty (the SSLP) with potential use in a variety of application domains. Conducting a computational study of the SSLP and demonstrating that significant gains can be made by the application of the D^2 method to SSLP. As a by-product of this experiment, we have developed SSLP test problems that can be used to test the performance of other algorithms. These test problems will be made available via SIPLIB at http://www.isye.gatech.edu/~sahmed/siplib/.

3. Solving some of the largest SCO problem instances reported in the literature to date. Some of these problem instances have up to over a hundred thousand constraints and over a million binary variables. Furthermore, this research has revealed that the convergence of the D^2 method on large-scale SCO problems is in fact attainable. In addition, the dissertation demonstrates the applicability of the D^2 approach to stochastic strategic supply chain planning and stochastic matching problem instances from the literature.

4. Finally, an extension of disjunctive decomposition to two-stage SMIP with continuous first-stage is made and a new branch-and-cut procedure is proposed.

1.4 Organization of the Dissertation

This dissertation consists of nine chapters and is organized as follows. The first paragraph of each chapter gives a summary of the ideas presented in that chapter. This is followed by some background/motivation before the main ideas are presented. A problem statement and assumptions are restated in the chapter if deemed necessary for continuity and completeness.

Chapter 2 provides a literature review of SP programming with a focus on two-stage SP models. SP properties, algorithmic and computational challenges, decomposition algorithms, and applications are discussed.

Chapter 3 gives a review of theory for D^2 for two-stage SMIP. In particular, a summary of the principles of disjunctive programming with set convexification are summarized. An illustrative application of the D^2 method to the solution of a small SCO example problem instance is provided.

In chapter 4 a computer implementation of three decomposition algorithms for SMIP is discussed. These are the D^2 algorithm, the D^2-BAC (branch-and-cut) algorithm derived by Sen and Sherali (2003) and the decomposition algorithm derived by Laporte and Louveaux (1993) for two-stage SMIP with binary first-stage. Issues with a computer implementation of the D^2 algorithm are discussed and illustrative pseudo code for various parts of the algorithm is given.

Chapter 5 reports on the solution of some of the largest stochastic combinatorial optimization problems arising in server location under uncertainty. Some of these SSLP instances consist of thousands of constraints and up to a million binary decision variables.

Chapter 6 presents a comprehensive study on server location problems under uncertainty. Several valid inequalities for the SSLP models are derived and

computational experience with using the D^2 algorithm to solve several randomly generated large-scale problem instances are reported.

The main goal of chapter 7 is to investigate the performance of the D^2 method for cases in which the second stage has continuous variables. This is in contrast to the computational experiments with SSLP which is purely binary in both stages. In these experiments, we use two problems from the literature: one dealing with stochastic strategic supply chain planning, and another with stochastic bipartite matching. While the latter is a combinatorial problem, using linear programming in the second stage suffices. Hence, we include a report on these experimental results in this chapter.

The current D^2 algorithm is convergent under the assumption that the first-stage solutions are extreme points of the first-stage feasible set. Chapter 8 extends the D^2 theory to allow for the solution of SMIP problems with continuous first-stage. This is one of those cases in which the presence of continuous variables in the first-stage tends to make the solution harder for decomposition algorithms. Finally, a conclusion is given and contributions of the research and future directions along this line of work are given in Chapter 9.

CHAPTER 2

LITERATURE REVIEW

This chapter provides a literature review of SP with a focus on two-stage SMIP models. Models for two-stage SP problems with recourse, chance constrained problems, and multistage SP with recourse are given. The properties of SLP and SMIP recourse models and their differences are reviewed and the algorithmic challenges pointed out. A review of decomposition algorithms for SP is given with a focus on algorithms for SMIP. Finally, several example applications of SP are discussed.

2.1 Stochastic Programming Models and Properties

The need to incorporate uncertainty in mathematical programming models resulted in the field of SP. Early work started with Dantzig (1955) and Beale (1955). Their model involves an action followed by observation and reaction or *recourse*. Charnes and Cooper (1959) developed an alternative model called *chance* or *probabilistically constrained* programming. Even though both methods have their roots in statistical decision theory (Wald, 1950), SP focuses on methods of solution and analytical properties instead of constructing derivatives and updating probabilities. For a thorough understanding of SP the books by Kall and Wallace (1994) , Prékopa (1995) and Birge and Louveaux (1997), a survey article by Birge (1997) and a tutorial paper by Sen and Higle (1999), provide valuable resources. Recent resources on SMIP include survey articles by Schultz et al. (1996), Sen (2003) and Ph.D.

theses by Stougie (1985), van der Vlerk (1995) and Carøe (1998).

2.1.1 Two-Stage Recourse Problems

Consider a model in which a decision vector x associated with a system must be chosen in such a way that the consequences of such decisions are evaluated against several alternative outcomes of a random variable $\tilde{\omega}$ within an optimal choice model. The decision x could be a design decision for example. The random variable $\tilde{\omega}$ is used for modeling data uncertainty in the model while x is referred to as the first-stage decision. Then the performance of such a system under uncertainty is also a random variable. Therefore, measures such as expectation and other moments of performance can be naturally considered. In SP the consequences of the first-stage decisions are measured through an optimization problem referred to as the *recourse* problem, which enables the decision-maker to adapt their decision to the random variable realizations. While the SP framework allows a variety of measures (Takriti and Ahmed (2002)), the predominant measure is the "expectation". This risk measure allows the use of LP-based methodologies. However, several alternative nonlinear risk measures have been incorporated with the SP models (see e.g. Riis and Schultz (2003), Schultz (2003), Ogryczak and Ruszczynski (2002), and Bertsimas and Sim (2003)).

The general two-stage SMIP recourse model to minimize "expected cost" can be written as follows:

$$\text{Min } c^{\top}x + E[f(x,\tilde{\omega})], \tag{2.1a}$$

$$\text{s.t. } Ax \geq b, \tag{2.1b}$$

$$x \geq 0,\ x_j \text{ integer},\ j \in J_1, \tag{2.1c}$$

and for any scenario (realization) ω of $\tilde{\omega}$

$$f(x,\omega) = \text{Min } q(\omega)^\top y, \tag{2.2a}$$

$$\text{s.t. } W(\omega)y \geq r(\omega) - T(\omega)x, \tag{2.2b}$$

$$y \geq 0,\ y_j \text{ integer},\ j \in J_2. \tag{2.2c}$$

Problem (2.1) is the first-stage problem while problem (2.2) is the second-stage problem and is generally referred to as the recourse problem or scenario subproblem. In problem (2.1), c is a known vector in $\Re^{n_1}$; $E[.]$ is the usual mathematical expectation operator; constraints (2.1b) are the first-stage constraints with $A \in \Re^{m_1 \times n_1}$ and $b \in \Re^{m_1}$; constraints (2.1c) restrict some of the first-stage decision variables to be integer or binary, that is, J_1 is an index set consisting of some or all of the first-stage variables $x \in \Re^{n_1}$.

In problem (2.2), $q(\omega)$ is a cost vector in $\Re^{n_2}$ for scenario ω; y is the second-stage decision variable; constraints (2.2b) are the second-stage constraints with $W(\omega) \in \Re^{m_2 \times n_2}$, $r(\omega) \in \Re^{m_2}$ and $T(\omega) \in \Re^{m_2 \times n_1}$; constraints (2.2c) restrict some of the recourse decision variables to be integer or binary, that is, J_2 is an index set consisting of some or all of the recourse decision variables $y \in \Re^{n_2}$. The matrices A and $W(\omega)$ are assumed to be rational matrices for all ω.

Although the second-stage (recourse) variable y continues to depend on the outcome ω, this dependence is not explicitly indicated in problem (2.2). This is because the subproblem for each outcome ω is decoupled from all other outcomes once a vector x is given. Thus this formulation emphasizes the loosely coupled nature of two-stage SP problems.

Throughout the dissertation we assume that the random variables have finite support so that

$$E[f(x,\tilde{\omega})] = \sum_{\omega \in \Omega} p_\omega f(x,\omega) \text{ and } \sum_{\omega \in \Omega} p_\omega = 1,$$

where p_ω is the probability of outcome for scenario $\omega \in \Omega$. Therefore, problem (2.1-2.2) can be rewritten as a large-scale deterministic equivalent problem or DEP as follows.

$$\text{Min } c^\top x + \sum_{\omega \in \Omega} p_\omega (q(\omega)^\top y^\omega) \tag{2.3a}$$

$$\text{s.t. } Ax \geq b, \tag{2.3b}$$

$$T(\omega)x + W(\omega)y^\omega \geq r(\omega) \ \forall \omega \in \Omega, \tag{2.3c}$$

$$x \geq 0, \ x_j \text{ integer}, \ j \in J_1, \tag{2.3d}$$

$$y^\omega \geq 0, \ y_j^\omega \text{ integer}, \ j \in J_2, \ \forall \omega \in \Omega. \tag{2.3e}$$

Note that the dependence of y on ω is now explicitly indicated here since the first-stage decision variable x couples all the scenarios. Thus x is sometimes referred to in the literature as the "linking" or "complicating" variable.

The matrices $W(.)$ and $T(.)$ are sometimes referred to in the literature as the *recourse* matrix and *technology* matrix, respectively. If the recourse matrix is deterministic, that is, $W(\omega) = W$, the SP problem is said to have *fixed recourse.* If the technology matrix is deterministic, that is, $T(\omega) = T$, the SP problem is said to have *fixed tenders.* The value function $f(x, \omega)$ is referred to as the *recourse function* due to its dependence on the first-stage decision x. Similarly, $E[f(x, \omega)]$ is referred to as the *expected recourse function* of the two-stage model. When the second-stage problem is feasible for all $x \in \Re^{n_1}$, the SP problem is said to possess the *complete recourse* property. When the second-stage problem is feasible for all $\Omega \times \{Ax \geq b, x \geq 0, x_j \text{ integer }, j \in J_1\}$, the SP problem is said to possess the *relatively complete recourse* property.

When the recourse matrix W has a special structure $W = [I, -I]$, the second-stage decision variables are continuous and the constraints (2.2b) have equality constraints, the resulting SP model is said to possess the *simple recourse* property. Such a problem is referred to as a stochastic program with simple recourse.

An example of such a problem is the *news vendor problem*, sometimes known as the *newsboy problem*. In this model the vendor must determine how many papers x to buy now at a cost c without knowing the demand represented by the random variable $\tilde{\omega}$, with known distribution and given a selling price p. It is generally assumed that there is no salvage value so that the papers bought in excess of the unknown demand are discarded, resulting in potential losses. In this case the recourse problem decision variables simply measure the deviation from an uncertain target. Accurate solution methods for continuous simple recourse models have been derived in Kall and Mayer (1996).

In the SLP models the objective functions and constraints are defined by affine/linear functions. SLP models remain the most widely studied and most of the applications reported in the literature belong to this class of models. SLP problems have been shown to be convex optimization problems (see e.g. Van Slyke and Wets (1969)) and therefore, convex analysis methods are applicable to this class of convex problems as well. Thus the recourse function, and hence the expected recourse function are both convex. Nevertheless, SLP problems lack the desirable numerical property of smoothness except under the condition of absolute continuity of the random variables (Kall, 1976).

In SMIP models the recourse function inherits the properties of SLP if only the first-stage decisions include integer restrictions. Otherwise, if the integer restrictions appear in the second-stage the SMIP is much more challenging. The objective function $f(x,\omega)$ is generally discontinuous and nonconvex in x for all $\omega \in \Omega$, and the expected recourse function is lower semi-continuous under the assumption of complete recourse and a weak covariance condition (Schultz, 1993).

2.1.2 Chance Constrained Problems

Sometimes, decisions must be made in such a manner that a chosen design must satisfy a reliability constraint. Such decision models lead to the so-called *probabilistic constraints* or *chance-constrained* stochastic programs. These constraints are generally of the form:

$$P\{h(x,\tilde{\omega}) \geq 0\} \geq \alpha, \tag{2.4}$$

where, α is the probability for the constraint to hold, and the function h is often modeled by a linear function. Such constraints are often used to model system reliability. Early work on this class of SP models can be found in Prékopa (1971).

Several chance constrained models with continuous random variables have been studied by Prékopa (1971). In particular, he showed that if the function h is linear/affine in x and the randomness only appears additively, and the random variable has log-concave probability density function, then the resulting feasible region is convex. The early work for these models was restricted to normally distributed random variables. More recently, however, Sen (1992) has shown that this does not hold for discrete random variables, and in this case, the set of feasible solutions can be represented as a disjunctive set. Finally, the choice of SP model to use, a recourse model or a chance-constrained model, or even some combination of these models depends on the modeler or decision-maker.

2.1.3 Multistage Recourse Problems

Even though the focus of this dissertation is on two-stage SMIP, the ideas presented and illustrated herein can be extended to the multistage case. Therefore, we shall give a summary of the multistage SMIP model following the formulation given in Lulli and Sen (2004).

Multi-stage stochastic programs have been studied from several perspectives. Birge (1985) presents a nested Benders' decomposition algorithm in which the two stage approach is extended in such a way as to allow us to approximate the value function by a piecewise linear approximation in each stage. Gassmann (1990) reports computational results with the nested Benders' decomposition algorithm, and subsequently, several applications have been addressed using this algorithm. Higle et al. (2002) derive the stochastic scenario decomposition method, which is a statistically motivated cutting plane algorithm for multi-stage SP. Other decomposition algorithms for multi-stage SP include the scenario aggregation method of Rockafellar and Wets (1991) and the diagonal quadratic approximation method of Ruszczynski (1993), Mulvey and Ruszczynski (1995) and Rosa and Ruszczynski (1996). These algorithms are based on modifications of the augmented Lagrangian methods. Rockafellar and Wets (1992) present a dual-based approach for multistage SP, and Higle and Sen (2002) use a sampled cutting plane approach based on this dual problem.

Consider a finite horizon sequential decision process under uncertainty. Let us denote by $\mathcal{T} = \{1, ..., |\mathcal{T}|\}$ the decision horizon and assume that the information is revealed by a discrete time stochastic process $\{\omega_t\}_{t=1}^{|\mathcal{T}|}$. The decision at time t is made based on the revealed information at that time. This means that decisions will be based on the set of decisions and the outcomes of the random variables in the previous stages. So let $\underline{x_t} = (x_1, ..., x_t)$ denote the vector of decisions made from stage 1 to stage t and let $\underline{\omega_t} = 1, ..., t$ be the corresponding vector of the random variable outcomes. Then a multistage SP can be given as follows:

$$\text{Min}\{c_1(\omega_1)x_1 + Q_1(x_1) \mid W_1x_1 \leq h_1(\omega_1),\ x_1 \in X_1\}, \tag{2.5}$$

where,

$$Q_t(\underline{x_t}) = E_{\tilde{\omega}_{t+1|\underline{\omega_t}}} \text{Min}\{c_{t+1}(\tilde{\omega}_{t+1})x_{t+1} + Q_{t+1}(\underline{x_{t+1}}) : \tag{2.6a}$$

$$T_{t+1}(\underline{\tilde{\omega}_{t+1}})\underline{x_t} + W_{t+1}x_{t+1} \leq h_{t+1}(\underline{\tilde{\omega}_{t+1}}),\ x_{t+1} \in X_{t+1}\} \tag{2.6b}$$

where, $t = 1, ..., |\mathcal{T}| - 1$, $Q_{|\mathcal{T}|} \equiv 0$, and $E_{\tilde{\omega}_{t+1}|\underline{\omega}_t}$ denotes the expectation with respect to the distribution of $\tilde{\omega}_{t+1}$ conditioned on the observation $\underline{\omega}_t$. It is assumed that ω_1 is known at time $t = 1$ and that $T_t(\underline{\omega}_t), W_t, c_t(\omega_t), h_t(\underline{\omega}_t)$ are rational matrices and vectors of conformable dimensions. The set X_t denotes restrictions on the decision variables requiring some of them to be integer.

As in the two-stage case, to ensure that problem (2.5) is well defined, we can impose the relatively complete recourse assumption. This implies that the expectation defining Q_t is finite for any policy $\underline{x}_t$. The assumption of finite support for the random vector ω implies that $\Omega = (\omega^1, ..., \omega^r)$ with probabilities $p^1, ..., p^r$. Therefore, we can represent uncertainty by means of scenarios, where a scenario is a realization of the random variable $(c(\omega), h(\omega), T(\omega))$ corresponding to an elementary atom $\omega \in \Omega$.

The evolution of all information trajectories over time in a multistage stochastic program can be represented by a *scenario tree* (Birge and Louveaux, 1997). The scenario tree represents the relationship between scenarios. Let $\aleph$ be the node set for the scenario tree. At each node of the tree we have a branch(es) to indicate future possible outcomes of the random variable from that node. A scenario includes one node at each stage and is represented by a path from the root node to a leaf node of the tree. The root node represents the first-stage (stage 1) while the leaf nodes represent the final stage (stage $|\mathcal{T}|$). Let the set of scenarios be given by $\mathcal{S} = \{1, ..., r\}$. Then the correspondence between nodes of the scenario tree and the 2-tuples $(t, s) \in \mathcal{T} \times \mathcal{S}$ is given by the surjective map $\mathcal{H} : \mathcal{T} \times \mathcal{S} \rightarrow \aleph$. Now if we associate a vector of decisions $x(\omega^s) = (x_1(\omega^s), ..., x_{|\mathcal{T}|}(\omega^s))$ with each scenario $s \in \mathcal{S}$, we can write the deterministic equivalent problem (DEP) formulation of the

multistage stochastic program (2.5) as follows:

$$\text{Min} \sum_{s=1}^{r} p^s [\sum_{t=1}^{|\mathcal{T}|} c_t(\omega_t^s) x_t(\omega^s)] \tag{2.7a}$$

$$\text{s.t. } W_1 x_1(\omega^s) \leq h_1(\omega_1^s), \quad \forall s \in \mathcal{S}, \tag{2.7b}$$

$$T_{t+1}(\underline{\omega}_{t+1}^s)\underline{x}_t(\omega^s) + W_{t+1}x_{t+1}(\omega^s) \leq h_t(\underline{\omega}_{t+1}^s), \ \forall s \in \mathcal{S}, \forall t \in \mathcal{T}, \tag{2.7c}$$

$$x_t(\omega^s) = [\sum_{u \in \mathcal{B}_t^s} p^u x_t(\omega^u)] / \sum_{u \in \mathcal{B}_t^s} p^u, \ \forall s \in \mathcal{S}, \forall t \in \mathcal{T}, \tag{2.7d}$$

$$x_t(\omega^s) \in X_t, \ \forall s \in \mathcal{S}, \forall t \in \mathcal{T}, \tag{2.7e}$$

where, $\mathcal{B}_s^t$ represents the set or bundle of scenarios that are indistinguishable from scenario s at time t, that is, all scenarios u for which $\omega_\tau^u = \omega_\tau^s$ for all $\tau = 1, ..., t$. A bundle of scenarios for any node $\mathcal{H}(t, s)$ of the scenario tree includes all paths passing through that node. The constraints (2.7d) state that all scenarios with the same history until the t-th stage must share the same decision until this stage. Therefore, these constraints are referred to as the *nonanticipativity* constraints in SP literature. They also imply that decisions depend only on information revealed in the past and not in the future. The nonanticipativity constraints can also be written for each node $\mathcal{H}(t, s)$ of the scenario tree as follows

$$X_{\mathcal{H}(t,s)} = x_t(\omega^s) \ \forall s \in \mathcal{S}, \ \forall t \in \mathcal{T}, \tag{2.8}$$

where, $X_{\mathcal{H}(t,s)}$ is a decision associated with the node $\mathcal{H}(t, s)$. Next we turn to the computational challenges of SP.

2.2 Computational Challenges

Let us now briefly summarize some of the algorithmic challenges of SP. The main computational challenges can be attributed to the evaluation of the multi-dimensional integral as was recently proved in Dyer and Stougie (2003). They showed that two-stage SP programs are actually $\#P$-hard, meaning that they have

the same complexity as the most difficult counting problems in combinatorics. The complexity class $\#P$ was first defined by Valiant (1979). This class is a set of integer-valued functions that express the number of accepting computations of a nondeterministic Turing machine of polynomial time complexity. Therefore, the notions $\#P$ and $\#P$-hard or $\#P$-completeness express the hardness of problems that count the number of solutions. Let $\mathbb{N}$ denote the set of nonnegative integers and let Σ be the finite alphabet of the input and output of the Turing machines considered. Then Fortnow (1997) defines $\#P$ as follows.

Definition 1. *The class $\#P$ consists of the functions $f : \Sigma^* \Rightarrow \mathbb{N}$ such that there exists a nondeterministic polynomial time Turing machine M such that for all inputs $x \in \Sigma^*$, $f(x)$ is the number of accepting paths of M.*

This implies that efficient or polynomial time algorithms for two-stage stochastic program solutions are not likely to be found. Even in the case of discrete random variables the total number of outcomes or scenarios may be too large for evaluating the expected value function. Thus one has to resort to approximations of the value function as explained momentarily.

In SMIP integrality requirements on (some of) the first-stage and/or second-stage decision variables introduce further computational complexity. Integer programming is generally NP-hard and therefore, SMIP models inherit this property. However, Dyer and Stougie (2003) have shown that $\#P$-hardness of the evaluation of the integrals involved remains the dominating factor in the overall complexity of SMIP problems. Once there are integrality requirements on the second-stage decision variables, the evaluation of one $f(x, \omega)$ requires the solution of an NP-hard problem. Furthermore, the convexity properties of the continuous recourse and expected recourse functions no longer hold. As pointed out earlier in Section 2.1.1, $f(x, \omega)$ is generally discontinuous and nonconvex in x for every $\omega \in \Omega$ (Schultz, 1993). Hence successful SLP solution methods are difficult to adapt for

the solution of SMIP problems.

The two main approaches to generating approximations of the scenario outcomes are data-aggregation and data-selection. Note that these approximations have to do with approximations of the value function and not approximations of the feasible set. In data-aggregation the algorithms lead to successive approximation methods in which finer discretizations of the sample space are created based on the solution of an aggregated stochastic program. Methods based on data-aggregation can be found in Frauendorfer (1992) and Edirisinghe and Ziemba (1996) for two-stage stochastic programs and Frauendorfer (1994) for multistage stochastic programs.

Data-selection methods arise in sample-based methods. Römish and Schultz (1991) and Shapiro (1991) use a fixed point and perform a statical analysis of the output. Shapiro and Homem de Mello (1998) suggest solving a sequence of sample approximations with increasing sample size to obtain asymptotic results. However, as the sample size gets larger (and therefore the approximating problem) each iteration may become computationally demanding. Higle and Sen (1991) and later, Higle and Sen (1996) and Higle and Sen (1999) derive a stochastic decomposition algorithm (SD) to speed up the computations associated with such a method, whereby approximations generated earlier in the iterations are sequentially updated.

2.3 Algorithms for Stochastic Programming

There are two fundamental approaches to solution methods for SP; direct methods using SP structure and decomposition approaches. Direct approaches are largely for SLP and include extreme-point methods (Kall (1979), Strazicky (1980), Birge (1995)), interior point methods (Lustig et al. (1991) and Lustig et al. (1994)), and column splitting method (Carpenter et al., 1991). Since the focus of this dissertation

is on decomposition approaches, we shall review some of the decomposition methods that have shown potential for solving large-scale problems. The basic idea behind decomposition approaches is to decompose the large-scale SP into a master program and subproblems for each scenario, and then solving the smaller pieces of the problem separately in a decomposition-coordination setting.

The decomposition approaches can be divided into two categories based on how the SP problem is decomposed. Resource-directive decomposition methods decompose the large-scale SP problem stage-wise or time-wise. On the other hand, price-directive methods decompose the SP problem scenario-wise. Stage-wise decomposition scales well with the number of scenarios in the two-stage case. However, the scalability of stage-wise decomposition with respect to multistage SLP problems remains unclear (Sen, 2003). In this dissertation we follow stage-wise decomposition for two stage SMIP and therefore, our review on decomposition algorithms will be focused as such. Before reviewing the algorithms for SMIP, we first state the L-shaped method for SLP since it provides a framework for the algorithmic setting in this dissertation, and then point out other SLP methods.

2.3.1 Decomposition Algorithms for SLP

The size of an SP model grows linearly with the number of possible realizations of the random parameters. Further, this number also increases exponentially with the number of decision stages in the model in the multistage case. Consequently, the algorithmic developments in this area focused on exploiting the SP model structure. This in turn led to the development of the L-shaped method (Van Slyke and Wets, 1969) for SLP. The L-shaped method is essentially a Dantzig-Wolfe decomposition (Dantzig and Madansky, 1961) of the dual or Benders' decomposition (Benders, 1962) of the primal. These methods deal with linear models and are cutting-plane-based (see e.g. Kelley (1960)).

The L-shaped method, derived by Van Slyke and Wets (1969), forms the fundamental decomposition algorithm on which many other SP methods are based. The name comes from the *dual-block angular* structure of the linear case for problem (2.3). In this approach a master program in x and η is built, where the variable η represents the expected recourse function evaluations. Thus the assumption of finite support is taken in order to make this approach possible. The master program takes the form:

$$\text{Min } c^\top x + \eta, \tag{2.9a}$$

$$\text{s.t. } Ax \geq b, \tag{2.9b}$$

$$D_\ell x \geq d_\ell, \ \ell = 1, ..., r, \tag{2.9c}$$

$$E_\ell x + \eta \geq e_\ell, \ \ell = 1, ..., s, \tag{2.9d}$$

$$x \geq 0, \ \eta \in \Re, \tag{2.9e}$$

and for each scenario $\omega \in \Omega$ the subproblem to solve is

$$f(x, \omega) = \text{Min } q^\top y, \tag{2.10a}$$

$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x, \tag{2.10b}$$

$$y \geq 0, \tag{2.10c}$$

Constraints (2.9c) are the so-called *feasibility cuts* while constraints (2.9d) are called *optimality cuts*. The feasibility cuts determine $\{x \mid E[f(x, \tilde{\omega})] < +\infty\}$ and the optimality cuts provide linear supports of $E[f(x, \tilde{\omega})]$ on its domain of finiteness. The L-shaped algorithm can be summarized as follows.

The L-Shaped Algorithm

Step 0. Initialization:

Set $s \leftarrow 0$, $t \leftarrow 0$ and $k \leftarrow 0$.

Step 1. Solve the master program

Set $k \leftarrow k+1$. Solve master program (2.9).
Let (x^k, η^k) be an optimal solution. If no constraint (2.9d) is present, η^k is set to $-\infty$ and not considered in the computation of x^k.

Step 2: Solve scenario subproblems

Solve scenario subproblem (2.10b) for all $\omega \in \Omega$
If for some ω subproblem is infeasible
Let σ^k be the associated dual rays and define
$D_{r+1} = \sigma^k T(\omega)$ and $d_{r+1} = \sigma^k r(\omega)$ to generate a feasibility cut (2.9c).
Set $r \leftarrow r+1$. Add feasibility cut to master program and return to Step 1.
else
Let π^k be the associated dual multipliers and define
$E_{s+1} = \sum_{\omega\in\Omega} p_\omega \pi^k(\omega) T(\omega)$ and $e_{s+1} = \sum_{\omega\in\Omega} p_\omega \pi^k(\omega) r(\omega)$ to generate an optimality cut (2.9c).
Let $w^k = e_{s+1} - E_{s+1} x^k$

Step 2: Termination

If $\eta^k \geq w^k$, stop. The solution x^k is optimal.
Otherwise set $s \leftarrow s+1$, and add optimality cut to master program and return to Step 1.

Birge and Louveaux (1988) have proposed a multi-cut approach for this method in which the optimality cut is disaggregated for each scenario. This requires a variable $\eta(\omega)$ for all $\omega \in \Omega$ in the master program for approximating the function value

of each scenario subproblem. The L-shaped method has also been generalized to multistage SLP as a nested decomposition method by Birge (1985) and Gassmann (1990). Also, Laporte and Louveaux (1993) extended the L-shaped method to models with integer decision variables and derived the *integer* L-shaped method.

In many practical SP problems the number of scenarios is generally large. SP models often include some approximation of an underlying probability distribution. When the number of scenarios is very large and the underlying probability distribution is known, it is common to resort to sampling. Two main methods that embed sampling into the L-shaped method are the approaches by Dantzig and Glynn (1990) and Dantzig and Infanger (1991) and by Higle and Sen (1991) and Higle and Sen (1996).

The first approach is based on large samples to derive the cuts. In this approach, $f(x, \omega)$ is sampled in the L-shaped method instead of actually computing $E[f(x, \tilde{\omega})]$. Dantzig and Infanger (1991) report on the solution of experiments with large-scale problems. The results improve significantly with importance-sampling variance reduction techniques. In order to form confidence intervals on the optimal values, however, Infanger (1991) makes several assumptions.

The second approach, called stochastic decomposition (SD), generates many cuts with increasing samples based on previous samples. These cuts are updated and/or dropped as the algorithm continues processing. The authors assume complete recourse and a known lower bound on $f(x, \omega)$ (generally 0). They also assume that the set of dual solutions are bounded and the set $X = \{x \mid Ax \geq b, x \geq 0\}$ and Ω are also compact. This approach has been applied to solve large-scale network design problems (Sen et al., 1994).

2.3.2 Decomposition Algorithms for Two-Stage SMIP

Decomposition algorithms for integer SP models started to appear only recently. In SMIP, the type of integrality restrictions (binary, mixed-binary, general integer, mixed-integer) and where it appears (first-stage, second-stage) in the model greatly determines the type of decomposition algorithm suitable for the model. Therefore, we can categorize algorithms for SMIP based on the integrality restrictions on the decision variables in the model as in Sen (2003): (a) simple integer recourse models with random RHS, (b) binary first-stage, arbitrary second-stage, (c) binary first-stage, 0-1 MIP second-stage with fixed recourse, (d) binary first-stage, MIP second-stage, (e) continuous first-stage, integer second-stage and fixed tenders, and (f) 0-1 MIP in both stages with general random data.

(a) Simple Integer Recourse Models with Random RHS

Imposing integer restrictions on the second-stage SLP simple recourse model and rewriting the equality constraints into inequality constraints results in the simple integer recourse (SIR) model. This model has been extensively studied by Klein Haneveld et al. (1995) and Klein Haneveld et al. (1996). Let i denote the row index. Under the assumptions that all data elements in the problem are deterministic except the right-hand side, and that $r_i(\tilde{\omega})$ has finite support and that the technology matrix T has full rank, these researchers have shown that it is possible to compute the convex hull of the expected recourse function by using enumeration over each dimension i. However, the resulting problem only provides a lower bound since the first-stage feasible set $X \cap \mathbf{X}$ is not used in the convexification process. Consequently, branch-and-bound may be necessary to close the gap. More recently, van der Vlerk (2002) has extended this approach to general recourse models with totally unimodular recourse matrices.

(b) Binary First Stage, Arbitrary Second Stage

This class of SMIP models has pure 0-1 decision variables in the first-stage. Assuming relatively complete recourse and that a lower bound L on the expected recourse function is known or can be computed, Laporte and Louveaux (1993) derive valid inequalities that can be applied to the expected recourse function. They propose the earliest decomposition algorithm for this type of SMIP model which is illustrated in Birge and Louveaux (1997). Their approach follows the algorithmic setting of Benders' decomposition (Benders, 1962) (and consequently the L-shaped method (Van Slyke and Wets, 1969)) in that, at each iteration k, a master program is solved whose solution x^k is passed on to the scenario subproblem MIPs. All the scenario subproblem MIPs are solved exactly at each iteration of the algorithm.

The optimality cut proposed by Laporte and Louveaux (1993) has been shown to be generally weak. It is only sharp at the point x^k and that its value is at most L at all other feasible solutions. Nevertheless, Birge and Louveaux (1997) show how to improve the optimality cut when more information is available on $E[f(x^k, \tilde{\omega})]$, such as other bounds.

(c) Binary First Stage, 0-1 MIP Second Stage with Fixed Recourse

We now turn to SMIP models with pure binary first-stage and mixed-binary second-stage and fixed recourse matrix W. Under the assumption of relatively complete recourse, Sen and Higle (2000) propose the *common-cut-coefficient* theorem and a (D^2) algorithm for this class of problems. The methodology follows a sequential convexification of the recourse function problem in the context of Benders' decomposition, and is motivated by the need to avoid solving every subproblem from scratch in each iteration. The cuts are generated in both stages in this approach. The cuts in the second-stage, referred to as the D^2 cuts, are generated using disjunctive programming (see e.g. Balas (1979)). Under the C^3 theorem, one

cut generated for one scenario can easily be translated for the other scenarios. These cuts can also be derived using the reformulation-linearization technique (RLT) of Sherali and Fraticelli (2002). This SP solution approach forms the basis for this dissertation and is described in greater detail in the ensuing chapters.

(d) Binary First Stage, MIP Second Stage

The approach described in the last case extends to this case under relatively complete recourse and fixed recourse assumptions. However, the properties of branch-and-bound algorithms are now incorporated into the approach. Recently, Sen and Sherali (2003) proposed a D^2 with branch-and-cut or D^2-BAC algorithm for this class of SMIP problems. The essence of the method is to use information provided by the branch-and-bound tree on the second-stage to derive approximations of the value function of the second-stage MIP via disjunctive programming. This approach is also described in detail in the dissertation.

(e) Continuous First Stage, Integer Second Stage and Fixed Tenders

So far with the exception of the SIR models, all the models considered have binary first-stage. For the case with continuous first-stage and mixed-integer second-stage the situation is more complex. Finite termination of the method when the first-stage is continuous is far from obvious. Ahmed et al. (2004) derive a finite branch-and-bound algorithm for this class of SMIP models that is based on global optimization. They assume pure integer recourse and fixed tenders ($T(\omega) = T$) but the recourse matrix is allowed to be random.

The main observation in this approach is that the value function of a pure IP with integer W is constant over hyper-rectangles ("boxes"). Furthermore, if the set $X = \{x \mid Ax \geq b, x \geq 0\}$ is bounded, then there are finitely many such

hyper-rectangles. In fact, this observation was used by Schultz et al. (1998) to design an enumerative scheme for the first-stage decisions while the second-stage decisions were obtained using polynomial ideal theory. To fit the problem into a global optimization setting they transform the problem into the space of "tender variables" $\chi = Tx$. Then the transformed problem becomes:

$$\underset{\chi \in \Psi}{\text{Min}}\, \varphi(\chi), \tag{2.11a}$$

where $\Psi = \{\chi \mid Tx = \chi,\ x \in X\}$ and φ is defined as the sum of

$$\phi(\chi) = \text{Min}\{c^\top x \mid Tx = \chi,\ x \in X\} \text{ and } \Phi(\chi) = \sum_{\omega \in \Omega} p_\omega h(r(\omega) - \chi),$$

where $h(r(\omega) - \chi)$ denotes the value function resulting from the value of a pure IP with right-hand side $r(\omega) - \chi$.

(f) 0-1 MIP in Both Stages with General Random Data

Unlike the previous cases considered, this case has mixed-binary in both stages with general random data. There is no special structure associated with the random elements and therefore, it is easier to cast this SMIP model into the large-scale DEP formulation (2.3). This model was studied by Carøe (1998) who devised a branch-and-cut method in which the cuts are based on disjunctive programming. Carøe (1998) derives lift-and-project cuts of Balas et al. (1993) in the DEP setting. The cuts are in the $(x, y(\omega))$ space and are derived for each scenario. Other types of cuts can also be used.

The basic idea of the method is as follows. The LP relaxation of the DEP is first solved to optimality to get a solution $(\bar{x}, \bar{y}(\omega)\}_{\omega \in \Omega})$. If the solution satisfies the integrality restrictions then an optimal solution has been found and the method stops. Otherwise, a fractional variable solution is chosen and used for cut derivation, which involves forming and solving another LP for each scenario $\omega \in \Omega$. After

all the cuts are generated they are added to the DEP LP relaxation and the process repeated. Branching is embedded in this method just as in deterministic integer programming. Since lift-and-project cuts are generally costly in terms of computation time, Carøe (1998) suggests translating a cut derived for one scenario to another scenario if possible. This is possible under the assumption that the technology and recourse matrices are fixed, that is, $T(\omega) = T$ and $W(\omega) = W$.

Finally, let us point out that among the recent work that seem promising towards the development of practical algorithms for SMIP is Carøe (1998) and Carøe and Schultz (1999), who use the scenario decomposition approach of (Rockafellar and Wets, 1991) to develop a branch-and-bound algorithm and Norkin et al. (1998), who propose a stochastic branch-and-bound method for minimizing the expected value of an arbitrary function over a finite set.

2.4 Applications of Stochastic Programming

Due to the fact that many real-life problems have inherent uncertainty in them, applications for SP are vast. In this section we simply highlight a few of the applications where both SLP and SMIP have seen significant success and provide references for further reading. Unlike SLP models, most SMIP models started appearing in the literature only in the last few years. This has mainly been due to lack of practical algorithms to tackle these problems. For instance, Bertsimas (1994) presents a variety of SP problems with discrete decision variables but these problems are reduced to deterministic-equivalent or near equivalent problems. A lot of practical problems, such as capacity planning and strategic supply chain planning under uncertainty often involve discrete decision variables. Thus applications for SMIP will continue to grow as more practical solution methods for these problems are derived and implemented. Next we discuss applications of SP to telecommunication, transportation, finance, manufacturing and electricity power

generation.

2.4.1 Telecommunication

The system traffic, performance and reliability of telecommunications systems planning and operations naturally involve uncertainties. Therefore, SP naturally renders itself a viable approach to problems that arise in this field. Sen et al. (1994), for example, applied the SP planning methodology to an industrial-sized network planning problem for Sonet-Switched Network (SSN), and demonstrated improved network performance due to the SP model. This particular problem involves making network design and configuration decisions that require consideration of random point-to-point demands with high variance forecasts in the network. The authors used the stochastic decomposition (SD) method to solve the problem.

Another problem in telecommunications system that is amenable to the SP approach is the server location problem under uncertainty. These problems find many real-life applications in situations where facilities or "servers" have to be located at some given potential sites in order to provide some service to some potential "clients". In such problems uncertainty appears not only in the client demands, but also in the client availability and server location costs. For example, Wang et al. (2003) study the facility location problem for immobile servers with continuous stochastic demands. They present several models and provide heuristics for their solutions. Riis et al. (2004) study a server location problem for the deployment of mobile switching centers in a telecommunications network and report on the solution of a real large-scale problem instance using the SP approach.

2.4.2 Transportation

Many transportation models are commonly formulated as SP models from the Ferguson and Dantzig (1956) model. In particular, dynamic vehicle allocation has been one of the prominent areas in which SP has been applied. Dynamic vehicle allocation involves routing a set of vehicles (e.g. trucks, freight cars, planes) to meet demand along routes and to position them for anticipated future demands (loads). The objective is to maximize the total returns over given time horizons. See for example Powell (1988), Powell (1990), Frantzeskakis and Powell (1990) and Powell (1996) for various SP dynamic vehicle allocation models. Over the last few years, Powell and Gittoes (1996) and Powell et al. (2004) have developed approximations and an adaptive labeling algorithm that effectively approximate the value function at each time period and yield a form of dynamic approximation.

Other SP models in transportation include the widely studied stochastic vehicle routing problem. See for example Laporte et al. (2002) propose the integer L-shaped method for the capacitated vehicle routing problem with stochastic demands, Kenyon and Morton (2003) study the stochastic vehicle routing with random travel times, and Laporte et al. (1994) propose an exact solution for the *a priori* optimization of the probabilistic traveling salesman problem.

2.4.3 Electricity Power Generation

Another common area of application and source of developments for SP methods has been electricity power generation. The problem, usually referred to as the *unit commitment*, aims at finding a fuel cost optimal scheduling of startup/shutdown decisions and operation levels for power generation units over some given time horizon. (Carøe, 1998) and Carøe and Schultz (1998) study a unit commitment

problem in the presence of uncertainty in the load profiles and develops a two-stage SMIP model with integer first-stage and mixed-integer recourse. They apply a Lagrangian-based decomposition algorithm to solve a problem with real data for a German utility company. The problem has a total of 20,000 integer and 150,000 continuous decision variables with up to 180,000 constraints.

Other examples include Pereira and Pinto (1985) and Pereira and Pinto (1991), who use decomposition procedures for models of the Brazilian power system, and Takriti et al. (1996), who apply the progressive hedging algorithm to a model of the Michigan power system designed for daily scheduling. They report achieving a convergence to near optimal solutions quickly with potential savings over a deterministic procedure of almost $150,000 in generation costs for one sample week.

The recent deregulation of the electricity market has also led to the development of new SP models in this area. For example, Sen et al. (2003) develop a SP-based model for power portfolio optimization called DASH. This model is designed to help decision-makers to coordinate production decisions with opportunities in the wholesale power markets. They report that their model selects portfolio positions that perform well on a variety of scenarios generated through statistical modeling and optimization.

2.4.4 Finance

Finance problems inherently involve uncertainty due to the future time nature of financial returns and are therefore, amenable to the SP approach. The goal of SP is to provide a strategy for making decisions that hedge against unforseen scenarios and thus avoid potential losses. An excellent example of SP application to finance is the Russel-Yasuda Kasai Model reported in Carinõ et al. (1994), which won second prize in the 1993 Franz Edelman Award Competition for Management Science Achievement. In the model decisions are made for a Japanese insurance

company on how to optimally invest in assets to meet an uncertain liability stream over time. The investment returns are also random and the model includes legal constraints about the use of income to meet liabilities. The authors model the problem as a multistage SLP problem and report that the model yield $79 million in its first year of use. For a list of other successes of application of SP to finance see an article in *Business Week* (Coy, 1996)

Other finance models in finance are presented in Markowitz (1952) and Kusy and Ziemba (1986). For more recent work in this field see papers by Rockafellar and Uryasev (2000) and Rockafellar and Uryasev (2002) and a book by Uryasev and Pardalos (2001).

2.4.5 Manufacturing

Manufacturing usually involves complex operations in which randomness cannot be ignored. The demand and supply aspects of manufacturing are often characterized by randomness. In fact, the uncertainty inherent in manufacturing operations make this area also particularly amenable to SP models. In recent years there has been a lot of interest in applying the SP approach to tackle problems especially in capacity-planning and expansion and strategic supply chain management under uncertainty.

Eppen et al. (1989) provide a capacity-planning model at General Motors formulated as a SLP whose goal is to determine capacity for various products at a number of plants. They maximize an expected profit objective with a downside risk constraint. Ahmed and Garcia (2003) study the dynamic capacity acquisition and assignment problem under uncertainty using the SP approach. Given a set of resources and tasks, this problem seeks to find a minimum cost schedule of capacity acquisitions for the resources and the assignment or resources to tasks over a given planning horizon. This problem arises, for example, in the integrated planning of locations and capacities of distribution centers (DCs), and the assignment of

customers to the DCs, in supply chain applications. The randomness in the problem appear in the assignment costs and the processing requirements for the tasks. They formulate a SMIP model and apply a decomposition based branch-and-bound method (Ahmed et al., 2004) to solve numerous instances of the problem.

Application of SP to strategic supply chain management under uncertainty seems to have gained interest only in the last few years. Strategic supply chain planning involves the determination of production topology, plant sizing, product selection, product allocation among plants and vendor selection for raw materials. The goal is to maximize the expected profit over a given time horizon for the investment depreciation and operations costs. Uncertainty in strategic supply chain planning may appear in the product net price, product demand, raw material supply cost and production cost. Some recent work in this area include that of Escudero et al. (1996), MirHassani et al. (2000), Ahmed et al. (2003), and (Alonso-Ayuso et al., 2003). In particular, (Alonso-Ayuso et al., 2003) follow a two-stage SP approach for the problem and derive a *branch-and-fix coordination* (BFC) method and report on the solution of large-scale SMIP problem instances.

2.4.6 Other Applications

There are many other applications for SP. For instance, with the rise in global terrorism, we expect to see new SP models to address some of the problems involved for example, in both anti-terrorism and counter-terrorism, and in the prevention and counteraction of cyber attacks. These problems are amenable to the SP approach due to the inherently uncertainty in the problem data. Other SP applications include military applications (Morton et al., 1996; Baker et al., 2002), network interdiction (Cormican et al., 1998; Woodruff, 2002), and airport security (Simms, 1997).

CHAPTER 3

A SUMMARY AND ILLUSTRATION OF DISJUNCTIVE DECOMPOSITION WITH SET CONVEXIFICATION

This chapter has been published as Sen et al. (2002) and has been included in this dissertation for convenience in providing the needed background. In this chapter the disjunctive decomposition (D^2) algorithm for two-stage Stochastic Mixed Integer Programs (SMIP) is reviewed. This method uses the principles of disjunctive programming to develop cutting-plane-based approximations of the feasible set of the second-stage problem. At the core of this approach 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 other outcomes. An illustrative application of the D^2 method to the solution of a small SMIP illustrative example is provided.

SMIP comprise one of the more difficult classes of mathematical programming problems. Indeed, this class of problems combines the extremely large scale nature of stochastic programs and the inherent computational difficulties of combinatorial optimization. The main difficulty in solving two-stage stochastic mixed-integer programs is that the recourse costs are represented as the expected value of a mixed-integer program whose value function is far more complicated than the value function of a linear program. In general, the expected recourse function is non-convex and possibly discontinuous. In this chapter the Disjunctive Decomposition (D^2) algorithm with set convexification for two stage SMIP proposed by Sen and Higle (2000) is illustrated.

The method uses the principles of disjunctive programming to develop a cutting-plane-based approximation of the feasible set of the second-stage problem. This task is streamlined via the Common Cut Coefficients (C^3) Theorem (Sen and Higle, 2000), which provides a simple mechanism for transforming cuts derived for one instance of the second-stage problem into cuts that are valid for another instance. This significantly reduces the effort required to approximate the convexification of the feasible set, a task that must be undertaken for each possible outcome of the random variables involved. In this chapter, the D^2 algorithm and the manner in which the C^3 Theorem is used to reduce the computational effort are illustrated. Because the methodology is related to, but distinctly different from, the work of Carøe (1998), we also use this forum to highlight the relationship between the two approaches.

This chapter is organized as follows. In Section 3.1 the results of Sen and Higle (2000) are summarized, and identify connections between their work and that of Carøe (1998). In Section 3.2 the application of the D^2 Algorithm is illustrated through a simple numerical example with both first and second-stage binary variables. Finally, a discussion and our conclusions are found in Section 3.3.

3.1 Background

In this section the main results from Sen and Higle (2000) that are critical to the illustration of the D^2 algorithm are given. In particular, the C^3 theorem is reviewed the details of its application discussed. For a more thorough explanation of disjunctive decomposition concepts, proofs, and the derivation of the D^2 algorithm, the reader is referred to Sen and Higle (2000). Throughout this chapter the following SMIP problem is considered:

$$\underset{x \in X \cap \mathcal{B}}{\text{Min}} \; c^\top x + E[f(x, \tilde{\omega})], \tag{3.1}$$

where $X \subseteq \Re^{n_1}$ is a set of feasible first-stage decisions, $\mathcal{B} \subset \Re^{n_1}$ is the set of binary vectors, $\tilde{\omega}$ is a random variable defined on a probability space $(\Omega, \mathcal{A}, \mathcal{P})$, and for any $\omega \in \Omega$

$$f(x,\omega) = \text{Min } g_u^\top u + g_z^\top z, \tag{3.2a}$$

$$\text{s.t. } W_u u + W_z z = r(\omega) - T(\omega)x, \tag{3.2b}$$

$$u \in \Re_+^{n_u}, z \in \mathcal{B}^{n_z}. \tag{3.2c}$$

It is assumed that X is a convex polyhedron, Ω is a finite set, and that $f(x,\omega) < \infty$ for all $(x,\omega) \in X \times \Omega$. Moreover, it is assume that by using appropriately penalized continuous variables, the subproblem (3.2) remains feasible for any restriction of the integer variables z. Note that the inclusion of integer variables in the second-stage problem, (3.2), is the primary source of the computational and algorithmic challenges associated with (3.1). In particular, in order to evaluate the SMIP objective $cx + E[f(x,\tilde{\omega})]$, it is necessary to solve (implicitly or approximately) the MIP (3.2) for each $\omega \in \Omega$. Moreover, the structural difficulties associated with MIP objective functions are well documented (see, e.g., Blair and Jeroslow (1982) and Blair (1995)). These difficulties are compounded by the fact that the expected value operations associated with the SMIP objective function amounts to a convex combination of the complicated individual MIP objective functions. The C^3 Theorem exploits the specific structure of (3.2), thereby permitting a computationally streamlined development of SMIP objective function approximations.

3.1.1 Common Cut Coefficients

In an effort to develop approximations of the SMIP objective, we begin with an approximation of the convex hull of feasible integer points for (3.2). This set can

be expressed as a disjunction,

$$\mathcal{S} = \bigcup_{h \in H} \mathcal{S}_h,$$

where H is a finite index set, and the sets $\{S_h\}_{h \in H}$ are polyhedral sets represented as

$$\mathcal{S}_h = \{y \mid W_h y \geq r_h, y \geq 0\}$$

Within our setting, we have $y = (u, z)$ as in (3.2) and r_h includes $r(w) - T(w)x$. More formally, we note that the constraints in (3.2),

$$W_u u + W_z z \geq r(\omega) - T(\omega)x$$

vary with the first-stage decision, x, and the scenario ω. Consequently, the disjunctive representation of the set depends on $(x, \omega) \in X \times \Omega$,

$$\mathcal{S}(x, \omega) = \bigcup_{h \in H} \mathcal{S}_h(x, \omega), \tag{3.3}$$

where

$$\mathcal{S}_h(x, \omega) = \{y \mid W_{hu} u + W_{hz} z \geq r_h(x, \omega), u, z \geq 0\}.$$

A convex relaxation of the nonconvex set (3.3) can be represented by a collection of valid inequalities of the form

$$\pi_u^\top u + \pi_z^\top z \geq \pi_0(x, \omega).$$

While the disjunctive representation depends on (x, ω), the C^3 Theorem, which is stated below, ensures that as the argument changes, cut validity can be maintained by a shift in the right-hand-side element without altering the gradient of the cut. In the following we use $n_2 = n_u + n_z$.

Definition 1. *(The C^3 Theorem). Consider the stochastic program with fixed recourse as stated in (3.1), (3.2). For $(x, \omega) \in X \times \Omega$, let $Y(x, \omega) = \{y = (u, z) \mid Wy \geq r(\omega) - T(\omega)x, u \in \Re_+^{n_u}, z \in \mathcal{B}^{n_z}\}$, the set of mixed-integer feasible solutions for the second-stage mixed-integer linear program. Suppose that $\{C_h, d_h\}_{h \in H}$, is a*

finite collection of appropriately dimensioned matrices and vectors such that for all $(x, \omega) \in X \times \Omega$

$$Y(x,\omega) \subseteq \bigcup_{h \in H} \{y \in \Re_+^{n_2} \mid C_h y \geq d_h\}.$$

Let

$$\mathcal{S}_h(x,\omega) = \{y \in \Re_+^{n_2} \mid Wy \geq r(\omega) - T(\omega)x, C_h y \geq d_h\},$$

and let

$$\mathcal{S} = \bigcup_{h \in H} \mathcal{S}_h(x,\omega).$$

Let $(\bar{x}, \bar{\omega})$ *be given, and suppose that* $\mathcal{S}_h(\bar{x}, \bar{\omega})$ *is nonempty for all* $h \in H$ *and* $\pi^T y \geq \pi_0(\bar{x}, \bar{\omega})$ *is a valid inequality for* $\mathcal{S}(\bar{x}, \bar{\omega})$. *There exists a function,* $\pi_0 : X \times \Omega \rightarrow \Re$ *such that for all* $(x, \omega) \in X \times \Omega$, $\pi^T y \geq \pi_0(x, \omega)$ *is a valid inequality for* $\mathcal{S}(x, \omega)$.

Proof. See Sen and Higle (2000).

The C^3 Theorem ensures that a valid inequality for the set $S(\bar{x}, \bar{\omega})$ of the form $\pi^\top y \geq \pi_0(\bar{x}, \bar{\omega})$ can be translated to an inequality, $\pi^\top y \geq \pi_0(x, \omega)$ that is valid for the set $S(x, \omega)$. The cut coefficients, π, are common to both sets. Thus, one may derive the left hand side coefficients, π, which may be applied to all scenario problems. The right hand side $\pi_0(x, \omega)$ is derived as necessary for each pair (x, ω) using a strategy from reverse convex programming in which disjunctive programming is used to provide facets of the convex hull of reverse convex sets (Sen and Sherali, 1987). Given the valid inequalities, $\pi^\top y \geq \pi_0(x, \omega)$, a lower bound approximation for the scenario subproblem objective function is given by:

$$f(x,\omega) \geq f_0(x,\omega) \equiv \text{Min } g^\top y \tag{3.4a}$$

$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x \tag{3.4b}$$

$$\pi^T y \geq \pi_0(x,\omega) \tag{3.4c}$$

$$y \geq 0 \tag{3.4d}$$

$$\tag{3.4e}$$

We note that a version of Theorem 1 appears in Carøe (1998), although there are some critical distinctions between Sen and Higle (2000) and Carøe (1998). Specifically, while Sen and Higle (2000) work within the context of the temporal decomposition indicated in (3.1,3.2), Carøe (1998) works within the context of the "deterministic equivalent problem",

$$\begin{aligned} \text{Min} \quad & c^\top x + \sum_{\omega \in \Omega} p_\omega (g_u^\top u_\omega + g_z^\top z_\omega) \\ \text{s.t.} \quad & T(\omega)x + W_u u_\omega + W_z z_\omega = r(\omega) \qquad \forall \omega \in \Omega \\ & x \in \Re_+^{n_1},\ u_\omega \in \Re_+^{n_u},\ z_\omega \in \mathcal{B}^{n_z} \qquad \forall \omega \in \Omega. \end{aligned}$$

Accordingly, Carøe (1998)'s cuts may be translated from one scenario to another, while being restricted to the higher dimension of (x, u_ω, z_ω) as compared to the (u_ω, z_ω) dimension restriction of the Sen and Higle (2000) cuts. It follows that these cuts permit both a temporal and scenario decomposition (i.e., w.r.t to x and ω), while Carøe (1998)'s are restricted to only scenario decomposition (i.e., w.r.t ω). Another recent paper, Sherali and Fraticelli (2002) also uses cuts in this higher dimensional space. Their approach uses the reformulation-linearization technique (Sherali and Adams, 1990) to construct their approximation.

3.1.2 Convexification of the Right-Hand-Side Function

As discussed in Sen and Higle (2000), the function $\pi_0(x,\omega)$ is piecewise linear and concave in the first argument. That is,

$$\pi_0(x,\omega) = \underset{h \in H}{\text{Min}} \{\bar{\nu}_h(\omega) - \bar{\gamma}_h(\omega)^\top x\}$$

for a specified collection $\{\bar{\nu}_h(\omega), \bar{\gamma}_h(\omega)\}_{h \in H}$. Consequently, it is necessary to develop a convexification of the function in order to facilitate the solution of the lower bounding approximation (3.4). This is accomplished using reverse convex programming techniques, in which disjunctive programming concepts are used to obtain the convex hull of reverse convex sets (Sen and Sherali, 1987).

To begin, let the epigraph of $\pi_0(\cdot, \omega)$, restricted to $x \in X$ be defined as

$$\Pi_X(\omega) = \{(\theta, x) \mid x \in X, \theta \geq \pi_0(x, \omega)\},$$

where X is a polyhedral set such that

$$X = \{x \in \Re^{n_1}_+ \mid Ax \geq b\},$$

where $A \in \Re^{m_1 \times n_1}$ and $b \in \Re^{m_1}$.

Also let

$$E_h(\omega) = \{(\theta, x) \mid \theta \geq \bar{\nu}_h(\omega) - \bar{\gamma}_h(w)^\top x, Ax \geq b, x \geq 0\}. \tag{3.5}$$

Then $\Pi_X(\omega)$ can be defined in disjunctive normal form as

$$\Pi_X(\omega) = \bigcup_{h \in H} E_h(\omega).$$

Thus the epigraph of function π_0 can be represented as the union of half-spaces, which is a disjunctive set. In order to convexify this set, the notion of reverse polars from the theory of disjunctive programming (Balas, 1979) is applied. These sets (reverse polars) characterize the set of all valid inequalities of a disjunctive set, with the extreme points providing facets of the (closure of the) convex hull of the disjunctive set. The specific construction that is adopted is provided below, and will be referred to as the epi-reverse polar because it represents the reverse polar of the epigraph of π_0.

In the following, it is assumed that for all $x \in X$, $\theta \geq 0$ in (3.5). As long as X is bounded, there is no loss of generality with this assumption, because the epigraph can be translated to ensure that $\theta \geq 0$. The epi-reverse polar of this set, $\Pi^\dagger_X(\omega)$, is

defined as

$$\begin{aligned}\Pi_X^{\dagger}(\omega) = \big\{ & \sigma_0(\omega) \in \Re, \sigma(\omega) \in \Re^{n_1}, \delta(\omega) \in \Re \text{ such that} \\ & \forall h \in H, \ \exists\ \tau_h \in \Re^{m_1}, \tau_{0h} \in \Re \\ & \sigma_0(\omega) \geq \tau_{0h} \quad \forall h \in H \\ & \sum_h \tau_{0h} = 1 \\ & \sigma_j(\omega) \geq \tau_h^{\top} A_j + \tau_{0h} \bar{\gamma}_{hj}(\omega) \quad \forall h \in H, j = 1, ..., n_1 \\ & \delta(\omega) \leq \tau_h^{\top} b + \tau_{0h} \bar{\nu}_h(\omega) \quad \forall h \in H \\ & \tau_h \geq 0, \tau_{0h} \geq 0, \quad \forall h \in H \big\}. \end{aligned} \tag{3.6}$$

Note that the epi-reverse polar only allows those facets of the convex hull of $\Pi_X(\omega)$ that have positive coefficient for the variable θ. If $\{\bar{\nu}, \bar{\gamma}\}$ are given then

$$\begin{aligned}\text{conv}\big(\Pi_X(\omega)\big) = \big\{ (\theta, x) \ \mid \ & \forall (\sigma_0(\omega), \sigma(\omega), \delta(\omega)) \in \Pi_X^{\dagger}(\omega) \\ & x \in X, \theta \geq (\frac{\delta(\omega)}{\sigma_0(\omega)}) - (\frac{\sigma^{\top}(\omega)}{\sigma_0(\omega)}) x \big\}. \end{aligned}$$

Let $(\sigma_0^i(\omega), \sigma^i(\omega), \delta^i(\omega))_{i \in \mathcal{I}}$ denote the set of extreme points of the epi-reverse polar, and let $\nu^i(\omega) = \frac{\delta^i(\omega)}{\sigma_0^i(\omega)}$ and $\gamma^i(\omega) = \frac{\sigma^i(\omega)}{\sigma_0^i(\omega)}$. For each $(x, \omega) \in X \times \Omega$, let $\pi_c(x, \omega) = \text{Max}_{i \in \mathcal{I}} \{\nu_i(\omega) - \gamma_i(\omega)^{\top} x\}$. Then for each $\omega \in \Omega$, $\pi_c(\cdot, \omega)$ is a convex function. Moreover, the epigraph of $\pi_c(x, \omega)$ restricted to X is the closure of the convex hull of $\Pi_X(\omega)$. We refer to $\pi_c(\cdot, \omega)$ as the convex hull approximation of $\pi_0(\cdot, \omega)$, and note that $\pi_0(x, \omega) = \pi_c(x, \omega)$ whenever x is an extreme point of X.

3.1.3 An Algorithmic Context for the C^3 Theorem

As a preview of the D^2 algorithm, let us consider the scenario subproblems in a temporal decomposition of the SMIP, (3.1). If we let x^k denote the first-stage solution associated with the k^{th} algorithmic iteration, the subproblems are of the

form:

$$\begin{aligned} f^k(x,\omega) = \text{Min} \quad & g^\top y \\ \text{s.t.} \quad & W^k y \geq r^k(\omega) - T^k(\omega)x \\ & y \in \Re^{n_2}_+ \end{aligned} \tag{3.7}$$

Of course, in the first iteration we have

$$\begin{aligned} f^1(x,\omega) = \text{Min} \quad & g^\top y \\ \text{s.t.} \quad & Wy \geq r(\omega) - T(\omega)x \\ & y \in \Re^{n_2}_+, \end{aligned}$$

the LP relaxation of (3.2). Thus, the problem is initialized with $W^1 = W$, $r^1(\omega) = r(\omega)$, and $T^1(\omega) = T(\omega)$ as in (3.2). As iterations progress, cutting planes of the form

$$\pi^k y \geq \pi_c(x^k,\omega) = \underset{i\in I}{\text{Max}}\{\nu_i(\omega) - \gamma_i(\omega)^\top x^k\}$$

are added to the subproblem, thereby refining the approximation of the convex hull of integer solutions. As such, the vector π^k is appended to the matrix W^{k-1}, and the element identified $(\nu_i(\omega), \gamma_i(\omega))$ is appended to $\left(r^{k-1}(\omega), T^{k-1}(\omega)\right)$. Let $y^k(\omega) \in \text{argmin}\{g^\top y \mid W^k y \geq r^k(\omega) - T^k(\omega)x^k,\ y \in \Re^{n_2}_+\}$. If $z^k(\omega)$, the value assigned to integer variables in $y^k(\omega)$ is integer for all ω, then no update is necessary, and $W^{k+1} = W^k, r^{k+1}(\omega) = r^k(\omega)$, and $T^{k+1}(\omega) = T^k(\omega)$.

On the other hand, suppose that the subproblems do not yield integer optimal solutions. Let $j(k)$ denote an index j, for which $z^k_j(\omega)$ is non-integer for some $\omega \in \Omega$, and let $\bar{z}_{j(k)}$ denote one of the non-integer values $\{z^k_j(\omega)\}_{\omega\in\Omega}$. To eliminate this non-integer solution, a disjunction of the following form may be used:

$$\mathcal{S}_k(x^k,\omega) = \mathcal{S}_{0,j(k)}(x^k,\omega) \bigcup \mathcal{S}_{1,j(k)}(x^k,\omega)$$

where

$$\mathcal{S}_{0,j(k)}(x^k,\omega) = \left\{y \in \Re^{n_2}_+ \mid W^k y \geq r^k(\omega) - T^k(\omega)x^k \right. \tag{3.8a}$$

$$\left. -z_{j(k)} \geq -\lfloor \bar{z}_{j(k)} \rfloor \right\} \tag{3.8b}$$

and

$$\mathcal{S}_{1,j(k)}(x^k,\omega) = \{y \in \Re^{n_2}_+ \mid W^k y \geq r^k(\omega) - T^k(\omega)x^k \tag{3.9a}$$

$$z_{j(k)} \geq \lceil \bar{z}_{j(k)} \rceil \tag{3.9b}$$

The index $j(k)$ is referred to as the "disjunction variable" for iteration k. Our assumptions ensure that the subproblems remain feasible for any restriction of the integer variables, and thus both (3.8) and (3.9) are non-empty. Also, since the disjunction is based on an either-or-condition, $H = \{0,1\}$ is used. It should be noted that when the integer restrictions are binary, the right hand side of (3.8b) is zero, and the right hand side of (3.9b) is one. This is precisely the disjunction used in lift-and-project cuts of Balas et al. (1993).

Let $\lambda_{0,1}$ denote the vector of multipliers associated with (3.8a), and $\lambda_{0,2}$ denote the scalar multiplier associated with (3.8b). Let $\lambda_{1,1}$ and $\lambda_{1,2}$ be similarly defined for (3.9a) and (3.9b), respectively. Assuming that the sets defined in (3.8) and (3.9) are non-empty for all $\omega \in \Omega$, the following problem may be used to generate the common cut coefficients, π^k, in iteration k:

$$\begin{aligned}
\text{Max} \quad & E[\pi_0(\tilde{\omega})] - E[y^k(\tilde{\omega})]\pi \\
\text{s.t.} \quad & \pi_j \geq \lambda_{0,1}^\top W_j^k - I_j^k \lambda_{0,2} \ \forall j \\
& \pi_j \geq \lambda_{1,1}^\top W_j^k + I_j^k \lambda_{1,2} \ \forall j \\
& \pi_0(\omega) \leq \lambda_{0,1}^\top \left(r^k(\omega) - T^k(\omega)x^k\right) - \lambda_{0,2}^\top \lfloor \bar{z}_{j(k)} \rfloor \ \forall \omega \in \Omega \\
& \pi_0(\omega) \leq \lambda_{1,1}^\top \left(r^k(\omega) - T^k(\omega)x^k\right) + \lambda_{1,2}^\top \lceil \bar{z}_{j(k)} \rceil \ \forall \omega \in \Omega \\
& -1 \leq \pi_j \leq 1, \ \forall j, \ -1 \leq \pi_0(\omega) \leq 1, \ \forall \omega \in \Omega \\
& \lambda_{0,1}, \ \lambda_{0,2}, \ \lambda_{1,1}, \ \lambda_{1,2} \geq 0
\end{aligned} \tag{3.10}$$

where $I_j^k = \begin{cases} 0, & if \ j \neq j(k) \\ 1, & otherwise. \end{cases}$

In subsequent chapters we shall refer to problem (3.10) as C^3-SLP. The validity of the cut coefficients generated above follows from the disjunctive cut principle (Balas, 1979) which requires the multipliers (λ) to be chosen in such a way that the cut coefficients are greater than the aggregated columns as specified above. Since the coefficients π are independent of ω, the above LP generates common cut coefficients. This LP/SLP is formulated following the standard approach of generating valid inequalities in disjunctive programming (Sherali and Shetty, 1980), and it optimizes some measure of distance of the current solution $y^k(\omega)$ from the cut. It is interesting to note that this problem is a simple recourse problem, and may be interpreted as a stochastic version of the linear program used to generate the lift-and-project cuts.

Since the disjunction used for cut formation has $H = \{0, 1\}$, the epigraph $\pi_0(x, \omega)$ is a union of two polyhedral sets. Therefore, for each $\omega \in \Omega$, the following parameters are derived from an optimal solution of (3.10),

$$\bar{\nu}_0^k(\omega) = \lambda_{0,1}^\top r^k(\omega) - \lambda_{0,2} \lfloor \bar{z}_{j(k)} \rfloor,$$

$$\bar{\nu}_1^k(\omega) = \lambda_{1,1}^\top r^k(\omega) + \lambda_{1,2} \lceil \bar{z}_{j(k)} \rceil,$$

and

$$[\bar{\gamma}_h(\omega)]^\top = \lambda_{h,1}^\top T^k(\omega), \ \forall h \in H$$

are used to update the approximation of the polyhedron defined via (3.6), which we denote as $\Pi_X^\dagger(\omega)^k$. This polyhedron represents the epi-reverse polar, which provides access to the convexification of π_0. Correspondingly, for each $\omega \in \Omega$, the following LP is used to approximate $\pi_0(x, \omega)$:

$$\begin{aligned} \text{Max} \quad & \delta(\omega) - \sigma_0(\omega) - (x^k)^\top \sigma(\omega) \\ \text{s.t.} \quad & (\sigma_0(\omega), \sigma(\omega), \delta(\omega)) \in (\Pi_X^\dagger(\omega))^k \end{aligned} \tag{3.11}$$

In subsequent chapters we shall refer to problem (3.11) as RHS-LP. With an optimal solution to (3.11), $(\sigma_0^k(\omega), \sigma^k(\omega), \delta^k(\omega))$, we obtain $\nu^k(\omega) = \frac{\delta^k(\omega)}{\sigma_0^k(\omega)}$ and $\gamma^k(\omega) = \frac{\sigma^k(\omega)}{\sigma_0^k(\omega)}$. For each $\omega \in \Omega$, these coefficients are used to update the right-hand-side functions $r^{k+1}(\omega) = [r^k(\omega), \nu^k(\omega)]$, and $T^{k+1}(\omega) = [T^k(\omega)^\top; \gamma^k(\omega)]^\top$. Similarly, the solution to (3.10) is used to update the constraint matrix, $W^{k+1} = [(W^k)^\top; \pi^k]^\top$.

The master program is defined as:

$$\begin{aligned} \text{Min} \quad & c^\top x + F^k(x) \\ \text{s.t.} \quad & Ax \geq b \\ & x \in X \cap \mathcal{B} \end{aligned} \tag{3.12}$$

where $F^k(\cdot)$ is a piecewise linear approximation of the subproblem objective function, $E[f(x, \tilde{\omega})]$ in the k^{th} iteration.

3.1.4 Disjunctive Decomposition with Set Convexification

The Basic D^2 Algorithm (Sen and Higle, 2000) can be stated as follows:

Basic D^2 Algorithm

0. Initialize. $V_1 \leftarrow \infty$. $\epsilon > 0$ and $x^1 \in X$ are given. $k \leftarrow 1$, $W^1 \leftarrow W$, $T^1(\omega) \leftarrow T(\omega)$, and $r^1(\omega) = r(\omega)$.

1. Solve one LP Subproblem for each $\omega \in \Omega$ For each $\omega \in \Omega$, use the matrix W^k as well as the right hand side vector $r^k(\omega) - T^k(\omega)x^k$ to solve (3.7). If $\{y^k(\omega)\}_{\omega \in \Omega}$ satisfy the integer restrictions,

$V_{k+1} \leftarrow \text{Min}\{c^\top x^k + E[f(x^k, \tilde{\omega})], V_k\}$, and go to step 4.

2. Solve Multiplier/Cut Generation LP/SLP and Perform Updates Choose a disjunction variable $j(k)$.

(i) Formulate and solve (3.10) to obtain π^k and define $W^{k+1} = [(W^k)^\top; \pi^k]^\top$.

(ii) Using the multipliers λ_0^k, λ_1^k and the value $\tilde{z}_{j(k)}$ obtained in (i) solve (3.11) for each outcome ω. The solution defines $\nu^k(\omega)$ and $\gamma^k(\omega)$ which are used to

update the right hand side functions: $r^{k+1}(\omega) = [r^k(\omega),\ \nu^k(\omega)]$ and $T^{k+1}(\omega) = [T^k(\omega)^\top; \gamma^k(\omega)]^\top$.

3. Update and Solve one LP Subproblem for each $\omega \in \Omega$ For each $\omega \in \Omega$ solve (3.7) using W^{k+1} and $r^{k+1}(\omega) - T^{k+1}(\omega)x^k$. If $y^k(\omega)$ satisfies the integer restrictions for all $\omega \in \Omega$, $V_{k+1} \leftarrow \text{Min}\{c^\top x^k + E[f(x^k, \omega)], V_k\}$. Otherwise, $V_{k+1} \leftarrow V_k$.

4. Update and Solve the Master Problem Using the dual multipliers from the most recently solved subproblem for each $\omega \in \Omega$ (either step 1 or step 3), update the approximation F^k by adopting a standard decomposition method (e.g Benders (1962)). Let $x^{k+1} \in \text{argmin}\{c^\top x + F^k(x) \mid x \in X\}$, and let v_k denote the optimal value of the master problem. If $V_k - v_k \leq \epsilon$, stop. Otherwise, $k \leftarrow k+1$ and repeat from step 1.

3.2 An Illustration of the D^2 Algorithm

Consider the following two-stage SIP example problem with two scenarios:

$$\begin{aligned} \text{Min} \quad & -1.5x_1 - 4x_2 + E[f(x_1, x_2, \omega)] \\ \text{s.t.} \quad & x_1, x_2 \in \{0, 1\} \end{aligned}$$

where,

$$\begin{aligned} f(x_1, x_2, \omega) = \ \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 \\ \text{s.t.} \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 \geq -\omega^1 + x_1 \\ & -6y_1 - 1y_2 - 3y_3 - 2y_4 \geq -\omega^2 + x_2 \\ & y_1, y_2, y_3, y_4 \in \{0, 1\} \end{aligned}$$

The first-stage variables are $x = [x_1, x_2]^\top$, while the second-stage variables are $y = [y_1, y_2, y_3, y_4]^\top$. There are two scenarios $\omega_1 = [5, 2]^\top$ and $\omega_2 = [10, 3]^\top$, each occurring with probability 0.5. This instance is motivated by the example in Schultz

et al. (1998), where the second-stage involves general integer variables. To ensure that the subproblems remain feasible for any restriction on the integer variables, we include an artificial variable, denoted R, which is penalized in the objective at a rate of 100. Thus, we recast the problems as

$$\begin{aligned} \text{Min} \quad & -1.5x_1 - 4x_2 + 0.5f_1(x_1, x_2, \omega_1) + 0.5f_2(x_1, x_2, \omega_2) \\ \text{s.t.} \quad & x_1, x_2 \in \{0, 1\} \end{aligned}$$

where

$$\begin{aligned} f_1(x_1, x_2, \omega_1) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\ \text{s.t.} \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -5 + x_1 \\ & -6y_1 - 1y_2 - 3y_3 - 2y_4 + R \geq -2 + x_2 \\ & y_1, y_2, y_3, y_4 \in \{0, 1\}, \ R \geq 0 \end{aligned}$$

and

$$\begin{aligned} f_2(x_1, x_2, \omega_2) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\ \text{s.t.} \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -10 + x_1 \\ & -6y_1 - 1y_2 - 3y_3 - 2y_4 + R \geq -3 + x_2 \\ & y_1, y_2, y_3, y_4 \in \{0, 1\}, \ R \geq 0 \end{aligned}$$

In this problem we have the following input data:

$$A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix},$$

$$b = \begin{bmatrix} -1 \\ -1 \end{bmatrix},$$

$$W = \begin{bmatrix} -2 & -3 & -4 & -5 & 1 \\ -6 & -1 & -3 & -2 & 1 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix},$$

$$T(\omega) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ for both scenarios,}$$

$$r(\omega_1) = \begin{bmatrix} -5 \\ -2 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

$$\text{and } r(\omega_2) = \begin{bmatrix} -10 \\ -3 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}.$$

Note that with A and b as defined above, binary solutions are extreme points of $X = \{x : Ax \geq b, x \geq 0\}$, as required. We can now start the D^2 algorithm.

Iteration 1 ($k = 1$)

Step 0

The D^2 algorithm is initialized with the following master program:

$$\begin{aligned} \text{Min} \quad & -1.5x_1 - 4x_2 \\ \text{s.t.} \quad & -x_1 \geq -1 \\ & -x_2 \geq -1 \\ & x_1, x_2 \in \{0, 1\}. \end{aligned}$$

The initial master program yields $x^1 = [1, 1]^\top$. The upper and lower bounds are initialized as $V_0 = \infty$ and $v_0 = -5.5$, respectively. For the first iteration of the algorithm we set $V_1 = V_0$, $W^1 = W$, $T^1(\omega) = T(\omega)$, and $r^1(\omega) = r(\omega)$.

Step 1

For step 1 of the algorithm we use $x_1 = 1, x_2 = 1$ and solve the linear relaxation of the second-stage subproblem for ω_1 and ω_2, which we call LP_1 and LP_2, respectively:

$$\begin{aligned} LP_1 : f_1(1, 1, \omega_1) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\ \text{s.t.} \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -4 \\ & -6y_1 - 1y_2 - 3y_3 - 2y_4 + R \geq -1 \\ & -y_1 \geq -1 \\ & -y_2 \geq -1 \\ & -y_3 \geq -1 \\ & -y_4 \geq -1 \\ & y_1, y_2, y_3, y_4, R \geq 0. \end{aligned}$$

and

$$
\begin{aligned}
LP_2 : f_2(1,1,\omega_2) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
\text{s.t.} \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -9 \\
& -6y_1 - 1y_2 - 3y_3 - 2y_4 + R \geq -2 \\
& -y_1 \geq -1 \\
& -y_2 \geq -1 \\
& -y_3 \geq -1 \\
& -y_4 \geq -1 \\
& y_1, y_2, y_3, y_4, R \geq 0.
\end{aligned}
$$

The optimal solution for LP_1 is $y(\omega_1) = [0,1,0,0]^\top$, $R(\omega_1) = 0$. and for LP_2 is $y(\omega_2) = [0,1,0,0.5]^\top$, $R(\omega_2) = 0$.

Step 2

Since $y(\omega_2)$ does not satisfy the integer restrictions, we choose y_4 as the "disjunction variable" and create the disjunction $y_4 \leq 0$ or $y_4 \geq 1$ for LP_2. We formulate (3.10), which yields the vector π^1 for updating W^1 and the data for (3.11) whose optimal solution is used to update the right-hand side of the second-stage constraints. An optimal solution for (3.10) is $\pi^1 = [1,-1,1,-1,1]^\top$, $\lambda_{0,1} = [0,0,0,1,0,0]^\top$, $\lambda_{0,2} = 1$, $\lambda_{1,1} = [0,1,0,0,0,0]^\top$, and $\lambda_{1,2} = 1$. We obtain W^2 by appending π^1 to W^1: $W^2 = \begin{bmatrix} & [W^1] & & & \\ 1 & -1 & 1 & -1 & 1 \end{bmatrix}$. Using the solution from (3.10), we formulate and solve (3.11) for both ω_1 and ω_2. The optimal solution for ω_1 is $\delta(\omega_1) = -0.5$, $\sigma_0(\omega_1) = 0.5$, and $\sigma(\omega_1) = [0,0]^\top$. Based on this solution we update $r^1(\omega_1)$ and $T^1(\omega_1)$ as follows: $r^2(\omega_1) = \begin{bmatrix} [r^1(\omega_1)] \\ -1 \end{bmatrix}$, $T^2(\omega_1) = \begin{bmatrix} [T^1(\omega_1)] & \\ 0 & 0 \end{bmatrix}$. For ω_2 the optimal solution of LP (3.11) is $\delta(\omega_2) = -1$, $\sigma_0(\omega_2) = 0.5$, and $\sigma(\omega_2) = [0,-0.5]^\top$. Similarly, we update $r^1(\omega_2)$ and $T^1(\omega_2)$ as follows: $r^2(\omega_2) = \begin{bmatrix} [r^1(\omega_2)] \\ -2 \end{bmatrix}$, $T^2(\omega_2) = \begin{bmatrix} [T^1(\omega_2)] & \\ 0 & -1 \end{bmatrix}$.

This completes Step 2 of the algorithm.

Step 3

Solving (3.7), we obtain $y(\omega_1) = [0, 1, 0, 0]^\top$, $R(\omega_1) = 0$, and $y(\omega_2) = [0, 1, 0.2, 0.2]^\top$, $R(\omega_2) = 0$. The dual solutions are
$d(\omega_1) = [0, 14, 0, 0, 5, 0, 0]^\top$ and $d(\omega_2) = [0, 10.2, 7.6, 0, 1.2, 0, 0]^\top$. $V_2 \leftarrow V_1$, because the integer restrictions are not satisfied.

Step 4

Using the dual solution for each subproblem from Step 3, we formulate the "optimality cuts" as in Benders' decomposition (Benders, 1962). The resulting cuts are $0x_1 - 14x_2 + \theta_1 \geq -33$ for ω_1 and $0x_1 - 17.8x_2 + \theta_2 \geq -47$ for ω_2. Since the two scenarios are equally likely, the expected values associated with the cut coefficients yield $0x_1 - 15.9x_2 + \theta \geq -40$ as the optimality cut to add to the master program:

$$\begin{aligned} \text{Min} \quad & -1.5x_1 - 4x_2 + \theta \\ \text{s.t.} \quad & -x_1 \geq -1 \\ & -x_2 \geq -1 \\ & 0x_1 - 15.9x_2 + \theta \geq -40 \\ & x_1, x_2 \in \{0, 1\}. \end{aligned}$$

Solving the master program we get $x^2 = [1, 0]^\top$, $\theta = -40$ and an objective value of -41.5. Therefore, the lower bound becomes $v_2 = -41.5$. The upper bound remains the same, $V_2 = V_1 = \infty$, as before. This completes the first iteration of the algorithm. Since $V_2 > v_2$ $k \leftarrow 2$, and we begin the next iteration.

Iteration 2

Step 1

We start the second iteration by solving the following updated subproblems with $x^2 = [1, 0]^\top$:

$$
\begin{aligned}
LP_1 : f_1(1, 0, \omega_1) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
\text{s.t.} \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -4 \\
& -6y_1 - y_2 - 3y_3 - 2y_4 + R \geq -2 \\
& y_1 - y_2 - y_3 - y_4 + R \geq -1 \\
& -y_1 \geq -1 \\
& -y_2 \geq -1 \\
& -y_3 \geq -1 \\
& -y_4 \geq -1 \\
& y_1, y_2, y_3, y_4, R \geq 0.
\end{aligned}
$$

and

$$
\begin{aligned}
LP_2 : f_2(1, 0, \omega_2) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
\text{s.t.} \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -9 \\
& -6y_1 - y_2 - 3y_3 - 2y_4 + R \geq -3 \\
& y_1 - y_2 - y_3 - y_4 + R \geq -2 \\
& -y_1 \geq -1 \\
& -y_2 \geq -1 \\
& -y_3 \geq -1 \\
& -y_4 \geq -1 \\
& y_1, y_2, y_3, y_4, R \geq 0.
\end{aligned}
$$

The optimal solution for LP_1 is $y(\omega_1) = [0.108108, 1, 0.027027, 0.135135]^\top$ and for LP_2 is $y(\omega_2) = [0, 1, 0, 1]^\top$.

Step 2

Since $y(\omega_1)$ does not satisfy the integer restrictions, we choose y_4 as the "disjunction variable" and create the disjunction $y_4 \leq 0$ or $y_4 \geq 1$ for LP_1. We formulate and solve (3.10), which yields the data used to update W^1 and to formulate (3.11) whose optimal solution is used to update the right-hand side of the second-stage constraints. Solving LP (3.10) we obtain $\pi^2 = [0, -0.5, 0, -0.5, 1]^\top$, $\lambda_{0,1} = [0, 0, 0, 0, 0.5, 0, 0]^\top$, $\lambda_{0,2} = 0.5$, $\lambda_{1,1} = [0, 0.125, 0, 0.375, 0, 0, 0]^\top$, and $\lambda_{1,2} = 0.125$. We obtain W^3 by appending π^2 to W^2: $W^3 = \begin{bmatrix} & [W^2] & \\ 0 & -0.5 \quad 0 \quad -0.5 & 1 \end{bmatrix}$. Using the solution of (3.10) we formulate and solve (3.11) for both ω_1 and ω_2. The optimal solution for ω_1 is $\delta(\omega_1) = -0.25$, $\sigma_0(\omega_1) = 0.5$, and $\sigma(\omega_1) = [0, 0]^\top$. Based on this solution we update $r^2(\omega_1)$ and $T^2(\omega_1)$ as follows: $r^3(\omega_1) = \begin{bmatrix} [r^2(\omega_1)] \\ -0.5 \end{bmatrix}$, $T^3(\omega_1) = \begin{bmatrix} [T^2(\omega_1)] & \\ 0 & 0 \end{bmatrix}$.
For ω_2 the optimal solution of LP (3.11) is $\delta(\omega_2) = -0.5$, $\sigma_0(\omega_2) = 0.5$, and $\sigma(\omega_2) = [0, 0]^\top$. Similarly, we update $r^2(\omega_2)$ and $T^2(\omega_2)$ as follows: $r^3(\omega_2) = \begin{bmatrix} [r^2(\omega_2)] \\ -1 \end{bmatrix}$, $T^3(\omega_2) = \begin{bmatrix} [T^2(\omega_2)] & \\ 0 & 0 \end{bmatrix}$. This completes Step 2 of the algorithm.

Step 3

Solving (3.7) we obtain $y(\omega_1) = [0.055556, 1, 0.22222, 0]^\top$, $R = 0$, and $y(\omega_2) = [0, 1, 0, 1]^\top$, $R = 0$. The dual solutions are $d(\omega_1) = [5, 1, 0, 2, 0, 2, 0, 0]^\top$ and $d(\omega_2) = [0, 7.667, 0, 0, 0, 11.333, 0, 12.667]^\top$. $V_3 \leftarrow V_2$ because the integer restrictions have not been met.

Step 4

Using the dual solution for each subproblem from step 3, we formulate the "optimality cuts" as in Benders' decomposition (Benders, 1962). The resulting cuts are $-5x_1 - 1x_2 + \theta_1 \geq -30$ for ω_1 and $0x_1 - 7.667x_2 + \theta_2 \geq -47$ for ω_2. The expected

value associated with the cut coefficients yields $-2.5x_1 - 4.334x_2 + \theta \geq -38.5$ as the optimality cut to add to the master program:

$$\begin{aligned} \text{Min} \quad & -1.5x_1 - 4x_2 + \theta \\ \text{s.t.} \quad & -x_1 \geq -1 \\ & -x_2 \geq -1 \\ & 0x_1 - 15.9x_2 + \theta \geq -40 \\ & -2.5x_1 - 4.334x_2 + \theta \geq -38.5 \\ & x_1, x_2 \in \{0, 1\}. \end{aligned}$$

Solving the master program we get $x^3 = [0, 0]^\top$, $\theta = -38.5$ and an objective value of -38.5. Therefore, the lower bound becomes $v_2 = -38.5$. $k \leftarrow 3$, and we begin the next iteration.

Iteration 3

Step 1

We start the third iteration by solving the following updated subproblems with

$x^3 = [0,0]^\top$:

$$
\begin{aligned}
LP_1 : f_1(0,0,\omega_1) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
\text{s.t.} \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -5 \\
& -6y_1 - y_2 - 3y_3 - 2y_4 + R \geq -2 \\
& y_1 - y_2 - y_3 - y_4 + R \geq -1 \\
& 0y_1 - 0.5y_2 - 0y_3 - 0.5y_4 + R \geq -0.5 \\
& -y_1 \geq -1 \\
& -y_2 \geq -1 \\
& -y_3 \geq -1 \\
& -y_4 \geq -1 \\
& y_1, y_2, y_3, y_4, R \geq 0.
\end{aligned}
$$

and

$$
\begin{aligned}
LP_2 : f_2(0,0,\omega_2) = \quad \text{Min} \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\
\text{s.t.} \quad & -2y_1 - 3y_2 - 4y_3 - 5y_4 + R \geq -10 \\
& -6y_1 - y_2 - 3y_3 - 2y_4 + R \geq -3 \\
& y_1 - y_2 - y_3 - y_4 + R \geq -2 \\
& 0y_1 - 0.5y_2 - 1y_3 - 0.5y_4 + R \geq -1 \\
& -y_1 \geq -1 \\
& -y_2 \geq -1 \\
& -y_3 \geq -1 \\
& -y_4 \geq -1 \\
& y_1, y_2, y_3, y_4, R \geq 0.
\end{aligned}
$$

The optimal solution for LP_1 is $y(\omega_1) = [0,0,0,1]^\top$, $R(\omega_1) = 0$ and for LP_2 is $y(\omega_2) = [0,1,0,1]^\top$, $R(\omega_2) = 0$. The dual solutions are $d(\omega_1) =$

$[0, 7.6667, 0, 25.333, 0, 0, 0, 0]^\top$ and $d(\omega_2) = [0, 7.667, 0, 0, 0, 11.333, 0, 12.667]^\top$. We now have an incumbent integer solution $x = [0, 0]^\top$, $y(\omega_1) = [0, 0, 0, 1]^\top$, $y(\omega_2) = [0, 1, 0, 1]^\top$, $\theta = 0.5(-28) + 0.5(-47) = -37.5$. $V_4 \leftarrow \text{Min}\{-37.5, V_3\} = -37.5$. We go to step 4 of the algorithm.

Step 4

Using the dual solution for each subproblem from Step 3, we formulate the "optimality cuts" as in Benders' decomposition (Benders, 1962). The resulting cuts are $0x_1 - 7.667x_2 + \theta_1 \geq -28$ for ω_1 and $0x_1 - 7.667x_2 + \theta_2 \geq -47$ for ω_2, and the expected values yield $0x_1 - 7.667x_2 + \theta \geq -37.5$. as the optimality cut to add to the master program:

$$
\begin{aligned}
\text{Min} \quad & -1.5x_1 - 4x_2 + \theta \\
\text{s.t.} \quad & -x_1 \geq -1 \\
& -x_2 \geq -1 \\
& 0x_1 - 15.9x_2 + \theta \geq -40 \\
& -2.5x_1 - 4.334x_2 + \theta \geq -38.5 \\
& 0x_1 - 7.667x_2 + \theta \geq -37.5 \\
& x_1, x_2 \in \{0, 1\}.
\end{aligned}
$$

Solving the master program we get $x^4 = [1, 0]^\top$, $\theta = -36$ and an objective value of -37.5. Therefore, the lower bound becomes $v_3 = -37.5$. Since the upper bound ($V_3 = -37.5$) and the lower bound are now equal, the algorithm terminates and we have an optimal solution: $x = [0, 0]^\top, y(\omega_1) = [0, 0, 0, 1]^\top, y(\omega_2) = [0, 1, 0, 1]^\top$ and objective value -37.5. It so happens that both $[0, 0]^\top$ and $[1, 0]^\top$ are optimal for the master problem, but optimality can only be concluded for the point $[0, 0]^\top$, since that is the incumbent.

It is interesting to note that while the cuts used in LP_1 and LP_2 in iteration 3 were obtained at $x^1 = [1, 1]^\top$ and $x^2 = [1, 0]^\top$, integer solutions ($y(\omega_1) = [0, 0, 0, 1]^\top$ and $y(\omega_2) = [0, 1, 0, 1]^\top$) were obtained with the first relaxations solved at $x^3 =$

$[0,0]^\top$. The credit for this should go to the C^3 Theorem.

3.3 Summary

This chapter has presented the main results on set convexifications for large scale Stochastic Integer Programming and has given an illustration of the decomposition method called the D^2 algorithm. At the heart of this method is the C^3 Theorem, which allows both a temporal and scenario decomposition of the SMIP. We have used a simple example to illustrate the application of the D^2 algorithm. In this example the D^2 algorithm converges to an optimal solution in three iterations. The example clearly illustrates how the second-stage convexifications are sequentially carried out and how they impact the first-stage objective function.

The primary focus in this chapter is the generation of cutting planes within a temporal decomposition of two-stage SMIPs. We note, however, that cutting planes alone are typically inadequate for solving large mixed-integer programs. Thus, our ultimate goal is to use cuts such as those discussed in this chapter within a Branch-and-Cut (BAC) setting, where a careful generation of cuts is necessary to further enhance the success of BAC-type algorithms for solving SMIP problems. Therefore, the next chapter describes an algorithm in which the D^2 algorithm is incorporated into a branch-and-cut setting.

CHAPTER 4

COMPUTER IMPLEMENTATION OF THE D^2 ALGORITHM AND RELATED DECOMPOSITION ALGORITHMS

In this chapter we present details of a computer implementation of the D^2 algorithm derived by Sen and Higle (2000). Due to the dynamic nature of the various parts of the algorithm, implementation of such an algorithm is not an easy undertaking, but one that requires careful coding. The issues associated with the implementation are discussed and parts of the algorithm are illustrated using pseudo-code fragments. Two other algorithms are also implemented and briefly discussed. These are the disjunctive decomposition with branch-and-cut or D^2-BAC algorithm for two-stage SMIP presented by Sen and Sherali (2004), and the first decomposition algorithm for SMIP presented by Laporte and Louveaux (1993). We refer to the this algorithm as the L^2 algorithm.

The rest of the chapter is organized as follows. In the next section the algorithmic setting of the D^2 algorithm is given and in Section 4.2 the D^2 algorithm is formally stated. In Section 4.3 details of the D^2-BAC algorithm are given. The L^2 algorithm is discussed in Section 4.4. Details of a computer implementation of the D^2 algorithm are provided in Section 4.5. Finally the chapter ends with a summary.

4.1 Algorithmic Setting

The D^2 algorithm is derived in Sen and Higle (2000) and illustrated in Sen et al. (2002). In the D^2 algorithm there is a master program that forwards a first-stage decision x^k to the scenario subproblem in iteration k. Given x^k and the convex approximations developed in the previous iterations $k-1$, the k^{th} convex approximations of the scenario subproblems are developed and an updated representation of the second-stage objective function is obtained and passed back to the master program. In algorithmic iteration k the master program takes the following form:

$$\text{Min } c^\top x + \eta \tag{4.1a}$$

$$\text{s.t. } Ax \geq b, \tag{4.1b}$$

$$\beta^t x + \eta \geq \alpha^t, \; t = 1, ..., k, \tag{4.1c}$$

$$x \in \mathcal{B}^{n_1}. \tag{4.1d}$$

The variable η represents the expected recourse function evaluations. Constraints (4.1b) are the first-stage constraints, constraints (4.1c) are the Benders-type (Benders, 1962) optimality cuts, and constraints (4.1d) are the binary restrictions on the first-stage decision variable. In constraint (4.1c) the right-hand side $\alpha^t = E[\psi^t(\omega)^\top r^t(\omega)]$ and $\beta^t = E[\psi^t(\omega)^\top T^t(\omega)]$ for $t = 1, ..., k$. The vector $\psi^t(\omega)$ denotes an appropriately dimensioned vector of dual multipliers associated with the constraints of the LP relaxation of the scenario subproblem, which can be given as follows:

$$f_c^k(x, \omega) = \text{Min } q(\omega)^\top y, \tag{4.2a}$$

$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x, \tag{4.2b}$$

$$(\pi^t)^\top y \geq \pi_c^t(x, \omega), t \in \Theta_k, \tag{4.2c}$$

$$y \in \Re_+^{n_2}. \tag{4.2d}$$

The set Θ_k is the set of disjunctive decomposition cuts or D^2-cuts derived up to iteration k. In order to derive the common-cut-coefficients π a simple recourse SLP (the C^3-SLP, problem 3.10, Ch. 3) is solved. The function $\pi_c(x,\omega)$ is derived by solving another LP (the RHS-LP, problem 3.11, Ch. 3) for each scenario.

Let W^k be a matrix resulting from augmenting $\{(\pi^t)^\top\}_{t\in\Theta_k}$ to matrix W and $\rho_c^k(x,\omega)$ be the result of augmenting $\{\pi_c^t(x,\omega)\}_{t\in\Theta_k}$ to the left hand side $r(\omega) - T(\omega)x$. Thus subproblem (4.2) can be rewritten in compact form as follows:

$$f_c^k(x,\omega) = \text{Min } q(\omega)^\top y, \tag{4.3a}$$

$$\text{s.t. } W^k y \geq \rho_c^k(x,\omega), \tag{4.3b}$$

$$y \in \Re_+^{n_2}. \tag{4.3c}$$

4.2 The D^2 Algorithm

The D^2 algorithm can be stated as follows:

Algorithm D^2

begin

$V_1 \leftarrow \infty$;

choose $\epsilon > 0$;

solve_lps $\leftarrow$ *true*;

$x^1 \leftarrow \text{Min}\{c^\top x | Ax \geq b, x \in X \cap \mathcal{B}\}$;

$k \leftarrow 1$;

$W^1 \leftarrow W$;

$T^1(\omega) \leftarrow T(\omega)$;

$r^1(\omega) = r(\omega)$;

do

for $\omega \leftarrow 1$ to $|\Omega|$

Use matrix W^k, right-hand side vector $r^k(\omega) - T^k(\omega)x^k$
to formulate subproblem (4.3);
if *solve_lps* = *true*;
Solve (4.3) for scenario ω;
Let the objective value be $f^k(x^k, \omega)$;
Store solution $y^k(\omega)$;
else
Solve subproblem MIP for scenario ω;
Let the objective value be $f^k(x^k, \omega)$;
end (if-else)
end(for)
if *solve_lps* = *false* or $\{y^k(\omega)\}_{\omega \leftarrow 1 \text{ to } |\Omega|}$ satisfy the integer restrictions
$V_{k+1} \leftarrow \text{Min}\{c^\top x^k + \sum_{\omega}^{|\Omega|} p_\omega f^k(x^k, \omega), V_k\}$;
else
Choose a disjunction variable $j(k)$ from $\{y^k(\omega)\}_{\omega \leftarrow 1 \text{ to } |\Omega|}$.
Let its floor and ceiling be $\lfloor \bar{y}_{j(k)} \rfloor$ and $\lceil \bar{y}_{j(k)} \rceil$, respectively;
Formulate and solve the C^3-SLP (Problem 3.10, Ch. 3);
Obtain solution π^k, $\lambda^k_{0,1}$, $\lambda^k_{0,2}$, $\lambda^k_{1,1}$ and $\lambda^k_{1,2}$;
Update recourse matrix $W^{k+1} \leftarrow [(W^k)^\top; \pi^k]^\top$;
for $\omega \leftarrow 1$ to Ω
Use the multipliers $\lambda^k_{0,1}$, $\lambda^k_{0,2}$, $\lambda^k_{1,1}$, $\lambda^k_{1,2}$ and the values $\lfloor \bar{y}_{j(k)} \rfloor$ and $\lceil \bar{y}_{j(k)} \rceil$ to formulate and solve the RHS-LP (Problem 3.11, Ch. 3) for scenario ω;
Get solution and compute $\nu^k(\omega)$ and $\gamma^k(\omega)$
and update the right hand side functions:
$r^{k+1}(\omega) \leftarrow [r^k(\omega), \nu^k(\omega)]$ and
$T^{k+1}(\omega) = [T^k(\omega); \gamma^k(\omega)^\top]$;

Solve (4.3) using W^{k+1} and $r^{k+1}(\omega) - T^{k+1}(\omega)x^k$.

Store solution $y^k(\omega)$;

end (for)

if $\{y^k(\omega)\}_{\omega \leftarrow 1 \text{ to } \Omega}$ satisfy the integer restrictions

$V_{k+1} \leftarrow \text{Min}\{c^\top x^k + E[f(x^k, \tilde{\omega})], V_k\}$,

$f^k \leftarrow E[f(x^k, \tilde{\omega})]$;

end (if)

end (if-else)

Derive an optimality cut of type (4.1c) using the dual solutions from the currently solved subproblem LPs (4.3);

Append the cut to the master program (4.1) and solve to get x^{k+1};

Let v_k denote the optimal value of the master problem;

if $x^k = x^{k-1}$

solve_lps $\leftarrow$ *false*;

else

solve_lps $\leftarrow$ *true*;

while $V_k - v_k > \epsilon$;

end (Algorithm)

Note that no explicit branch-and-bound is done in the algorithm. All the MIPs (that is, master problem (4.1) and subproblem MIPs for (4.3)) are given to an MIP solver and are solved exactly. The scenario subproblem MIPs are generally NP-hard problems and may be difficult to solve. In addition, for large scale problems the number of these MIPs can be very large. Therefore, our approach avoids solving subproblem MIPs for upper bounding unnecessarily at every iteration of the algorithm. Instead we only solve subproblem MIPs when the first-stage solution stops changing, usually as the algorithm nears termination.

Subproblem MIP solves can also be initiated when the percent gap between the lower and upper bound remains constant for a preset number of iterations. At all other iterations we solve subproblem LP relaxations (4.3), which are much faster and the sequential addition of "D^2-cuts" to the feasible region of the second-stage problem leads to the convexification of the region leading to potentially integral solutions. To fully close the gap between the lower and the upper bound for certain problem instances, we add the optimality cut of Laporte and Louveaux (1993) stated in the Section 4.4 just after we solve the scenario subproblem MIPs. This usually happens in the final iteration of the algorithm.

4.3 The D^2-BAC Algorithm

We now briefly describe and state the D^2-BAC algorithm. We refer the reader to Sen and Sherali (2004) for the derivation of the algorithm. In this approach the second-stage integer subproblems are solved using a "truncated" branch-and-bound (TB&B) tree, thus allowing for subproblem MIP "partial" solves. Realizing the fact that the subproblems are generally NP-hard, the decomposition method may get bogged down in attempts to solve subproblems to optimality, even while the particular first-stage solution is no where near the neighborhood of an optimal solution. Thus the essence of the D^2-BAC approach is to allow for partial solves of the integer subproblems, so that ultimately the partial solves start to yield optimal solutions.

The fundamental idea in this approach is the observation that a branch-and-bound (B&B) tree together with the LP relaxations at the nodes embodies a disjunction and provides important information that can be used in approximating subproblem MIP value functions. By using the disjunctive principle (Balas, 1979), Sen and Sherali (2004) obtain linear inequalities or cuts that are used to build value function approximations for the subproblem MIPs.

Let $Q(\omega)$ denote the set of terminal nodes of the TB&B that have been generated for subproblem associated with scenario ω. For any node $q \in Q(\omega)$, let $z_{ql}(\omega)$ and $z_{qh}(\omega)$ denote vectors whose elements are used to define lower and upper bounds, respectively, on the second stage integer variables. Then the subproblem LP relaxation for node q in algorithmic iteration k can be written as follows:

$$\text{Min } q^\top y, \tag{4.4a}$$

$$\text{s.t. } W^k y \geq r^k(\omega) - T^k(\omega)x^k, \tag{4.4b}$$

$$y \geq 0, \tag{4.4c}$$

$$y \geq z_{ql}(\omega), \tag{4.4d}$$

$$-y \geq -z_{qh}(\omega). \tag{4.4e}$$

The corresponding dual LP is:

$$\text{Max } \theta_q(\omega)^\top [r^k(\omega) - T^k(\omega)x^k] + \psi_{ql}(\omega)^\top z_{ql}(\omega) - \psi_{qh}(\omega)^\top z_{qh}(\omega), \tag{4.5a}$$

$$\text{s.t. } \theta_q(\omega)W^k + \psi_{ql}(\omega)^\top - \psi_{qh}(\omega)^\top \leq g^\top, \tag{4.5b}$$

$$\theta_q(\omega) \geq 0,\ \psi_{ql}(\omega) \geq 0,\ \psi_{qh}(\omega) \geq 0, \tag{4.5c}$$

where the vectors $\psi_{ql}(\omega)$ and $\psi_{qh}(\omega)$ are appropriately dimensioned vectors.

In this approach each scenario subproblem has its own TB&B tree and the optimal dual multipliers for at each terminal node of the tree are used to obtain a lower bounding function on the subproblem MIP objective function. Without loss of generality it is assumed that the LP subproblems at the nodes are always feasible and that the nodes are fathomed when their lower bounds exceed the best upper bound obtained. The lower bounding functions at the nodes provide a disjunctive description of an approximation to the MIP value function. Such a lower bounding function may be obtained by requiring that $x \in X$ and that the following disjunction holds true:

$$\eta \geq \theta_q(\omega)^\top [r^k(\omega) - T^k(\omega)x] \ + \ \psi_{ql}(\omega)^\top z_{ql}(\omega) \ - \ \psi_{qh}(\omega)^\top z_{qh}(\omega)$$
$$\text{for at least one } q \in Q(\omega). \tag{4.6a}$$

Note that disjunction (4.6a) is valid since any optimal solution of the second-stage must be associated with at least one of the nodes $q \in Q(\omega)$.

For each $q \in Q(\omega)$ we can associate an epigraph

$$E_q^k(\omega) = \{(\eta, x) \mid \eta \geq \bar{\nu}_q^k(\omega) - \bar{\gamma}_q^k(\omega)^\top x, Ax \geq b, x \geq 0, \eta \geq 0\}$$

with

$$\bar{\nu}_q^k(\omega) = \theta_q^k(\omega)^\top r^k(\omega) + \psi_{ql}^k(\omega)^\top z_{ql}(\omega) - \psi_{qh}^k(\omega)^\top z_{qh}(\omega)$$

and

$$\bar{\gamma}_q^k(\omega) = T^k(\omega)^\top \theta_q^k(\omega)$$

The validity of (4.6a) implies that the epigraph of the subproblem (MIP) value function for scenario $\omega \in \Omega$ is a subset of the following disjunctive set:

$$\Pi_k(\omega) = \{(\eta, x) \in \cup_{q \in Q(\omega)} E_q^k(\omega).\}$$

A convexification of this set is used to derive lower bounding functions for use in the master program. This is achieved by applying the disjunctive cut principle to this disjunction. The epi-reverse polar of $\Pi_k(\omega)$, which is denoted by $\Pi_k^\dagger(\omega)$, is as follows:

$$\begin{aligned}
\Pi_k^\dagger(\omega) = \big\{ & \sigma_0(\omega) \in \Re, \sigma(\omega) \in \Re^{n_1}, \zeta(\omega) \in \Re \mid \quad \forall q \in Q(\omega), \exists\, \tau_q(\omega) \geq 0, \tau_{0q}(\omega) \in \Re_+ \\
\text{s.t.} \quad & \sigma_0(\omega) \geq \tau_{0q}(\omega) \quad \forall q \in Q(\omega) \\
& \sum_{q \in Q(\omega)} \tau_{0q}(\omega) = 1 \\
& \sigma_j(\omega) \geq \tau_q^\top(\omega) A_j + \tau_{0q}(\omega) \bar{\gamma}_{qj}^k(\omega) \quad \forall q \in Q(\omega), j = 1, ..., n_1 \\
& \zeta(\omega) \leq \tau_q(\omega)^\top b + \tau_{0q}(\omega) \bar{\nu}_q^k(\omega) \quad \forall q \in Q(\omega) \\
& \tau_q(\omega) \geq 0, \tau_{0q}(\omega) \geq 0, \quad \forall q \in Q(\omega) \big\}.
\end{aligned} \tag{4.7}$$

In order to get an optimality cut to add to the master program, an epi-reverse polar LP that characterizes the disjunction is formed and solved for each scenario

subproblem. We shall refer to this LP as the ERP-LP and is given as follows:

$$\begin{aligned} \text{Max} \quad & \zeta(\omega) - \eta^k(\omega)\sigma_0(\omega) - (x^k)^\top \sigma(\omega) \\ \text{s.t.} \quad & (\sigma_0(\omega), \sigma(\omega), \zeta(\omega)) \in \Pi_k^\dagger(\omega). \end{aligned} \tag{4.8}$$

Let $(\sigma_0^k(\omega), \sigma^k(\omega), \zeta^k(\omega))$ denote an optimal solution of the ERP-LP (4.8). Then a disjunctive cut that provides a lower bound on the subproblem MIP value function is given by

$$\sigma_0^k(\omega)\eta(\omega) + \sigma^k(\omega)^\top x \geq \zeta^k(\omega). \tag{4.9}$$

The optimality cut to be included in the master program at iteration k is therefore given by

$$\eta + E[\frac{\sigma^k(\tilde{\omega})}{\sigma_0^k(\tilde{\omega})}]^\top x \geq E[\frac{\zeta^k(\tilde{\omega})}{\sigma_0^k(\tilde{\omega})}]. \tag{4.10}$$

It is appropriate to note here that the novelty of this approach is the fact that disjunctive programming is used to approximate the value function of MIP problems. In the D^2 algorithm, disjunctive programming is used to provide tight relaxations of the set of feasible integer points. Consequently, applying this branch-and-bound approach to the D^2 method results in a branch-and-cut method for SMIP. The D^2-BAC algorithm can be summarized as follows:

Algorithm D^2-BAC

0. Initialize. Let $\epsilon > 0$ and $x^1 \in X \cap \mathcal{B}$ be given. Set $V_1 \leftarrow \infty$, $k \leftarrow 1$, $W^1 \leftarrow W$, $T^1(\omega) \leftarrow T(\omega)$, and $r^1(\omega) = r(\omega)$.

1. Apply the D^2 algorithm to solve one LP subproblem for each $\omega \in \Omega$. For each $\omega \in \Omega$, use the matrix W^k as well as the right hand side vector $r^k(\omega) - T^k(\omega)x^k$ to solve (4.2) as in the D^2 algorithm and generate and add D^2 cuts to each scenario subproblem if possible.

2. Update the approximation. For each $\omega \in \Omega$ partially "solve" one MIP subproblem using branch-and-bound and form and solve the ERP-LP

(4.8), and after processing all ω, derive an optimality cut. If $y^k(\omega)$ satisfies integral restrictions for all $\omega \in \Omega$, then $V_{k+1} \leftarrow \text{Min}\{c^\top x + E[f(x^k, \tilde{\omega})], V_k\}$. Add the optimality cut to the master program.

3. Solve the master program and check the stopping criterion. Let $x^{k+1} \in \text{argmin}\{c^\top x + \eta^k) \mid x \in X \cap \mathcal{B}\}$, and let v_k denote the optimal value of the master problem. If $V_k - v_k \leq \epsilon$, stop. Otherwise, $k \leftarrow k + 1$ and repeat from step 1.

As far as implementation is concerned, we propose to allow for partial solves only when it is beneficial to do so. Otherwise, as pointed out earlier, and as shown by the computational experiments in Chapter 7, the algorithm may get bogged down in attempts to "dive" deep into the TB&B tree even for first-stage solutions that are not anywhere near the neighborhood of an optimal solution. Thus one may activate the TB&B process by specifying the number of nodes to explore when, for example, there is no significant improvement in the algorithmic lower bound. Finally, one can view the D^2 algorithm as a special case of the D^2-BAC algorithm outlined above in that, in the D^2 algorithm we do not "dive" into the TB&B, but simply stay at the root node.

4.4 The L^2 Algorithm

Laporte and Louveaux (1993) derive an optimality cut for the piecewise linear approximation of the expected value function as follows. Let L be the lower bound on the value of the recourse function as defined earlier in Section 4.1. For a k^{th} first-stage feasible solution x^k, let $x_i = 1$, $i \in S_k$ and $x_i = 0$, $i \notin S_k$, let $f^k = E[f^k(x^k, \tilde{\omega})]$ be the corresponding expected second-stage value. Then the

optimality cut is defined as

$$\eta \geq (f^k - L)(\sum_{i \in S_k} x_i - \sum_{i \notin S_k} x_i) - (f^k - L)(|S_k| - 1) + L \tag{4.11}$$

where $|S_k|$ is the cardinality of S_k. The quantity $(\sum_{i \in S_k} x_i - \sum_{i \notin S_k} x_i)$ takes the value $|S_k|$ only when x is the k^{th} feasible solution. In this case the right hand side takes that value f^k. Otherwise, it is always less than or equal to $|S_k|$ and the right hand side takes a value less than or equal to L. Hence, it follows that the cut is sharp at x^k and that its value is at most L at all other binary solutions. Nevertheless, Laporte and Louveaux (1993) show how to improve the optimality cuts when more information is available on $E[f(x^k, \tilde{\omega})]$, such as other bounds. The L^2 algorithm can be stated as follows:

Algorithm L^2

0. Initialize. Let $\epsilon > 0$ and $x^1 \in X \cap \mathcal{B}$ be given. Set $V_1 \leftarrow \infty$, $k \leftarrow 1$; compute L.

1. Solve one MIP Subproblem for each $\omega \in \Omega$. For each $\omega \in \Omega$, use the matrix W as well as the right hand side vector $r(\omega) - T(\omega)x^k$ to solve subproblem MIP. Set
$V_{k+1} \leftarrow \text{Min}\{c^\top x^k + E[f(x^k, \tilde{\omega})], V_k\}$ and $f^k \leftarrow E[f(x^k, \tilde{\omega})]$.

2. Update and Solve the Master Problem. Using L, x^k and f^k derive an optimality cut (4.11). Append the cut to the master program (4.1). Let $x^{k+1} \in \text{argmin}\{c^\top x + \eta) \mid x \in X \cap \mathcal{B}\}$, and let v_k denote the optimal value of the master problem. If $V_k - v_k \leq \epsilon$, stop. Otherwise, $k \leftarrow k + 1$ and repeat from step 1.

Like the D^2 algorithm, no explicit branching is done this L^2 algorithm and the master and subproblem MIPs are solved exactly by the CPLEX MIP solver.

Note that in the algorithm subproblem MIPs are solved at every iteration of the algorithm. This together with the generally weak optimality cut (4.11) does have an adverse effect on the performance of this algorithm as our computational experiments show.

4.5 Computer Implementation

All the three decomposition algorithms were implemented in the C programming language and work in conjunction with the general-purpose programming system CPLEX 7.0 ILOG (2000) for solving all the LP and MIP problems. In this section we highlight some important issues to consider when implementing the D^2 algorithm and provide some pseudo-code for the main steps of the algorithm.

4.5.1 Implementation Issues

The C programming data structures were used to store and handle all the program data and took advantage of the dynamic memory allocation capabilities of C. Due to computational efficiency and memory considerations, all the problem matrix data such as A, W^k, $r(\omega)^k$ and $T(\omega)^k$ are stored in sparse matrix format arrays. The data W^k, $r(\omega)^k$ and $T(\omega)^k$ grow at every iteration of the algorithm whenever a D^2 cut is generated. Thus care must be taken in how the data is stored in order to avoid program efficiency and memory problems.

The master problem is stored as a CPLEX LP object to which optimality cuts are added at every iteration k of the algorithm. Similarly, the second-stage scenario subproblem LP and MIP are kept as CPLEX LP/MIP objects, respectively, but the right hand side $(r^k(\omega) - T^k(\omega))$ is computed and reset for each scenario $\omega \in \Omega$ before optimization. The D^2 cut coefficients π^k are appended to the subproblem LP/MIP object as well as the matrix array W^{k-1}, stored separately. This matrix is needed

for forming the C^3-SLP (Problem 3.10, Ch. 3). The right hand side coefficients at iteration k are added to the arrays $r^{k-1}(\omega)$ and $T^{k-1}(\omega)$ for each scenario $\omega \in \Omega$, respectively.

We create a CPLEX LP object for the C^3-SLP (Problem 3.10, Ch. 3) only when needed and free it after optimizing the SLP and getting the optimal solution. To select a disjunction variable our implementation scans the scenarios in order and chooses a variable whose solution is the most fractional for the first scenario with a fractional solution. That is, for algorithmic iteration k and the first scenario ω with a fractional solution $y(\omega)$, the disjunction variable index $j(k)$ is determined by $\{j(k) = j : \text{argmin}_j\{|y_j(\omega) - 0.5|\}\}$.

In order to guarantee algorithmic convergence, the C^3-SLP (Problem 3.10, Ch. 3) must be formed with the constraint matrix composed of the original constraint matrix W and all the π's that were generated by using those disjunction variables whose indices are smaller than the index for the disjunction variable chosen in this iteration (see Sen and Higle (2000) for the proof of convergence). All the other π's are excluded from the C^3-SLP (Problem 3.10, Ch. 3) constraint matrix. Thus we keep track of the disjunction variable at which each cut is generated by using an array to store the disjunction variable index for each π^k.

In forming the C^3-SLP (Problem 3.10, Ch. 3) objective function coefficients when the second-stage LP solutions do not satisfy integer requirements for at least one scenario, Sen and Higle (2000) propose to use the expected value of the solutions. However, to guarantee that the D^2 cut generated actually cuts off the current fractional solution, the conditional expectation with respect to the scenarios with a fractional solution for the selected disjunction variable should be used. This is critical to the algorithm since the D^2 cut generated and added to the scenario subproblem influences the choice of the first-stage solution via the optimality cut (4.1c).

Similar to the creation of the CPLEX LP object for the C^3-SLP (Problem 3.10, Ch. 3), the CPLEX LP object for the RHS-LP (Problem 3.11, Ch. 3) is dynamically created for each scenario, optimized, and then freed at each iteration of the algorithm. Thus we avoid unnecessary use of computer memory to store potentially several CPLEX LP objects. All the appropriate solution data from each CPLEX LP are extracted before freeing the object. Next we provide some pseudo-code for some selected steps in the D^2 algorithm.

4.5.2 Illustrative Pseudo-Code

We highlight the major steps of the algorithm and decompose and store various pieces of data in a manner that is commensurate with the necessary computations. To differentiate between mathematical and programming constructs, we adopt the use of a `typewriter` font when referring to the later. We also use the **bold** font for function names.

The first step of the algorithm after initialization is to optimize all the scenario subproblem LPs. In this step it is necessary to compute and set the right hand side $r^k(\omega) - T^k(\omega)x^k$ for each scenario subproblem CPLEX LP object. After optimization of each scenario subproblem we need to store the optimal primal and dual solutions for use in the formation of the C^3-SLP (Problem 3.10, Ch. 3) and potentially, in computing the Benders-type optimality cut (4.1c). For each algorithmic iteration k let us define the sparse matrix format arrays as follows:

$\texttt{mat\_r}[\omega] = r^k(\omega)$

$\texttt{mat\_T}[\omega] = T^k(\omega)$

$\texttt{rhs}[\omega] = r^k(\omega) - T^k(\omega)x^k.$

Let `solnY` denote the sparse matrix format array for storing the optimal primal solutions $\{y^k(\omega)\}_{\omega=1}^{S}$ and let `duals` denote the array for the scenario subproblem

LP dual solution $\{\psi^k(\omega)\}$, respectively. Let the pointer to the subproblem CPLEX LP object be given by `sub_lp`. Let us also index the scenarios by `i` and denote the total number of scenarios by $S = |\Omega|$. In all that follows (`matrix` $\times$ `vector`) and (`vector` $\times$ `vector`) multiplications are carried out in the usual manner. The array `x` stores the first-stage solution x. The pseudo-code for a procedure for updating and solving scenario subproblem LPs can be stated as follows:

```
procedure solvesubprobLPs(sub_lp, x, mat_r, mat_T, solnY, duals)
{
   exp_obj = 0.0;
   for (i = 0; i < S; i++){
      rhs[i] = mat_r[i] - mat_T[i]×x;
      setSubprobLP_rhs(sub_lp, rhs[i]);
      optimizeLP(sub_lp);
      exp_obj += getObj(sub_lp);
      getprimalSoln(sub_lp, solnY[i]);
      getdualSoln(sub_lp, duals);
      computeUpdateBendersCut(duals);
   }
}
```

The procedure **solvesubprobLPs** solves the LP relaxation of the subproblem for each scenario by first resetting the right-hand side of the LP according to

the scenario right hand side data and then calling an LP solver to optimize the problem. Note that for scenario subproblems with randomness in the objective function, the random objective coefficients have to be reset for each scenario within the *for* loop. The procedure for updating and solving the scenario subproblem MIPs **solvesubprobMIPs** is the same **solvesubprobLPs** except that we impose integer restrictions on the scenario subproblem relaxation.

The next step of the algorithm is to check whether the solution satisfies integer requirements. If it does then we simply form an optimality cut to add to the master program. Otherwise, we need to select a disjunction variable and form the C^3-SLP (Problem 3.10, Ch. 3) to get the left-hand side coefficients of the D^2 cut and the multipliers for forming the RHS-LP (Problem 3.11, Ch. 3) for each scenario.

To form the objective function of the C^3-SLP (Problem 3.10, Ch. 3) it is imperative that the computation of the objective coefficients be conditioned on the subproblem scenarios with a fractional solution for the disjunction variable component. The objective function of the C^3-SLP (Problem 3.10, Ch. 3) is given by Max $E[\pi_0(\tilde{\omega})] - E[y^k(\tilde{\omega})]\pi$ (see Sen and Higle (2000)). Let us denote the arrays for storing the optimal solutions of the variables π_0 by `pi_0` and the variables π by `pi`. Also let the corresponding objective coefficient arrays be denoted by `pi_0_coefs` and `pi_coefs`, respectively, and let the scenario probabilities be stored in the array `scen_probs`. Let `d` $= j(k)$ be the disjunction variable index in iteration k and let `EPS` be the zero tolerance for the integer requirement on the binary solution. A procedure for computing the conditional C^3-SLP (Problem 3.10, Ch. 3) objective coefficients can be given as follows:

```
procedure computeC3LPobjCoefs(solnY, pi_0_coefs, pi_coefs,
scen_probs, d)
```

```
{
  sum_prob = 0.0;
  for (i = 0; i < S; i++){
    if(solnY[i][d] > EPS && solnY[i][d] < 1-EPS)
      sum_prob += scen_probs[i];
  }
  for (i = 0; i < S; i++){
    if(solnY[i][d] > EPS && solnY[i][d] < 1-EPS)
      pi_0_coefs[i] = scen_probs[i]/sum_prob;
    else
      pi_0_coefs[i] = 0.0;
  }
  for (j = 0; j < n2; j++)
    pi_coefs[j]= 0.0;
  for (i = 0; i < S; i++){
    if(solnY[i][d] > EPS && solnY[i][d] < 1-EPS){
      for (j = 0; j < n2; j++)
        pi_coefs[j] += pi_0_coefs[i]*solnY[i][j];
    }
  }
```

}

Now let us consider the procedure for forming and solving the *RHS*-LP (Problem 3.11, Ch. 3). Let the optimal solution to the C^3-SLP (Problem 3.10, Ch. 3) be stored in the arrays `pi` $= \pi^k$, `lambda_01` $= \lambda_{01}^k$, `lambda_11` $= \lambda_{11}^k$, and the coefficients `lambda_02` $= \lambda_{02}^k$ and `lambda_12` $= \lambda_{12}^k$. Note that the array `pi` stores the common-cut coefficients while the arrays `lambda_01` and `lambda_11` store the multipliers for the computation of $\bar{\nu}_0^k(\omega)$, $\bar{\nu}_1^k(\omega)$, $\bar{\gamma}_0^k(\omega)$ and $\bar{\gamma}_1^k(\omega)$ to be used in forming the *RHS*-LP (Problem 3.11, Ch. 3) for each scenario $\omega \in \Omega$. Let us denote the arrays to store these multipliers by `nuBar_0[`ω`]`, `nuBar_1[`ω`]`, `gammaBar_0[`ω`]` and `gammaBar_1[`ω`]`, respectively. A procedure for updating the subproblem constraint matrix W^k (denoted `mat_W`) can be named:

```
procedure append(mat_W, pi)
```

Let the floor and ceiling on the disjunction variable `d` $= j(k)$ in iteration k be denoted by `yBar_floor` $= \lfloor \bar{y}_{j(k)} \rfloor$ and `yBar_ceil` $= \lceil \bar{y}_{j(k)} \rceil$, respectively. Let the arrays for storing an optimal solution to the *RHS*-LP (Problem 3.11, Ch. 3) be denoted by `sigma_0` $= \sigma_0^k(\omega)$, `sigma` $= \sigma^k(\omega)$ and `delta` $= \delta^k(\omega)$, respectively. Then a procedure for forming and solving the *RHS*-LP (Problem 3.11, Ch. 3) for scenario `i` can be given by:

```
procedure formAndSolveRHSLP(rhs_lp, x, lambda_01[i], lambda_11[i],
  lambda_02[i], lambda_12[i], gammaBar_0[i], gammaBar_1[i],
  yBar_floor, yBar_ceil, d)
  {
```

```
    nuBar_0[i] = lambda01^T[i]×rhs[i] - lambda02^T[i]×yBar_floor;
    nuBar_1[i] = lambda11^T[i]×rhs[i] - lambda12^T[i]×yBar_ceil;
    gammaBar_0[i] = lambda01^T[i]×mat_T[i];
    gammaBar_1[i] = lambda11^T[i]×mat_T[i];
    formRHSLP_rhs(rhs_lp, x, nuBar_0[i],nuBar_1[i], gammaBar_0[i],
    gammaBar_1[i]);
    optimizeLP(rhs_lp);
    getprimalSoln(rhs_lp, sigma_0[i]);
    getprimalSoln(rhs_lp, sigma[i]);
    getprimalSoln(rhs_lp, delta[i]);
  }
```

Next, we need a procedure for updating the scenario subproblem right-hand side. Such a procedure given an optimal solution to the *RHS*-LP (Problem 3.11, Ch. 3) for scenario i can be stated as follows:

```
procedure updateRhs( sigma_0[i], sigma[i], delta[i])
{
  nu[i] = delta[i]/sigma_0[i];
  for (j = 0; j < n1; j++)
    gamma[i][j] = sigma[i][j]/sigma_0[i];
```

```
    append(rhs[i], nu[i]);
    append(mat_T[i], gamma[i]);
  }
```

The remaining important part of the algorithm is the computation of the aggregated Benders-type optimality cut given in equation (4.1c). Let the coefficients associated with the variable x of the cut be stored in the array `beta` and the right hand side coefficient be stored in `alpha`. A procedure for generating the optimality cut can be given as follows:

```
procedure computebenderscutCoefs(duals_0, mat_r, mat_T, beta, alpha)
{
    alpha = 0.0;
    for (j = 0; j < n1; j++)
      beta[j] = 0.0;
    for (i = 0; i < S; i++) {
      alpha += scen_prob[i]*duals[i]×mat_r[i];
      for (j = 0; j < n1; j++)
        beta[j] += scen_prob[i]*duals[i]×mat_T[i];
    }
}
```

We now turn to the D^2-BAC algorithm and highlight the branch-and-bound part of the algorithm as required in Step 2 of the algorithm. For each $\omega \in \Omega$ we need to partially solve one subproblem MIP using a TB&B procedure. The optimal dual solutions from the terminal nodes of the TB&B tree are used in forming *ERP*-LP (4.8). The optimal solution of *ERP*-LP for all outcomes $\omega \in \Omega$ are then used to derive an optimality cut to add to the master program. Let us define the following program variables:

`max_numnodes`: maximum number of nodes to explore in the TB&B tree.

`fathomed_numnodes`: number of fathomed nodes in the TB&B tree.

`branch_index`: branching variable index.

`best_bound`: current best bound for the TB&B tree.

`var_floor`: floor of the branching fractional solution.

`var_ceil`: ceiling of the branching fractional solution.

`bblist_ptr`: pointer to list of subproblem LPs that have not been explored.

`bbdata_ptr`: pointer to a data structure for storing optimality cut data.

In the implementation we keep the list of node subproblem LPs that have not been explored as a linked list and `bblist_ptr` is the pointer to this list. A procedure for performing the TB&B step for a given scenario `s` can be written as follows (the names of the functions describe the operations to perform):

```
Procedure TBB(bblist_ptr, bbdata_ptr, subdata_ptr, stochdata_ptr,
    max_numnodes, s)
    fathomed_numnodes = 0;
    best_bound = ∞;
```

```
initialize(bblist_ptr, bbdata_ptr, subdata_ptr, s);
do
{
   removeFrontNode(bblist_ptr, front_node);
   setSubprobBranchConstr(sub_lp, front_node);
   setSubprobRhs(sub_lp, stochdata_ptr, s);
   CPXsolve(sub_lp);
   CPXgetsolution(sub_lp, sub_obj, soln_y[s], dual[s])
   fractional = isFractionalSolution(soln_y[s], branch_index);
   if(fractional == false)
   {
      updateBestBound(bbdata_ptr sub_obj, best_bound);
      fathomThisNode(bbdata_ptr, dual[s]);
      fathomed_numnodes++;
   }
   else
   {
      if(sub_obj ≥ best_bound)
      {
         fathomThisNode(bbdata_ptr, dual[s]);
```

```
            fathomed_numnodes++;
        }
        else
        {
            var_floor = getfloor(soln_y[s], branch_index);
            var_ceil = getceil(soln_y[s], branch_index);
            newnode = createNode(front_node, branch_index,
            var_floor);
            addNode(bblist_ptr, newnode);
            newnode = createNode(front_node, branch_index,
            var_ceil);
            addNode(bblist_ptr, newnode);
        } endif
    } endif
    num_nodes = fathomed_numnodes + bblist.size;
} while(bblist_ptr.size > 0 && num_nodes < max_numnodes);
```

4.6 Summary

This chapter has presented details of a computer implementation of the D^2 algorithm. The issues associated with the implementation are discussed and critical parts of the algorithms illustrated using pseudo-code fragments. Due to the fact

parts of the data in the algorithm grow with algorithmic iterations, we took advantage of the C programming data structures to store and handle all the program data in sparse matrix format and to use the dynamic memory allocation capabilities of C. Computational studies of the implemented algorithm will be reported in the following chapters.

CHAPTER 5

COMPUTATIONAL RESULTS FOR STOCHASTIC COMBINATORIAL OPTIMIZATION PROBLEMS

Combinatorial optimization problems have applications in a variety of sciences and engineering. In the presence of data uncertainty, these problems lead to stochastic combinatorial optimization problems which result in very large scale combinatorial optimization problems. In this chapter, we report on the solution of some of the largest stochastic combinatorial optimization problems consisting of over a million binary variables. While the methodology is quite general, the specific application with which we conduct our experiments arises in stochastic server location problems. The main observation is that stochastic combinatorial optimization problems are comprised of loosely coupled subsystems. By taking advantage of the loosely coupled structure, we show that decomposition-coordination methods provide highly effective algorithms, and surpass the scalability of even the most efficiently implemented backtracking search algorithms.

One example of a SCO problem is the network design problem that determines which nodes and arcs should be built so as to provide network services at least "cost." Despite several decades of advances in CS/OR, combinatorial optimization problems continue to evoke parallels with galactic dimensions. For instance, a combinatorial optimization problem (e.g. network design) with 1000 variables can lead to a search space with 2^{1000} decision states! This is sometimes referred to as "combinatorial explosion." Moreover, these notorious problems belong to the class of NP-hard problems for which "good" algorithms are unlikely. It is no surprise

that combinatorial optimization remains a "Grand CS/OR Challenge" problem.

It turns out that the challenge of combinatorial optimization is magnified many-fold in cases where data is uncertain. For instance, a network design problem in which customer locations are unknown leads to a stochastic combinatorial optimization (SCO) problem. Even the smallest of practical SCO problems lead to an astronomical number of decision states, and easily exceed current computational capabilities. As one might expect, SCO problems provide an even grander challenge. In this chapter, we report on the solution of instances with the largest data sets for SCO problems to date. These instances, one of which contains over a million binary (0-1) variables, represent data for stochastic server location problems (SSLP) in which servers have to be located prior to demand realization.

These problems (SSLP) are said to have had a significant role in the economic downturn associated with the telecommunications sector of the U.S. economy. Unfortunately, the lack of algorithms for solving SSLP has prompted industry to use planning models that are based on one (deterministic) forecast. The telecommunications sector is now awash in unused server capacity that has resulted from a combination of inaccurate forecasts, and an over-reliance on deterministic planning. In a volatile economy, planning models should recognize that forecasts can be error prone, and should seek plans that are robust to forecasting errors. One approach that provides robust plans is known as stochastic programming (SP) (Birge and Louveaux, 1997) and incorporates multiple scenarios within a planning model. Over a decade ago, Sen et al. (1994) used the SP planning methodology to an industrial-sized network planning problem for Sonet-Switched Networks (SSN), and demonstrated improved network performance due to the SP model. Subsequently, others have reported solving the SSN instance using advanced computing architectures such as grid-computing (Linderoth and Wright, 2003). However, the types of algorithms used to solve SSN are based on stochastic linear programming, and do not allow binary variables. Networks that are being

implemented today consist of high speed optical fiber cables, and high speed optical switches which are very capital intensive. In addition, service providers like AT&T are unsure of customer demand (Doverspike, 2003). Network design models under these circumstances lead to SSLP, which is the class of models studied here. Unfortunately, network designers have not had access to algorithms that can solve SSLP of realistic dimensions.

This chapter begins with a statement of the stochastic server location problem (SSLP) as a stochastic combinatorial optimization model. We then proceed with a brief summary of previous computational experience with SCO models. Following this, we present our experimental setup and our computational results with SSLP. These experiments demonstrate that new decomposition methods (e.g. D^2 algorithm) provide a powerful tool for stochastic combinatorial optimization.

5.1 Stochastic Server Location Problems

The stochastic server location problems (SSLP) we study find applications in a variety of domains such as network design for electric power, internet services, telecommunications, and water distribution. In particular, consider a set of possible customer buildings in a metropolitan area for which a service provider is interested in installing optical fibers and switching equipment in the most profitable manner. Due to the uncertainty regarding the customer base for high speed services, telecommunications providers often adopt a very conservative approach to capital investment, leading to potential losses in revenue (Doverspike, 2003). Such problems are common in practice and can be formulated as the SSLP. Because of the variety of application domains for SSLP Wang et al. (2003), we use the names *server* and *client* in a generic sense.

Figure 5.1 gives an illustration of the stochasticity in the SSLP. The figure shows two scenarios, *1* and *2*. For *scenario 1* we have 4 potential server locations and 12

potential clients (see panel a) of Figure 5.1). In panel b), which depicts the clients that materialize in *scenario 1*, only 8 out of the 12 potential clients are available for service. Panel c) of this figure shows the optimal server locations if we only plan for *scenario 1*. For *scenario 2* we again have 4 potential server locations and 12 potential clients. However in this scenario, only 6 out of the 12 potential clients are available for service (see panel b)). In this case the optimal server locations are at the two sites shown in panel c). Although Figure 5.1 shows the optimal server locations and client-server assignments for each scenario, the goal of the SSLP is to find the overall optimal server locations (see panel d)) which accommodate all foreseeable scenarios. In this case the overall optimal solution is not optimal for either of the two scenarios. In the next subsection we formally give two model formulations for the SSLP.

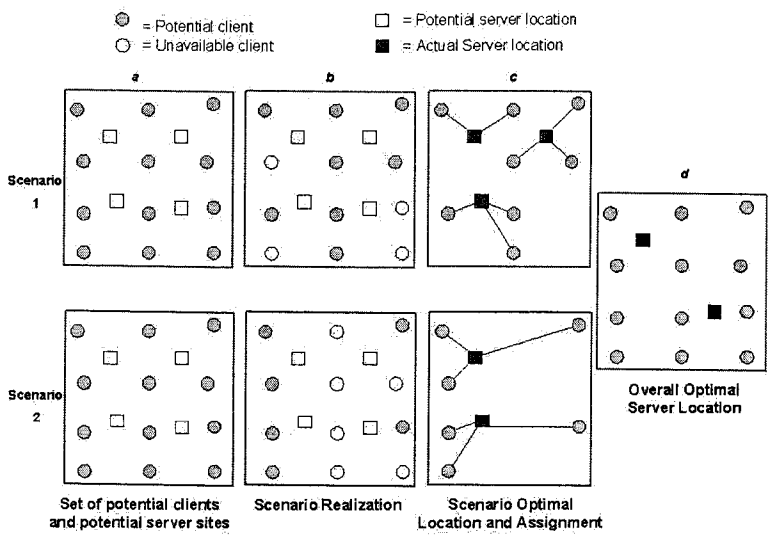

Figure 5.1: SSLP Illustration

5.2 Model Formulation

Let $\mathcal{I}$ and $\mathcal{J}$ be index sets for the clients and servers, with $|\mathcal{I}| = n$ and $|\mathcal{J}| = m$. Let $\mathcal{Z}$ denote a given set of zones. For $i \in \mathcal{I}$ and $j \in \mathcal{J}$ we define the following:

Data:

c_j: Cost of locating a server at location j

q_{ij}: Revenue from client i being served by server at location j

d_{ij}: Client i resource demand from server at location j

u: Server capacity

v: An upper bound on the total number of servers that can be located

w_z: Minimum number of servers to be located in zone $z \in \mathcal{Z}$

$\mathcal{J}_z$: The subset of server locations that belong to zone z

$$h^i(\omega) = \begin{cases} 1, & \text{if client } i \text{ is present in scenario } \omega, \omega \in \Omega, \\ 0, & \text{otherwise.} \end{cases}$$

$h(\omega)$: Client availability vector for scenario $\omega \in \Omega$ with elements $h^i(\omega), i \in \mathcal{I}$

p_ω: Probability of occurrence for scenario $\omega \in \Omega$

Decision variables:

$$x_j = \begin{cases} 1, & \text{if a server is located at site } j, \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{ij}^{\omega} = \begin{cases} 1, & \text{if client } i \text{ is served by a server at location } j \text{ under scenario } \omega, \\ 0, & \text{otherwise.} \end{cases}$$

The essence of the SSLP may be described as follows. Suppose that we place a server at location j. Then, this allocation costs c_j, and provides enough capacity to serve up to u amount of resource to clients. The revenue generated by serving client i from location j, is denoted q_{ij}. There is also a shortage cost (penalty) q_{j0} for each unit of demand that remains unserved among the clients assigned to server j. If client i is served by a server at location j, it uses d_{ij} units of resource from the server. Note that the dependence of resource utilization (d_{ij}) on the client-server pair allows us to model losses occurring from assigning client i to server j. Such considerations are important in certain networks like electricity and water. In cases where the losses are negligible (at least for planning purposes), one could use the same value of d_{ij} for all j, as long as i is fixed.

As far as operational considerations are concerned, we allow only one server to be installed at each location and each client can only be served by one server. There is also a requirement that a minimum number of servers denoted w_z, $z \in \mathcal{Z}$ be located in a given area or zone. Finally, the problem is to choose locations of servers and client-server assignments that maximize the total expected revenue subject to the given constraints. The SSLP can be stated as follows:

$$\text{Min} \quad \sum_{j\in\mathcal{J}} c_j x_j - \sum_{\omega\in\Omega} p_\omega \left(\sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{J}} q_{ij}^\omega y_{ij}^\omega - \sum_{j\in\mathcal{J}} q_{j0}^\omega y_{j0}^\omega \right) \tag{5.1a}$$

$$s.t. \quad \sum_{j\in\mathcal{J}} x_j \leq v, \tag{5.1b}$$

$$\sum_{j\in\mathcal{J}_z} x_j \geq w_z, \ \forall z \in \mathcal{Z} \tag{5.1c}$$

$$\sum_{i\in\mathcal{I}} d_{ij} y_{ij}^\omega - y_{j0}^\omega \leq u x_j, \ \forall j \in \mathcal{J}, \ \omega \in \Omega, \tag{5.1d}$$

$$\sum_{j\in\mathcal{J}} y_{ij}^\omega = h^i(\omega), \ \forall i \in \mathcal{I}, \ \omega \in \Omega, \tag{5.1e}$$

$$x_j \in \{0,1\}, \ \forall j \in \mathcal{J}, \tag{5.1f}$$

$$y_{ij}^\omega \in \{0,1\}, \ \forall i \in \mathcal{I}, j \in \mathcal{J}, \omega \in \Omega. \tag{5.1g}$$

$$y_{j0}^\omega \geq 0, \ \forall j \in \mathcal{J}, \ \omega \in \Omega. \tag{5.1h}$$

Formulation (5.1a-5.1h) is the so called deterministic equivalent problem (DEP) in Stochastic Programming. When one solves this problem, one obtains a recommendation to locate servers in locations that will hedge against a variety of scenarios in which certain clients do not materialize. The variables y_{ij}^ω are decisions that will be implemented in the future, when scenario ω is finally observed. The location variables (x) are referred to as first-stage decisions, and the assignment variables (y^ω) are referred to as recourse (or second-stage) decisions. Unlike the first-stage variables, the latter are dependent on the scenario ω.

The constraints provide a mechanism to impose the operational requirements. Thus constraints (5.1b) satisfy the requirement that only up to a total of v available servers can be installed. The zonal requirements that specify how many servers are necessary in each zone are given by constraint (5.1c). Constraints (5.1d) dictate that a server located at site j can serve only up to its capacity u. We have introduced a variable y_{j0}^ω in this constraint to accommodate any overflows that are not served due to limitations in server capacity. These overflows result in a loss of revenue at a rate

of q^{ω}_{j0}. Unlike the deterministic version of such problems, the inclusion of an artificial variable may allow a client to be assigned to servers that are not located. However, penalty costs associated with such an assignment may result in such high costs as to preclude it in an optimal solution, unless server capacity is so limited that some clients have to be turned away. For cases in which the server capacities are severely restricted, linear overflow costs may not provide an appropriate modeling tool and an extension of this model may be necessary. Since most of our experiments will be conducted on instances with sufficient server capacity, overflows will be zero, and linear overflow costs will suffice.

Continuing with a description of the rest of the model, the requirement that each available client is served by only one server is given by constraints (5.1e). Constraints (5.1f) and (5.1g) are the binary restrictions on the decision variables. Finally, constraints (5.1h) are the nonnegativity requirements on the overflow variables.

If we denote the number of scenarios by S, where $S = |\Omega|$, and the number of zones by $|\mathcal{Z}|$, then this DEP formulation has $m(1 + nS)$ variables and $m + |\mathcal{Z}| + (m + n)S$ constraints. The number of scenarios can be very large in general and therefore, this formulation is a large scale problem and can get out of hand very quickly. Hence, the need to decompose it. For example, for a problem instance with 10 potential servers, 50 potential clients and 2000 scenarios, with no zonal constraints, we have $1,000,010$ binary variables and $120,010$ constraints!

The above formulation can be considered as a basic model from which other SSLP models can be extended. For instance, we can easily make an extension to a SSLP with multiple server types each with a different capacity. Thus we can simply define new variables for each server type and add new constraints on the server type requirements to the model. In any event, the additional constraints would depend on the specific application at hand.

We will now decompose the DEP into a two stage SMIP with complete recourse and binary variables in both stages. The two stage SSLP can be formally stated as follows:

$$\text{Min} \sum_{j\in\mathcal{J}} c_j x_j - E[f(x,\tilde{\omega})] \tag{5.2a}$$

$$s.t. \sum_{j\in\mathcal{J}} x_j \leq v, \tag{5.2b}$$

$$\sum_{j\in\mathcal{J}_z} x_j \geq w_z, \ \forall z \in \mathcal{Z} \tag{5.2c}$$

$$x_j \in \{0,1\}, \ \forall j \in \mathcal{J}, \tag{5.2d}$$

where $E[.]$ is the usual mathematical expectation with

$$E[f(x,\tilde{\omega})] = \sum_{\omega\in\Omega} p_\omega f(x,\omega),$$

and for any x satisfying the constraints (5.2b - 5.2d) and $\omega \in \Omega$ we define

$$f(x,\omega) = \text{Min} \ -\sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} q_{ij} y_{ij} + \sum_{j\in\mathcal{J}} q_{j0} y_{j0} \tag{5.3a}$$

$$\text{s.t.} \sum_{i\in\mathcal{I}} d_{ij} y_{ij} - y_{j0} \leq u x_j, \ \forall j \in \mathcal{J}, \tag{5.3b}$$

$$\sum_{j\in\mathcal{J}} y_{ij} = h^i(\omega), \ \forall i \in \mathcal{I}, \tag{5.3c}$$

$$y_{ij} \in \{0,1\}, \ \forall i \in \mathcal{I}, j \in \mathcal{J}, \tag{5.3d}$$

$$y_{j0} \geq 0, \ \forall j \in \mathcal{J}. \tag{5.3e}$$

As before, the x_j's are the first-stage decision variables. We observe that although the second-stage (recourse) variables y_{ij}'s continue to depend on the outcome ω, this dependence is not explicitly indicated here. This is because the

subproblem for each outcome ω is decoupled from all other outcomes once a vector x is given. This formulation emphasizes the loosely coupled nature of SCO problems, and while this decomposition framework has been extensively used for stochastic linear programming (Cook et al., 1998), its use for SCO problems has been limited. Readers familiar with the current state of computations with SCO should feel free to proceed to section 5.4.

5.3 Previously Solved SCO Instances

Before presenting our computational results, it is appropriate to briefly examine previous computational experiments with SCO problems. This section also serves to illustrate the variety of applications in which uncertainty must be accommodated within combinatorial optimization. Table 5.1 gives a summary of some of the largest SCO instances that have been reported. The headings are as follows: "Scens" is the number of scenarios in the DEP instance, "Vars" is the number of total decision variables, "Bins" is the number of binary decision variables, "Ints" is the number of general integer decision variables, and "Constrs" is the number of constraints in the problem instance. The first three instances are available on the SIP test problem library (http://www.isye.gatech.edu/~sahmed/siplib/). The sizes of the instances shown are for the corresponding DEP formulation.

Problem dcap243_500 is a two-stage stochastic integer program arising in dynamic capacity acquisition and allocation under uncertainty. The problem has complete recourse, mixed-integer first-stage variables, pure binary second-stage variables, and discrete distributions. The formulation and computational results for this class of problems are reported in (Ahmed and Garcia, 2003) and Ahmed et al. (2004), respectively.

SEMI4 is a two-stage multi-period stochastic integer problem that arises in

Table 5.1: Summary of Previously Reported SCO Problems (DEP)

Name	Scens	Vars	Bins	Ints	Constrs
dcap243_500	500	18,018	18,006		9,012
SEMI4	4	39,820		612	23,370
SIZES10	10	825	110		341
SSCh_c5	23	3,768	114		3,933
SGAP_28	45	2,745	2,745		2,835
SVRP_100	*a*	10,000	10,000		*b*
E160-2_FRP	15	16,753	16,753		32,455

a: The recourse function has a closed-form expression
b: IP formulation has exponentially many constraints

dcap243_500 - Ahmed and Garcia (2003) and Ahmed et al. (2004)
SEMI4 - Barahona et al. (2001)
SIZES10 - Jorjani et al. (1995)
SSCh_c5 - Alonso-Ayuso et al. (2003)
SGAP_28 - Albareda-Sambola et al. (2002)
SVRP_100 - Laporte et al. (2002)
E160-2_FRP - Alonso et al. (2000)

planning semiconductor tool purchases. This model, which has mixed-integer first-stage variables and continuous second-stage variables, was solved by researchers at IBM (Barahona et al., 2001). Problem SIZES10 is an instance of a two-stage multi-period stochastic mixed-integer program arising in the product substitution applications. The problem formulation and data are given in Jorjani et al. (1995).

Problem SSCh_c5 is an instance of a strategic supply chain planning problem under uncertainty with continuous and binary variables. This problem appears in (Alonso-Ayuso et al., 2003) as problem $c5$. SGAP_28 is an instance of a stochastic generalized assignment problem and is reported in Albareda-Sambola et al. (2002) as problem instance number 28. The reformulation of this problem has both continuous and binary variables. Problem SVRP_100 is a stochastic vehicle routing problem (SVRP) in which the first-stage decisions chart a route for each vehicle and the second-stage calculates an expected penalty cost for not completing a route in case of excess demand Laporte et al. (2002). The last instance in the table is a full recourse policy model for the air traffic flow management problem (TFMP) under uncertainty in airport arrival and departure and airspace due to weather conditions Alonso et al. (2000).

An examination of the data in Table 5.1 reveals that the total number of integer variables in these SCO problems is not more that 20,000 for the DEP formulation. One may consider SCO instances of this size as representing the current state of the art.

5.4 Computational Testing

In this section, we report our computational experience in using the D^2 algorithm to solve instances whose DEP formulations are an order-of-magnitude larger than those discussed in the previous section. We compare our computational results with those obtained by the ILOG CPLEX 7.0 programming system (ILOG, 2000). It is

widely recognized that the latter is among the more efficient commercial systems that implement B&B (backtracking search).

5.4.1 Problem Instance Generation

A number of instances of problem (5.2-5.3) were generated randomly as follows. Problem data were randomly generated from the uniform distribution while scenario data were generated from the Bernoulli distribution. The server location costs were generated randomly from the uniform distribution in the interval [40, 80] and the client demands were generated in the interval [0, 25]. The client-server revenue were set at one unit per unit of client demand. The overflow costs q_{j0}, for all $j \in \mathcal{J}$, were fixed at 1000, which was a high enough penalty cost to warranty no overflows in the optimal solution.

The scenario data were generated as follows. The availability of a potential client in each scenario was generated from the Bernoulli distribution with $p = 0.5$, with a 1 indicating the presence (availability) of the client and a 0 indicating the absence (unavailability) of the client. For each problem instance the different sets of scenarios were generated using different random seeds to allow for independent scenarios. Each scenario was given an equal probability of occurrence and contained the outcomes for all the clients in the problem instance. All scenarios were checked to make sure that there were no duplicate scenarios in a given problem instance.

The degree of difficulty of an instance can be controlled by the ratio (r) of the total server capacity to the maximum possible total demand. This ratio is defined by $r = v.u/\sum_{i\in\mathcal{I}} \mathrm{Max}_{j\in\mathcal{J}}\{d_{ij}\}$, where the numerator is the total server capacity and the denominator is the total maximum demand. It reflects how much total server capacity is available to satisfy possible maximum overall client demand. A value of $r \geq 1.0$ means that the servers can satisfy the total client demand while a value of $r < 1.0$ implies that server capacity may be insufficient to satisfy client

demand. Instances in which the server capacity is highly limited, piecewise linear overflow costs may be more appropriate.

As a mnemonic, the instances were named SSLP_m_n, where m = number servers and n = number of clients. The number of servers ranged through $m = 5, 10,$ and 15 while the number of clients were set at $n = 25, 30, 45,$ and 50. The number of scenarios considered range from $S = 5$ to $S = 2000$. In particular, we report results on the problem instances SSLP_5_25, SSLP_5_50, SSLP_10_50, and SSLP_15_45, and briefly mention about the other instances.

5.4.2 Computational Results

The D^2 algorithm was implemented in C, with all small models (LP and MIP) solved by using the ILOG CPLEX 7.0 (ILOG, 2000) callable library. As a benchmark we applied the CPLEX MIP solver to the large scale DEP formulation (5.1) for each of the two-stage problem instances with the CPLEX parameters set at the following values: "set mip emphasis 1" (emphasizes looking for feasible solutions), "set mip strategy start 4" (uses barrier at the root), and "branching priority order on x" (first branches on any fractional component of x before branching on y). Bob Bixby (ILOG CPLEX) participated in the choice of these parameter settings. A CPU time limit of $10,000$ secs was imposed and any problem instance run taking more than this time limit was considered terminated. All the problems that took less than this time limit converged to an optimal solution and the percentage gap between the lower bound and the upper bound was equal to 0%. In all our computational experiments the 0-1 master programs were solved to optimality at each iteration.

All the experiments were run on a Sun 280R with 2 UltraSPARC-III+ CPUs running at 900 MHz. The results for the first set of experiments are summarized in Tables 5.2 through 5.5. The numbers of variables and constraints shown are that

of the DEP formulation. While most of the column headings in the tables are self-explanatory, we should clarify the term % Z_{IP} Gap. Entries in this column indicate the percentage difference between the optimal value of the SCO instance, and its continuous relaxation (which can be solved using stochastic linear programming). The number of algorithmic iterations are given in the column "Iters". Another observation from the tables is that the number of D^2 cuts added to the second-stage SMIP is less than the number of algorithmic iterations. This is because D^2 cuts are not generated in those iterations where all second-stage subproblem relaxations yield binary solutions. Finally, each CPU time (secs) shown in the tables records an average of three runs for the problem instance.

Table 5.2: Computational Results for Problem Instance SSLP_5_25

				D^2	D^2	D^2	CPLEX
Scens	Bins	Constrs	% Z_{IP} Gap	Iters	Cuts	CPU	CPU
5	630	151	33.11	16	1	0.13	0.12
10	1,255	301	21.25	17	5	0.22	0.46
25	3,130	751	23.47	17	10	0.42	1.82
50	6,255	1,501	24.03	17	6	0.53	4.58
100	12,505	3,001	24.93	17	10	1.03	14.69

The experimental results for problems SSLP_5_25 and SSLP_5_50 are given in Tables 5.2 and 5.3, respectively. Note that for all problem instances, the gap between the SLP objective of the DEP and the SMIP objective is over 20%. Hence the stochastic linear programming (continuous) relaxation of these SSLP instances does not provide very good approximations, and combinatorial optimization becomes necessary. The D^2 algorithm performs better than CPLEX for all the problem instances except the smallest instance of the second problem. We expect CPLEX to perform better on smaller problem instances since there is some overhead in decomposing small sized problems.

Table 5.4 shows the results for the problem SSLP_10_50, which is much larger and takes substantially much more time to solve. As expected CPLEX has the

Table 5.3: Computational Results for Problem Instance SSLP_5_50

				D^2	D^2	D^2	CPLEX
Scens	Bins	Constrs	% Z_{IP} Gap	Iters	Cuts	CPU	CPU
5	1,255	276	21.75	28	6	0.52	0.30
10	2,505	551	22.37	26	4	0.50	0.79
25	6,255	1,376	22.57	26	4	0.58	3.52
50	12,505	2,751	20.74	33	11	1.64	10.35
100	25,005	5,501	20.48	32	13	3.95	33.25

smallest CPU time on the smallest instance but fails to solve the rest of the problems. On the other hand, the D^2 method solves all the problems in a reasonable amount of time. The last problem instance in the table is the largest and has $1,000,010$ variables and $120,010$ constraints! As shown in Figure 5.2 the performance of the D^2 algorithm is linear with increasing problem size. This is a desired algorithmic behavior for scalability. We also got similar results by increasing the number of clients to 75 (problem SSLP_10_75) but the computation times are about 1.4 times larger on average for all the scenarios.

Table 5.4: Computational Results for Problem Instance SSLP_10_50

				D^2	D^2	D^2	CPLEX	
Scens	Bins	Constrs	$\%Z_{IP}$ Gap	Iters	Cuts	CPU	CPU	% Gap
5	2,510	301	10.49	209	189	78.25	80.53	
10	5,010	601	11.38	264	257	171.49	$> 10,000$	0.19
25	12,510	1,501	10.81	286	281	248.81	$> 10,000$	0.34
50	25,010	3,001	10.89	252	250	295.95	$> 10,000$	0.44
100	50,010	6,001	11.07	300	299	480.46	$> 10,000$	9.02
500	250,010	30,001	10.75	309	307	1902.20	$> 10,000$	38.17
1,000	500,010	60,001	11.07	322	321	5410.10	$> 10,000$	99.60
2,000	1,000,010	120,001	11.01	308	307	9055.29	$> 10,000$	46.24

Figure 5.3 shows a typical graph of convergence of upper and lower bounds when applying the D^2 method to SSLP_10_50. We note that the bounds have been translated so that they are nonnegative. The results shown are for problem instance SSLP_10_50 with 100 scenarios. As can be seen in the figure, the lower bound increases close to the optimal value in less than half the total number of iterations. However, good upper bounds are calculated only after first-stage solutions stabilize and this causes the method to continue for the remaining iterations without changing the lower bound significantly. Once no changes are detected in the first-stage solution, a good upper bound is calculated by solving the MIP subproblems. This immediately lowers the upper bound. Moreover, a cut proposed in (Laporte and Louveaux, 1993) is added without any additional computation, and the method typically stops after this iteration. For smaller instances however (e.g. SSLP_5_25),

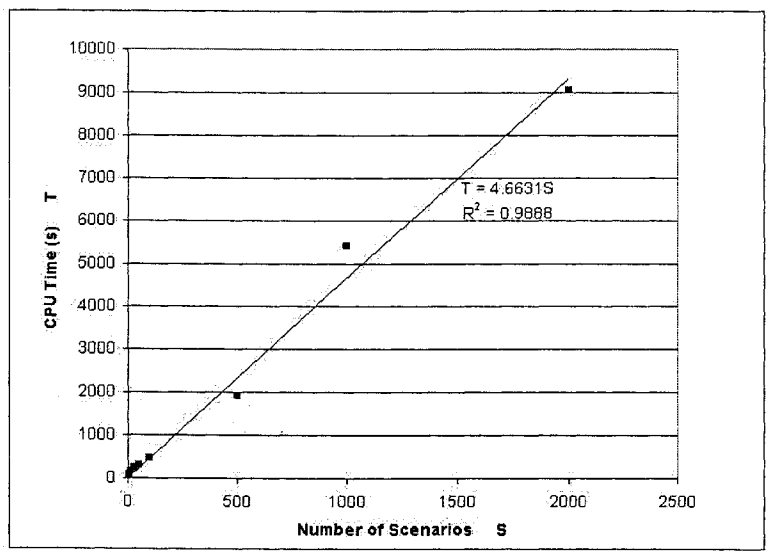

Figure 5.2: CPU Time for SSLP_10_50 Using the D^2 Method

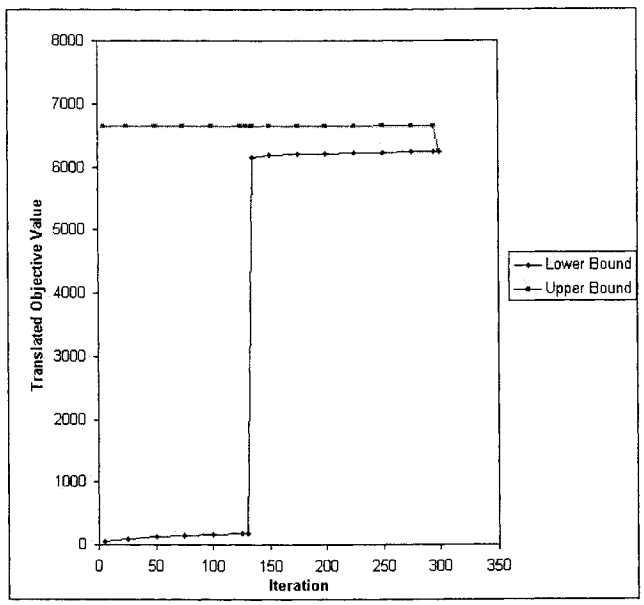

Figure 5.3: Convergence of the D^2 Algorithm for problem instance SSLP_10_50 with 100 scenarios

the D^2 cuts are sufficient to provide linear relaxations which yield binary solutions for all scenarios. As the size of m and n increase, it becomes difficult for the linear relaxation to provide binary solutions, and solving the MIP becomes necessary to improve the upper bound.

In Table 5.5 we report the results for problem SSLP_15_45 with the number of scenarios ranging over the set $5, 10$ and 15. These instances are evidently much more challenging to solve since CPLEX could not even solve the smallest instance. The D^2 algorithm solves all the problem instances but takes much longer than the time it takes to solve instances of problem SSLP_10_50. These results show that while an increase in the number of scenarios do not affect the scalability of the D^2 algorithm in an adverse way, increases in the size of the master problem, (5.2a) - (5.2d), as well as the size of the subproblems, (5.3a)- (5.3e), do have an adverse effect on scalability.

Table 5.5: Computational Results for Problem Instance SSLP_15_45

				D^2	D^2	D^2	CPLEX	
Scens	Bins	Constrs	$\%Z_{IP}$ Gap	Iters	Cuts	CPU	CPU	% Gap
5	3,390	301	6.88	146	145	110.34	$> 10,000$	1.19
10	6,765	601	6.53	454	453	1,494.89	$> 10,000$	0.27
15	10,140	901	5.62	814	813	7,210.63	$> 10,000$	0.72

Table 5.6 gives the results obtained for an experiment aimed at studying the effect of the ratio of the total server capacity to the total maximum client demand on the computational effort required by D^2 algorithm. In particular, we consider the performance of the D^2 algorithm on the problem instance SSLP_10_50 with 100 scenarios for different values of r ranging from 0.9 to 2.0. The CPLEX solver could not solve any of the DEP instances and so we excluded the results from Table 5.6.

Decreasing r results in increased computation times, an indication that tightly constrained instances are more computationally demanding. The D^2 algorithm solves all the problem instances, but it takes substantially longer to solve the

Table 5.6: Problem Instance SSLP_10_50 with 100 scenarios for different values of r

Ratio r	% Z_{IP} Gap	D^2 Iters	D^2 Cuts	D^2 Time
0.90	3.35	618	617	7896.97
1.00	6.03	543	542	5296.02
1.25	8.38	348	347	900.77
1.50	11.07	300	299	480.46
1.75	14.19	236	227	240.85
2.00	17.45	243	198	207.09

instances with $r \leq 1.00$. Despite the fact that the instances associated with $r = 1, r = 0.9$ resulted in instances with smaller gaps between an SLP relaxation, and the SMIP instance, tighter capacity constraints lead to several iterations in which the first-stage solution leads to overflows in the second-stage. Therefore, the algorithm has to overcome this "infeasibility" by generating possibly more first-stage solutions (implying more algorithmic iterations) before converging to the optimal solution.

5.4.3 Computational Experiment for SSLPs with Replications

A computational experiment to assess the performance of the D^2 algorithm on SSLP instances with replications was conducted. Five replications for SSLP with a fixed number of server locations (m), potential clients (n) and scenarios, were randomly generated with each problem instance generated as described in Section 5.4.1. To ensure independence of the random elements, different random seeds were used for generating all the random data for all the problem instances. The

Table 5.7: Computational Results for SSLPs with Replications

			D^2 Iters		D^2 Cuts		D^2 CPU Time (secs)			
m	n	Scens	Mean	Dev	Mean	Dev	Min	Max	Mean	Dev
5	25	50	20.60	2.51	10.00	2.55	0.53	0.82	0.73	0.12
5	25	100	20.60	2.88	13.00	2.00	1.03	1.84	1.48	0.32
5	50	50	25.00	5.15	5.80	3.03	0.68	1.64	0.98	0.39
5	50	100	25.60	4.16	10.60	7.09	1.25	3.95	1.89	1.15
10	50	50	233.40	25.75	229.40	27.19	138.71	295.95	228.89	63.14
10	50	100	243.00	45.31	240.60	45.63	228.68	480.00	318.12	100.93
10	50	500	298.40	18.53	297.20	18.39	1616.12	1902.20	1753.88	126.42
10	50	1000	298.40	19.93	297.20	20.29	3307.67	5410.10	3948.13	864.08
10	50	2000	308.60	11.82	307.60	11.82	8530.37	9571.04	8975.60	389.12
15	45	5	147.00	7.35	136.50	8.10	58.94	181.53	119.19	50.47
15	45	10	303.40	156.58	297.20	160.58	1306.46	2988.65	1930.00	921.65
15	45	15	739.33	64.86	738.33	64.86	5244.14	7210.63	6208.23	983.80

computational results for the experiment are reported in Table 5.7. The mean and standard deviation (Dev) for the D^2 iterations, D^2 cuts and CPU times for the five replications are reported. The minimum (Min) and maximum (Max) CPU times are

also reported. As shown in the table the number of D^2 iterations, D^2 cuts and mean CPU times increase with the number of scenarios as well as problem size as observed in the previous experiments with no replications. However, the results show that there is some variability in computation times among problem replications for some cases as indicated by the standard deviations. Note that the CPU times for SSLP instances with no replications are actually larger than the mean times reported in Table 5.7 except for the first two problem instances and the problem instance with $m = 15$, $n = 45$ and 5 scenarios. In fact these CPU times are the maximum in most of the cases.

5.4.4 Experiment with the L^2 Method

We applied the L^2 method described in Chapter 4 to some of the SSLP instances. Table 5.8 gives the computational results. The problem instances are named SSLP$m.n.S$, where m is the number of potential server locations, n is the number of potential clients, and S is the number of scenarios. As can be seen in table the L^2 algorithm and the CPLEX MIP solver applied to the DEP fail to solve the larger instances within the time limit. In particular, the CPLEX MIP solver has smaller CPU times than the L^2 algorithm on the first four (smaller) problem instances. However, unlike CPLEX, the L^2 algorithm is able to solve SSLP10.50.50 and SSLP10.50.100 to optimality.

The D^2 algorithm performs about twice as fast as the L^2 algorithm on the first four problems instances, and over six times as fast on the next two. The relative better performance of the D^2 method can be attributed to the C^3 theorem (Sen and Higle, 2000). Furthermore, scenario subproblems MIP solves are not performed at every iteration of the algorithm, but only when necessary.

Table 5.8: Computational Results for SSLP Instances

	D^2	D^2	L^2	CPU(secs)			Gap
Instance	Iters	Cuts	Iters	D^2	L^2	DEP	DEP
SSLP5.25.50	17	6	32	0.53	76.72	4.58	
SSLP5.25.100	17	10	32	1.03	379.70	14.69	
SSLP5.50.50	33	11	32	1.64	174.66	10.35	
SSLP5.50.100	32	13	32	3.95	568.87	33.25	
SSLP10.50.50	252	250	1024	295.95	1978.38	>10,000	0.44%
SSLP10.50.100	300	299	1024	480.46	2780.76	>10,000	9.02%
SSLP10.50.500	309	307	1024	1902.20	>10,000	>10,000	38.17%
SSLP10.50.1000	322	321	1024	5410.10	>10,000	>10,000	99.60%
SSLP10.50.2000	308	307	1024	9055.29	>10,000	>10,000	46.24%
SSLP15.45.5	146	145	146	110.34	>10,000	>10,000	1.19%
SSLP15.45.10	454	453	454	1,494.89	>10,000	>10,000	0.27%
SSLP15.45.15	814	813	814	7,210.63	>10,000	>10,000	0.72%

5.4.5 Preliminary Experiment with the D^2-BAC Method

The final computational experiment involves the application of the D^2-BAC algorithm to the SSLP instances. We conducted preliminary experiments to assess the performance of the algorithm by varying the number of nodes to explore in the truncated branch-and-bound (TB&B) tree. The current branch-and-bound tree follows a breadth-first strategy with node selection always favoring the node with the best objective value.

We report on one computational experiment in which the maximum number of nodes to explore in the truncated branch-and-bound tree was set at 3. Whenever there was no significant improvement in the lower bound ($< 0.001\%$) for two consecutive iterations of the algorithm, the branch-and-bound process was activated. Otherwise, no branch-and-bound was performed. Table 5.9 shows the results of the experiment.

Table 5.9: Preliminary SSLP Results Using D^2-BAC Algorithm

Instance	D^2-BAC Iters	D^2 Cuts	Total Nodes	CPU(secs)	GAP
SSLP5.25.50	19	8	123	0.57	
SSLP5.25.100	19	12	282	1.30	
SSLP5.50.50	33	11	111	1.26	
SSLP5.50.100	28	11	216	2.45	
SSLP10.50.50	287	285	9405	329.23	
SSLP10.50.100	277	274	21399	468.77	
SSLP10.50.500	322	318	177525	2773.88	
SSLP10.50.1000	368	366	492231	9599.14	
SSLP10.50.2000	147	147	617898	$> 10,0000$	100.00%
SSLP15.45.5	126	122	663	41.38	
SSLP15.45.10	420	417	7140	1501.00	
SSLP15.45.15	607	596	16806	6466.12	

The results in the table show an increase in computation times for some instances and a decrease for others compared to the results obtained for the D^2 algorithm shown in Table 5.8. There is a decrease in computation time

for SSLP5.50.50, SSLP5.50.100, SSLP15.45.5, and SSLP15.45.15. Note that SSLP10.50.2000 could not be solved within the specified computation time limit. Nonetheless, there is some decrease in computation time for problem instances SSLP15.45.5 and SSLP15.45.15, which have the largest first-stage and second-stage decision variable dimensions. Also note that the number of algorithmic iterations as well as the number of D^2 cuts added in this case has decreased significantly compared to the D^2 method.

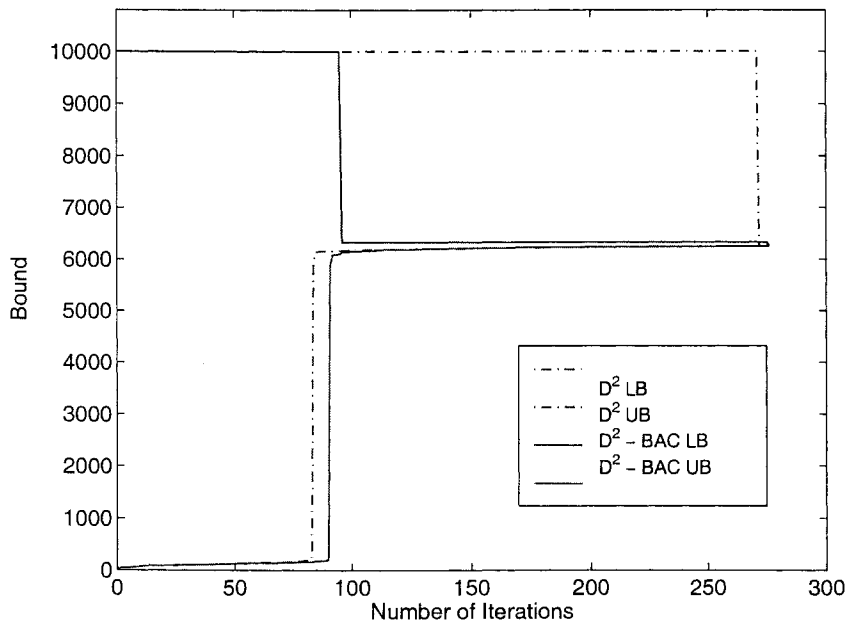

Figure 5.4: Convergence of the D^2 and D^2-BAC Algorithms for SSLP10.50.100

The convergence of upper and lower bounds for the D^2 and the D^2-BAC algorithms for problem instance SSLP10.50.100 are given in Figure 5.4. We note that the bounds have been translated so that they are nonnegative. As can be seen in the figure, the lower bound increases close to the optimal value in well less than half the total number of iterations for both algorithms. This happens a little earlier for the D^2 algorithm than for the D^2-BAC algorithm. However, good upper bounds are calculated much earlier for the D^2-BAC algorithm due to the branch-and-bound

process for each scenario subproblem, which seem to generate optimality cuts that cause the first stage solution to stabilize much faster. For the D^2 algorithm good upper bounds are calculated only after first-stage solutions stabilize, usually in the final iteration of the algorithm. After finding improved lower bounds both methods continue for the remaining iterations without changing the lower bound significantly. Nevertheless, due to early upper bounding, the D^2-BAC method has a very small percent gap early on than the D^2 algorithm.

5.4.6 SSLPs with Zonal Constraints

All the SSLP instances studied so far do not have the zonal constraints. Therefore, we generated a second set of SSLP instances with the zonal constraints added. The zonal constraints were generated arbitrarily as follows. For the problem instances with 5 potential server locations two zones were created with $z_1 = \{1, 2, 3\}$ and $z_2 = \{4, 5\}$, where $\mathcal{Z} = \{z_1, z_2\}$. For problem instances with 10 potential server locations three zones were created with $z_1 = \{1, 2, 3\}$, $z_2 = \{4, 5, 6, 7\}$ and $z_3 = \{8, 9, 10\}$. Finally, for problem instances with 15 potential server locations five zones were created with $z_1 = \{1, 2, 3\}$, $z_2 = \{4, 5, 6\}$, $z_3 = \{7, 8, 9\}$, $z_4 = \{10, 11, 12\}$ and $z_5 = \{13, 14, 15\}$. We required that at least one server be installed in each zone for all the problem instances, that is, $w_z = 1, \forall z \in \mathcal{Z}$.

The computational results for the SSLP instances are reported in Table 5.10. The problem instances are named SSLP$s.m.n.S$, where s is the problem number, m is the number of potential server locations, n is the number of potential clients, and S is the number of scenarios. The D^2 method solves all the problems and has significantly less CPU times compared to CPLEX for all the instances except SSLP2.15.45.5. CPLEX fails to solve six of the problem instances but solves all the smaller instances as well as SSLP2.15.45.5 and SSLP2.15.45.10. Finally, we note that SSLP instances with zonal constraints have significantly reduced CPU times

Table 5.10: Computational Results for Problem Set 2

				D^2	D^2	CPLEX (DEP)	
Instance	Constrs	% Z_{IP}	Iters	Cuts	CPU	CPU	Gap (%)
SSLP2.5.25.50	1503	23.60	12	4	0.23	2.32	
SSLP2.5.25.100	3003	22.98	14	5	0.66	7.82	
SSLP2.5.50.50	2753	21.09	21	3	0.76	6.61	
SSLP2.5.50.100	5503	21.60	17	3	1.28	20.16	
SSLP2.10.50.50	3004	12.45	122	121	54.62	>10800	0.42
SSLP2.10.50.100	6004	12.37	112	111	85.69	>10800	0.60
SSLP2.10.50.500	30004	12.50	150	149	499.11	>10800	33.92
SSLP2.10.50.1000	60004	12.53	134	133	986.07	>10800	75.92
SSLP2.10.50.2000	120004	12.60	134	133	2346.47	>10800	116.51
SSLP2.15.45.5	306	4.90	46	45	6.98	0.89	
SSLP2.15.45.10	606	3.11	54	53	12.71	865.85	
SSLP2.15.45.15	906	3.05	69	68	507.00	>10800	0.54

compared to the instances with none. This is an indication of decrease in problem difficulty as a result of the addition of the zonal constraints.

5.4.7 SSLPs with Unequal Scenario Probabilities

All the scenarios in the original SSLP instances have equal probabilities of occurrence. To study SSLPs with unequal scenario probabilities we generated a third set of SSLP instances by assigning randomly generated probabilities to each scenario for all the SSLP instances. The new probabilities p_ω for all $\omega \in \Omega$ were randomly generated as follows. First, for each scenario $\omega \in \Omega$ a probability $0 < \bar{p}_\omega < 1$ was generated from *uniform*(0,1). Then the probability of outcome for each scenario $\omega \in \Omega$ was computed using the formula $p_\omega = \frac{\bar{p}_\omega}{\sum_{\omega \in \Omega} \bar{p}_\omega}$, where $\sum_{\omega \in \Omega} p_\omega = 1$ as required. We used a different random seed for each problem instance to allow for independence among the problem sets.

Table 5.11: Computational Results for Problem Set 3

			D^2	D^2	CPLEX (DEP)	
Instance	% Z_{IP}	Iters	D^2 Cuts	CPU	CPU	Gap (%)
SSLP3.5.25.50	22.99	17	9	0.61	5.73	
SSLP3.5.25.100	25.77	17	10	1.21	15.65	
SSLP3.5.50.50	20.76	32	9	2.00	10.14	
SSLP3.5.50.100	20.77	27	11	4.57	34.07	
SSLP3.10.50.50	11.15	265	263	278.68	>10800	0.83
SSLP3.10.50.100	11.19	247	244	351.14	>10800	0.98
SSLP3.10.50.500	10.93	266	264	1402.37	>10800	32.29
SSLP3.10.50.1000	11.07	260	258	2895.93	>10800	67.02
SSLP3.10.50.2000	10.97	288	286	7212.39	>10800	45.93
SSLP3.15.45.5	7.14	102	100	27.88	1244.91	
SSLP3.15.45.10	6.06	181	179	137.91	>10800	1.95
SSLP3.15.45.15	4.49	195	193	931.84	>10800	1.79

Table 5.11 gives the results for the SSLP instances. The D^2 method solves all the problem instances in reasonable time and has smaller CPU times for all the problem instances compared to CPLEX. CPLEX solves the first four (smaller)

problem instances and SSLP3.15.45.5 but fails to solve the rest of the instances. Note that the assignment of random probabilities instead of the same probability for all the scenarios results in some improvement in CPU times for all the problem instances but the first four. The CPLEX MIP solver does not show improvement in the CPU times for the first four problem instances but is able to solve problem instance SSLP3.15.45.5 to optimality. These results reveal that indeed scenario probabilities have a significant effect on problem difficult for the SSLP. The case in which all scenarios are weighed equally shows more difficult in solving the problem instances compared to one in which they are not.

5.5 Summary

This chapter presents computational results with some of the largest stochastic combinatorial optimization (SCO) instances to date. While the methods discussed in this chapter are applicable to a variety of SCO problems, our computational results are presented for the stochastic server location problem (SSLP). These and other SCO problems result in very large scale instances which are comprised of loosely coupled subsystems. By taking advantage of the loosely coupled structure of SCO problems, we show that the divide-and-conquer paradigm of decomposition-coordination methods provide a highly effective algorithm, and surpasses the scalability of even the most efficiently implemented backtracking search algorithms. The study also reveals that the computational difficulty in solving SSLPs is not only dependent on the large number of variables, constraints and scenarios in the SSLP instance, but also on the zonal constraints and scenario probabilities.

CHAPTER 6

STOCHASTIC SERVER LOCATION PROBLEMS

This chapter is a further study of the stochastic server location problem (SSLP) introduced in the previous chapter. We model this problem as a two-stage SMIP problem in which we make the strategic decision of locating the servers in the first-stage. The operational/tactical decision of assigning clients to servers is made in the second-stage after the availability of the clients is revealed. We derive several valid inequalities for the SSLP and report on the computational experience with using the (D^2) algorithm for SMIP to solve several randomly generated large-scale problem instances. The study shows that it is always beneficial to use the SMIP model in making the strategic decision of obtaining optimal server locations rather than the deterministic model in which all potential clients are assumed to be available for resource allocation in the future. Real-time or on-line considerations for the SSLP are also made.

As discussed in Chapter 5, SSLPs have a limited number of "servers" with limited capacity of some resource that must be located at some given potential locations to serve known resource demands of potential "clients". The names *server* and *client* are used in a generic sense because of the variety of application domains for SSLP. The uncertainty in this problem appears in the presence or absence of the potential clients for service after the servers are located. Unlike typical stochastic facility location problems in which uncertainty is modeled as a continuous random variable, the uncertainty in the problem in consideration is unique in that it is modeled by discrete Bernoulli-type random variables. This kind of stochasticity

has been considered in other problems such as stochastic routing problems studied in Berman and Simchi-Levi (1988) and Laporte et al. (1994). Albareda-Sambola et al. (2002) present exact solutions to a class of stochastic generalized assignment problems in which the randomness is also modeled by random variables with the Bernoulli distribution.

These server location problems under uncertainty have many real-life applications. For example, Wang et al. (2003) consider the facility location problem for immobile servers with continuous stochastic demands. They present several models and provide heuristics for their solutions. Riis et al. (2004) study a server location problem for the deployment of mobile switching centers in a telecommunications network using the stochastic programming (SP) (Birge and Louveaux, 1997) approach.

We follow the SP approach in which uncertainty is incorporated into the decision making problem through a collection of future scenarios. The goal is to make the strategic decision of choosing servers to be located in such a way that the system performs well under all the selected possible future scenarios. A *scenario* refers to a set of potential clients that do materialize, and any strategic decision is evaluated against all foreseeable future scenarios. In contrast, deterministic combinatorial optimization models recommend making strategic decisions under the assumption that only one scenario is possible in the future. However, stochastic combinatorial optimization (SCO) problems such as the SSLP result in very large-scale models as a result of the need to accommodate a large number of future scenarios. Each scenario is associated with a probability of occurrence, denoted p_ω, $\omega \in \Omega$, where Ω is the entire collection of scenarios. We optimize the expected costs as is common in many such SCO models (Birge and Louveaux, 1997).

The rest of the chapter is organized as follows. In the next section we derive valid inequalities or specialized cuts for the SSLP. We report on our computational study

involving various instances of the SSLP in Section 6.2. Real-time considerations for SSLPs are discussed in Section 6.3 before ending the chapter with a summary.

6.1 Valid Inequalities

In this subsection we derive some valid inequalities or specialized cuts for the SSLP. These constraints can be used to strengthen the linear programming (LP) relaxation of the model, and can be either global or scenario-dependent. Global valid inequalities are valid for all scenarios while scenario-based inequalities are only valid for a given scenario. We derive both types of valid inequalities based on simple polyhedral considerations, minimal cover and lifted cover inequalities.

6.1.1 Simple Server-Client Constraints

Since a client i can only be assigned to a server j that has been located, then the following simple constraints are valid for the SSLP DEP formulation:

$$x_j - y_{ij}^{\omega} + y_{j0}^{\omega} \geq 0, \forall i \in \mathcal{I},\ \forall j \in \mathcal{J},\ \forall \omega \in \Omega. \tag{6.1}$$

Constraints (6.1) tighten the linear programming relaxation of the DEP model. However, the total number of these constraints may be very large. For example, if we denote the number of potential server locations by m, the number of potential clients by n, and the number of scenarios by S, then the total number of these constraints is equal to mnS. In general S is large and therefore, adding constraints (6.1) to the SSLP model can result in a very large problem instance. We note that constraint (6.1) has the artificial variable y_{j0}^{ω} which should have a high penalty in the objective function. This is different from common rules used for formulating deterministic integer programs. The artificial variable is required due to the fact the SSLP model may allow a client to be assigned to servers that are not located (see Chapter 5).

6.1.2 Minimal Cover Inequalities

We now develop minimal cover inequalities for the SSLP as follows. Let the resource demand constraints (5.1d or 5.3b) for the SSLP without the overflow variables be given by

$$\sum_{i\in\mathcal{I}} d_{ij}y_{ij}^{\omega} \leq u_j x_j, \forall j \in \mathcal{J}, \tag{6.2}$$

and let for a server $j \in \mathcal{J}$ and scenario $\omega \in \Omega$

$$Y_j^{\omega}(x_j) = \left\{ y_{ij}^{\omega} \in B^n : \sum_{i\in\mathcal{I}} d_{ij}y_{ij}^{\omega} \leq u_j x_j \right\}. \tag{6.3}$$

Now let the set $\mathcal{N} = \{1, ..., \ell\}$, where $\ell = mn$, and assume that $d_{ij} \geq 0$ $\forall i \in \mathcal{I}$ and $\forall j \in \mathcal{J}$. Then $\mathcal{C}(x_j) \subseteq \mathcal{N}$ is a cover if

$$\sum_{i\in\mathcal{C}(x_j)} d_{ij} > u_j. \tag{6.4}$$

Let $\mathcal{C}(x_j)$ be a cover and let $\mathcal{C}^+(x_j)$ denote a subset of $\mathcal{C}(x_j)$ for which $d_{ij} > 0$ for all $i \in \mathcal{C}^+(x_j)$.

Proposition 1. *If $\mathcal{C}^+(x_j) \subseteq \mathcal{N}$ is a cover, then the cover inequality*

$$\sum_{i\in\mathcal{C}^+(x_j)} y_{ij}^{\omega} - y_{j0}^{\omega} \leq (|\mathcal{C}^+(x)| - 1)x_j \tag{6.5}$$

is valid for $Y_j^{\omega}(x_j)$, and thus valid for SSLP.

Proof. (see Wolsey (1998)) We show that if for some $j \in \mathcal{J}$ vector $y_j^{\omega\mathcal{C}'(x_j)} = \{y_{ij}^{\omega}\}_{\forall i\in\mathcal{C}'(x_j)}$ does not satisfy the inequality (6.5), then $y_j^{\omega\mathcal{C}'(x_j)} \notin Y_j^{\omega}(x_j)$. If $\sum_{i\in\mathcal{C}^+(x_j)} y_{ij}^{\omega\mathcal{C}'(x_j)} > (|\mathcal{C}^+(x_j)| - 1)x_j$ then $|\mathcal{C}'(x_j) \bigcap \mathcal{C}^+(x_j)| = |\mathcal{C}^+(x_j)|$ and thus $\mathcal{C}^+(x_j) \subseteq \mathcal{C}'(x_j)$. Then $\sum_{i=1}^{n} d_{ij}y_{ij}^{\omega\mathcal{C}'(x_j)} = \sum_{i\in\mathcal{C}'(x_j)} d_{ij} \geq \sum_{i\in\mathcal{C}^+(x_j)} d_{ij} > u_j x_j$. Thus, $y_j^{\omega\mathcal{C}'(x_j)} \notin Y_j^{\omega}(x_j)$. □

This cover inequality remains valid for any value (0 or 1) of the first-stage variable x_j and for any scenario $\omega \in \Omega$. Thus the inequality can be viewed as a global constraint and it imposes the restriction that if a server j is located in the first-stage, then the total demand of the clients to be assigned to that server cannot exceed the capacity of the server.

The drawback with generating cover inequalities is that since the number of variables and constraints is generally large, there is potentially an exponential number of these inequalities that can be generated. Nevertheless, one can adopt a procedure in which a fixed number of cover inequalities are generated for each knapsack constraint in the problem. The cover inequality (6.5) is *minimal* if $\mathcal{C}^+(x_j) \setminus \{i\}$ is not a cover for any $i \in \mathcal{C}^+(x_j)$. We outline two procedures for generating minimal covers, *Minimal Cover 1* and *Minimal Cover 2*, which differ in the way the d_{ij}'s for each $j \in \mathcal{J}$ are ordered:

Procedure *Minimal Cover 1*

begin

Set $cover \leftarrow false, sum \leftarrow 0$;

Order the $d_{ij} > 0$'s in non-increasing order $d_{1j} \geq ... \geq d_{\ell j}$;

for $i \leftarrow 1, ..., \ell$

$sum \leftarrow sum + d_{ij}$;

$count \leftarrow count + 1$;

if $(sum > u_j)$

$cover =$ true;

$end = i$

break;

end if

end for

if ($cover$ = true)

Minimal cover: $\sum_{i=1}^{size} y_{ij}^{\omega} \leq (end-1)x_j$;

else

No cover generated;

end if-else

end Procedure

The *Minimal Cover 1* procedure either returns a minimal cover for a given knapsack constraint or determines that none exists. Additional minimal covers may be generated based on the first minimal cover obtained through a simple modification of the above procedure. Unlike *Minimal Cover 1* procedure, the next procedure starts out by rearranging the d_{ij}'s in non-decreasing order:

Procedure *Minimal Cover 2*

begin

Set $cover \leftarrow false, sum \leftarrow 0$, $start \leftarrow 0$;

Order the d_{ij}'s in non-decreasing order $d_{1j} \leq ... \leq d_{\ell j}$;

for $i \leftarrow 1, ..., \ell$

if $d_{ij} > 0$;

$sum \leftarrow sum + d_{ij}$;

else

$start \leftarrow start + 1$;

if $(sum > u_j)$

$cover$ = true;

$end = i$

break;

end if

end for

if ($cover$ = true)

$sum \leftarrow sum - d_{start,j}$;

while $(sum > u_j)$

$start \leftarrow start + 1$;

$sum \leftarrow sum - d_{start,j}$;

end while

Minimal cover: $\sum_{i=start}^{end} y_{ij}^{\omega} \leq (end - start)x_j$;

else

No cover generated;

end if-else

end Procedure

The above procedure also either returns a minimal cover for a given knapsack constraint or determines that none exists. Again, additional minimal covers may be generated based on the first minimal cover obtained through a simple modification of the above procedure.

6.1.3 Lifted Cover Inequalities

The minimal cover inequalities derived in the previous subsection may not be as strong as possible. Therefore, we need a procedure that allows us to find facet-defining covers. Wolsey (1998) describes a procedure to lift cover inequalities resulting in facet-defining inequalities for $\text{conv}(Y_j^\omega(x_j))$ when $\mathcal{C}^+(x_j)$ is a minimal cover and $d_{ij} \leq u_j x_j$ for all $i \in \mathcal{I}$ for a given $j \in \mathcal{J}$. The basic idea is to find, for a given $j \in \mathcal{J}$, the best possible values for α_{ij} for $i \in \mathcal{N} \setminus \mathcal{C}^+(x_j)$ such that the inequality

$$\sum_{i \in \mathcal{C}^+(x_j)} y_{ij}^\omega + \sum_{i \in \mathcal{N} \setminus \mathcal{C}^+(x_j)} \alpha_{ij} y_{ij}^\omega \leq (|\mathcal{C}^+(x_j)| - 1)x_j \tag{6.6}$$

is valid for $Y_j^\omega(x_j)$, and thus valid for SSLP. The cover inequality (6.6) is valid for $Y_j^\omega(x_j)$ and dominates the inequality (6.5). Thus we can adopt the procedure for lifting cover constraints given in Wolsey (1998)

6.1.4 Scenario-Based Valid Inequalities

We now turn to scenario-based valid inequalities based on the resource demand constraints and the client availability scenario outcome. We now consider strengthening resource demand constraints

$$\sum_{i \in \mathcal{I}} d_{ij} y_{ij}^\omega - y_{j0}^\omega \leq u_j x_j, \quad \forall j \in \mathcal{J}, \tag{6.7}$$

for each scenario $\omega \in \Omega$ subproblem. Let $\mathcal{I}^\omega \subseteq \mathcal{I}$ be the set of clients that actually show up for resource allocation in scenario $\omega \in \Omega$. Then these constraints can be strengthened as follows:

Proposition 2. *The following constraints are valid for SSLP DEP formulation:*

$$\sum_{i \in \mathcal{I}^\omega} d_{ij} y_{ij}^\omega - y_{j0}^\omega \leq u_j x_j, \forall j \in \mathcal{J}. \tag{6.8}$$

For a given server $j \in \mathcal{J}$ constraint (6.8) is stronger than the corresponding constraint (6.7) since we restrict the resource demands to only those clients that are present in the scenario. Thus constraints (6.8) are only valid for each scenario and can easily be derived after each scenario realization. Unlike the D^2 cuts derived via the disjunctive decomposition method, these cuts do not maintain the fixed recourse property as required by the D^2 method. Therefore, these cuts cannot be used under the D^2 setting, which is applicable to the two-stage SMIP formulation with fixed recourse.

6.2 Computational Experience

In this section the results of a computational study are reported. We investigate the effect of several parameters on the effort required to solve several randomly generated SSLP instances and compare the performance of the CPLEX 7.0 MIP solver (ILOG, 2000) on the DEP instances to that of the D^2 algorithm on the corresponding two-stage problem instances.

6.2.1 Problem Instance Generation

The problem instances we experiment with are some of the randomly generated SSLP instances with equal scenario probabilities reported in Chapter 5. Here we consider extensions to these instances, which act as the "benchmark", by adding

the valid inequalities derived in Section 6.1. The first problem set is an extension of the benchmark test set by adding simple server constraints (6.1) to the problem instances while the second problem set has simple server constraints (6.1) and minimal cover inequalities (6.5) added.

The addition of the valid inequalities to the SSLP instances greatly increases the number of constraints in the problem. We also note that the valid inequalities are generated *a priori* and added to the SSLP instances before optimization, while the D^2 cuts are generated sequentially and added to the problem instance during the execution of the D^2 algorithm. Two minimum cover cuts were generated for each demand resource constraint in the SSLP instance using the *Minimal Cover 1* procedure and the *Minimal Cover 2* procedure, respectively. Both the procedures were implemented in C.

The problem instances are named SSLP$s.m.n.S$ as in the previous chapter, where s is the problem set number, m is the number of potential server locations, n is the number of potential clients, and S is the number of scenarios. We set $s = 2$ for the problem set with simple client/server constraints added and $s = 3$ for the problem set with both simple client/server constraints and minimal cover inequalities added. We omit s for the benchmark problem instances.

Table 6.1 gives the problem characteristics for the benchmark problem instances. The headings of the table are as follows: "Constrs" is the number of constraints, "Bins" is the number of binary decision variables, "Cvars" is the number of continuous decision variables, and "Dens" is the density of the constraint matrix of the SSLP instance.

Table 6.1: Set 1 SSLP Instance Dimensions

	DEP				SUBPROBLEM			
Instance	Constrs	Bins	Cvars	Dens	Constrs	Bins	Cvars	Dens
SSLP5.25.50	1,501	6,255	250	0.0013	30	130	5	0.0715
SSLP5.25.100	3,001	12,505	500	0.0007	30	130	5	0.0715
SSLP5.50.50	2,751	12,505	250	0.0007	55	255	5	0.0393
SSLP5.50.100	5,501	25,005	500	0.0004	55	255	5	0.0393
SSLP10.50.50	3,001	25,010	500	0.0007	60	510	10	0.0345
SSLP10.50.100	6,001	50,010	1,000	0.0003	60	510	10	0.0345
SSLP10.50.500	30,001	250,010	5,000	0.0001	60	510	10	0.0345
SSLP10.50.1000	60,001	500,010	10,000	0.0009	60	510	10	0.0345
SSLP10.50.2000	120,001	1,000,010	20,000	0.0007	60	510	10	0.0345
SSLP15.45.5	301	3,390	75	0.0066	60	690	15	0.0340
SSLP15.45.10	601	6,765	150	0.0033	60	690	15	0.0340
SSLP15.45.15	901	10,140	375	0.0022	60	690	15	0.0340

6.2.2 Computational Results

All the experiments were run on a Sun 280R with 2 UltraSPARC-III+ CPUs running at 900 MHz. To optimize the large-scale SSLP DEP formulation for each of the problem instances the CPLEX MIP solver parameters are set at the following values: "set mip emphasis 1" (emphasizes looking for feasible solutions), "set mip strategy start 4" (uses barrier at the root), and "branching priority order on x" (first branches on any fractional component of x before branching on y). These settings gave us the best CPU times. A CPU time limit of $10,800$ seconds (3 hrs) was imposed and any problem instance run taking more than this time limit was stopped and the percent gap from optimality reported. All the problems that took less than this time limit converged to an optimal solution and the percentage gap between the lower bound and the upper bound was equal to 0%.

Table 6.2 shows some parameters for analyzing the goodness of the stochastic model (see e.g. Birge and Louveaux (1997)) compared with a deterministic one in which all potential clients are assumed to be available for resource allocation in the future. The headings of the table are as follows. In the first column is the problem instance name. The second column shows VSS_D, the value of the stochastic solution and is expressed as

$$VSS_D = Z_{IP} - EEV_D$$

where, Z_{IP} is the optimal objective value of SSLP (shown in the fourth column), EV_D is the solution value for the deterministic scenario problem in which all potential clients are assumed to be available, and EEV_D (shown in the third column) is the expected result of using the EV_D solution and is expressed as

$$EEV_D = \sum_{\omega \in \Omega} p_\omega Z^\omega_{IP_D}$$

where, $Z^\omega_{IP_D}$ is the optimal objective value of scenario $\omega \in \Omega$ subproblem with the first-stage solution being the EV_D solution. The fifth column shows the optimal

objective value of the LP relaxation of the SSLP problem instance (which can be solved using stochastic linear programming), and $\%Z_{IP}$ Gap is the percentage difference between the optimal value of the SSLP instance, and its continuous relaxation and can be expressed as

$$\%Z_{IP} = \frac{(Z_{LP} - Z_{IP})}{Z_{LP}} \times 100.$$

Note that determining the value of the stochastic solution (VSS) using the

Table 6.2: Stochastic Solutions for SSLP Instances

Instance	% VSS_D	EEV_D	Z_{IP}	Z_{LP}	% Z_{IP} Gap
SSLP5.25.50	25.44	-90.660	-121.600	-160.063	24.03
SSLP5.25.100	23.77	-97.100	-127.370	-169.667	24.93
SSLP5.50.50	8.32	-285.560	-311.480	-392.995	20.74
SSLP5.50.100	8.37	-296.600	-323.700	-407.051	20.48
SSLP10.50.50	21.93	-284.680	-364.640	-409.194	10.89
SSLP10.50.100	23.12	-272.290	-354.190	-398.297	11.07
SSLP10.50.500	24.08	-265.048	-349.136	-391.185	10.75
SSLP10.50.1000	23.72	-268.287	-351.711	-395.477	11.07
SSLP10.50.2000	24.63	-261.718	-347.262	-390.231	11.01
SSLP15.45.5	77.21	-59.800	-262.400	-280.490	6.45
SSLP15.45.10	72.44	-71.800	-260.500	-278.689	6.53
SSLP15.45.15	73.47	-67.269	-253.602	-253.602	5.62

expected value (EV) as defined in Birge and Louveaux (1997) is not appropriate here since the SSLP model requires that the scenario subproblem right hand side client availability vector $h(\omega)$ for scenario $\omega \in \Omega$ be a Bernoulli random variable. Therefore, computing the expected scenario for determining EV (expected value) could violate the (0-1) requirements for the elements of $h(\omega)$ (see Chapter 5).

The VSS_D values are positive for all the instances with the % VSS values as shown in Table 6.2, an indication that it always pays off to use the stochastic model in making the strategic decision of obtaining optimal server locations other than the deterministic model where it is assumed that all potential clients will be available for resource allocation in the future. For example, for the problem instance SSLP.15.45.15 the optimal strategic decision resulting from using the

stochastic approach is the installation of five servers (1,4,8,11,15) while that from the deterministic model is nine servers (1,4,7,8,11,12,13,14, and 15). Also note that for all problem instances, Z_{IP} Gap is fairly large, which implies that the stochastic linear programming (continuous) relaxation of these SSLP instances does not provide very good approximations to the original problem.

In Table 6.3 we present the computational results for benchmark problem (SSLP instances with no specialized cuts added) as reported in Chapter 5 for convenience. In the table the headings are as follows: "Iters" is the number of algorithmic iterations for the D^2 algorithm, "D^2 Cuts" gives the number of D^2 cuts generated, "D^2 Time" is the CPU time in seconds to solve the two-stage problem instance using the D^2 algorithm, "DEP Time" is the CPU time in seconds to solve the DEP instance directly using the CPLEX MIP solver, and "Gap" is the percentage B&B optimality gap.

Table 6.3: Computational Results for the Benchmark Problem Set

	D^2	D^2	D^2	CPLEX (DEP)	
Instance	Iters	Cuts	CPU	CPU	Gap
SSLP5.25.50	17	6	0.53	4.58	0
SSLP5.25.100	17	10	1.03	14.69	0
SSLP5.50.50	33	11	1.64	10.35	0
SSLP5.50.100	32	13	3.95	33.25	0
SSLP10.50.50	252	250	295.95	>10,800	0.44
SSLP10.50.100	300	299	480.46	>10,800	9.02
SSLP10.50.500	309	307	1902.20	>10,800	38.17
SSLP10.50.1000	322	321	5410.10	>10,800	99.60
SSLP10.50.2000	308	307	9055.29	>10,800	46.24
SSLP15.45.5	146	145	110.34	>10,800	1.19
SSLP15.45.10	454	453	1,494.89	>10,800	0.27
SSLP15.45.15	814	813	7,210.63	>10,800	0.72

Table 6.4 shows the results for the second problem set. The addition of the simple client/server constraints (6.1) results in a significant increase in the total number of constraints in each problem instance. For example, the number of constraints in the DEP problem instance SSLP2.10.50.2000 with over a million

variables increases from 120,001 to 1,120,001! Nevertheless, there is a general improvement (about 20% reduction) in the CPLEX CPU times in solving the relatively smaller DEP problem instances except SSLP2.5.25.100. Moreover, the CPLEX MIP solver is able to solve problem instance SSLP2.15.45.10 to optimality, with slightly improved B&B optimality gaps for SSLP2.15.45.5 and SSLP2.15.45.15. The performance of CPLEX on DEP instances SSLP2.10.50.1000 and SSLP2.10.50.2000 deteriorates, probably due to the increase in the problem size.

Table 6.4: Computational Results for SSLP with Simple Client/Server Constraints

			D^2	D^2	D^2	CPLEX (DEP)	
Instance	Constrs	% Z_{IP}	Iters	Cuts	CPU	CPU	Gap
SSLP2.5.25.50	7751	4.40	12	4	1.46	3.50	0
SSLP2.5.25.100	15501	4.83	14	5	2.64	18.4	0
SSLP2.5.50.50	15251	0.94	27	5	4.22	8.24	0
SSLP2.5.50.100	30501	0.26	26	6	9.00	26.91	0
SSLP2.10.50.50	28001	1.84	209	206	350.40	>10,800	0.70
SSLP2.10.50.100	56001	1.78	217	216	535.49	>10,800	1.03
SSLP2.10.50.500	280001	1.42	231	230	2228.99	>10,800	52.73
SSLP2.10.50.1000	560001	1.56	259	258	4943.28	>10,800	∞
SSLP2.10.50.2000	1120001	2.12	236	235	9454.13	>10,800	∞
SSLP2.15.45.5	3676	2.15	49	48	24.01	>10,800	0.98
SSLP2.15.45.10	7351	3.46	126	125	132.32	8879.89	0
SSLP2.15.45.15	11026	2.62	334	333	1251.59	>10,800	0.60

The CPU times for the D^2 algorithm increases in general especially on the first four instances (smaller instances) but decreases by an average of 84% on the last three instances. The addition of the simple client/server constraints to the smaller problem instances results increased CPU times per scenario subproblem LP relaxation due to their increased size. However, the addition of the constraints to the problem instances with the largest decision space dimensions results in significant gains for the D^2 algorithm. Note that there is a significant decrease in the % Z_{IP} values due to the tightening of the feasible region for the LP relaxation. Thus the scenario subproblem LP relaxations in the D^2 algorithm provide improved objective

function values in general.

Table 6.5 gives the computational results for the third problem set. The table heading "Covers" gives the total number of minimum cover constraints in the DEP instance. Note that the addition of both simple client/server constraints and minimum cover constraints results in a further increase in the total number of constraints in the problem instances. The results show some improvement in the CPU times for the D^2 algorithm on some of the problem instances, with significant gains on the last three problem instances (with the largest decision space dimensions). The performance of the CPLEX MIP solver deteriorates for all the problem instances. In this case the solver cannot solve the DEP instance SSLP5.15.45.10 to optimality. We can attribute the apparent problem difficulty to the increased problem sizes.

Table 6.5: Computational Results for SSLPs with Simple Client/Server Constraints and Cover Constraints

				D^2	D^2	D^2	CPLEX (DEP)	
Instance	Constrs	Covers	$\%Z_{IP}$	Iters	Cuts	CPU	CPU	Gap
SSLP3.5.25.50	8901	500	4.40	14	5	1.68	9.67	
SSLP3.5.25.100	17751	1,000	4.83	17	6	1.96	44.05	
SSLP3.5.50.50	17701	500	0.94	26	6	4.07	21.08	
SSLP3.5.50.100	35301	1,000	0.26	26	7	7.66	74.18	
SSLP3.10.50.50	30201	1,000	1.84	216	204	316.77	>10,800	1.02
SSLP3.10.50.100	60301	2,000	1.78	230	218	589.99	>10,800	1.45
SSLP3.10.50.500	301101	10,000	1.42	244	233	2386.85	>10,800	121.48
SSLP3.10.50.1000	602101	20,000	1.56	263	252	4976.73	>10,800	∞
SSLP3.10.50.2000	1204101	40,000	2.12	255	244	10482.95	>10,800	∞
SSLP3.15.45.5	3926	150	2.15	60	44	24.23	>10,800	1.52
SSLP3.15.45.10	7761	300	3.46	182	166	256.15	>10,800	0.28
SSLP3.15.45.15	11596	450	2.62	335	319	2283.50	>10,800	0.57

We conducted further computational experiments to investigate the effect of the lifted minimum cover constraints and the scenario-based constraints on the SSLP difficulty. The findings from these experiments are not reported but can be summarized as follows. The addition of minimum cover constraints and lifted minimum cover constraints without the simple client/server constraints,

respectively, resulted in significant gains (over 50 %) in computation times for the last three problem instances only. There were no significant gains on the other problem instances. The addition of scenario-based constraints to the SSLP instances did not result in significant improvements in computation times in general.

6.3 Real-Time Considerations

We now consider the real-time application of the D^2 algorithm to the SSLP in making recourse actions. We place special emphasis on the real-time character of the SSLP when the operational/tactical decisions have to be made in real-time or on-line. Note that this takes place after the strategic decision has been made, that is, after the servers have been located. The real-time or on-line decision making process poses a need for sufficiently short response times while faced with the burden of incomplete knowledge. In particular, the scenarios to be realized over time are not known with certainty when making the strategic decision. However, the operational/tactical decision is made after the scenario information is revealed and requires the optimal solution to the second-stage scenario subproblem.

We note that the scenario subproblem is in essence a resource allocation problem in which limited resources have to be allocated to available clients at minimum cost or maximum profit. Such problems are common in many real-time or on-line applications characterized by some uncertainty in the data. For example, in real-time distributed computer applications often computer tasks (clients) are required to be executed on computers (servers) based on unknown availability or load-balance conditions. Scheduling under uncertainty also calls for real-time decision making under uncertainty. For example, Sand and Engell (2003) follows a two-stage stochastic integer programming approach to real-time scheduling in processing industries flexible batch plants. The authors point out that virtually all uncertainty conscious models in this particular application area are based on a

defensive strategy, which avoids extensive rescheduling activities. They consider the SP approach as one in which the need for recourse actions is not deemed a burden, but as optimization potential.

We now propose how to make timely recourse decisions for the SSLP for real-time applications. Even though what follows is specific to the SSLP, we note that it carries on to other applications. In order to make an operational/tactical decision the scenario subproblem has to be solved for the realized scenario. However, when making the strategic decision x, the problem is solved using the D^2 algorithm, thus providing not only the optimal strategic decision x^* but also the optimal tactical decisions $y_{ij}^{*\omega}$ for each scenario $\omega \in \Omega$. Hence the collection of the optimal tactical decisions $\{y_{ij}^{*\omega}\}_{\omega\in\Omega}$ provides *optimal policies* to be used in making decisions in real-time or on-line that require short response times. Therefore, the optimal policies can be stored in some database after the strategic decision is made. Consequently, it is not necessary to solve scenario subproblem in order to make a tactical decision in the future when a scenario $\omega \in \Omega$ is revealed. The optimal solution for the scenario can simply be looked up or retrieved from the database in significantly less time compared to optimizing the scenario subproblem.

When a scenario $\omega \in \Omega$ is realized no computations are necessary. However, it may so happen that a scenario $\omega' \notin \Omega$ is revealed. The obvious thing to do in this case is to simply solve the scenario subproblem "from scratch" for the new scenario ω'. The scenario subproblem is an integer program and may potentially take longer to solve than the time required to make a quick decision in real-time or on-line. Therefore, we propose taking advantage of the information provided by the D^2 method as a "warm start" for solving the new scenario subproblem. The motivation for doing this is the need to make an optimal decision in a timely manner as required by the real-time application.

When solving the two-stage SSLP using the D^2 approach, the D^2 algorithm

generates the D^2 cuts via the common-cut-coefficient theorem (Sen and Higle, 2000) and appends them to subproblem LP relaxation. The D^2 cuts tightens the scenario subproblem LP relaxation feasible set, potentially leading to integral solutions. Consequently the second-stage subproblem LP relaxation takes the following form:

$$\text{Min } q(\omega)^\top y, \tag{6.9a}$$

$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x, \tag{6.9b}$$

$$(\pi^k)^\top y \geq \pi_c^k(x,\omega),\ k \in \Theta \tag{6.9c}$$

$$y \in \Re_+^{n_2}. \tag{6.9d}$$

In formulation (6.9) k is the algorithmic iteration index, Θ is the set of algorithm iteration indices at which a D^2 cut is generated, and the π^k's are the so called common-cut-coefficients, and are valid for all the scenarios. The right-hand side of the D^2 cut is determined for each scenario $\omega \in \Omega$. Therefore, we suggest dealing with problem (6.9) for any new scenario $\omega' \notin \Omega$. First, the right-hand side $\pi_c^k(x,\omega')$ is computed for each $k \in \Theta$ and then the subproblem is re-optimized with integer restrictions imposed. This problem provides a tighter formulation for the new scenario and may potentially result in reduced computation time for making an optimal tactical decision for the new scenario. We refer the reader to Chapter (Sen and Higle, 2000) for details on computing $\pi_c^k(x,\omega')$ for each $k \in \Theta$.

6.4 Summary

This chapter has provided a further study of the SSLP. In order to strengthen the LP relaxation of the model, valid inequalities or specialized cuts for the SSLP have been derived. The computational results indicate that there is significant improvement in computation time for some SSLP instances by the addition of the specialized cuts to the model. The study shows that it always pays off to use the SP

in making the strategic decision of obtaining optimal server locations rather than using the deterministic approach in which it is assumed that all potential clients will be available for resource allocation in the future. Finally, considerations for real-time or on-line operational/tactical decision making under uncertainty for the SSLP have been made.

CHAPTER 7

PERFORMANCE OF THE D^2 METHOD FOR MIXED-INTEGER (BINARY) SECOND STAGE

The computational results presented in Chapter 5 were based on purely combinatorial problems. In this chapter, we study the performance of the D^2 method on models involving both continuous and binary variables in the second stage. The specific classes of models that we use in our computational study are bipartite matching and strategic supply chain planning under uncertainty. While the former is a purely combinatorial model, it turns out that the data set provided in Kong and Schaefer (2004) has the property that the continuous relaxation of the combinatorial problem provides a solution to the combinatorial problem.

Since the studies of this chapter allow continuous variables in the second stage, we report our results with the bipartite matching problem in this section. The bipartite matching and strategic supply chain planning problems under uncertainty are generally referred to as stochastic matching (SM) (Kong and Schaefer, 2004) and stochastic strategic supply chain (SSCh) planning (Alonso-Ayuso et al., 2003) problems, respectively.

The models for the two problem classes quite differ in the type of second-stage decision variables as well as how uncertainty is revealed. The SM model has pure binary second-stage while the SSCh model has mixed-binary second-stage. However, due to the unimodularity of the recourse matrix and the integrality of second-stage right-hand side, the second-stage subproblem LP relaxation for the SM model provide integral solutions. Thus we treat the second-stage decision variables

as continuous. In the SM problem the uncertainty appears only in the objective function, while in the SSCh model it appears in both the objective function and the right-hand side data. Recall that the uncertainty in the SSLP appears only in the right-hand side data.

The uncertainty in the two problems dictate that the strategic decisions be made "here-and-now" while the operational or tactical decisions are determined after more precise information becomes available. This leads to a two-stage decision making process as in the SSLP in which the strategic decisions are made in the first-stage and the operational/tactical decisions are made in the second-stage when the uncertainty is revealed. Such types of problems find real life applications in a variety of sciences and engineering. However, solving such problems is a difficult undertaking due the integrality requirements and the uncertainty in the problem data. Furthermore, these problems are characterized by a large-scale nature and can easily become intractable even for the state-of-the-art commercial solvers.

The rest of this chapter organized as follows. In the next section we describe the SM model. The SSCh model is described in Section 7.2. We report on our computational experience with the algorithms in Section 7.3 and end the chapter with a summary.

7.1 Stochastic Matching

The second example application problem we consider is the two-stage stochastic bipartite matching problem studied by Kong and Schaefer (2004). The SM problem is a stochastic extension of the maximum-weight matching problem of Edmonds (1965) and can be formally stated as follows. A graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, is given. For each edge $e \in E$ there is a first-stage edge weight c_e and second-stage weights d_e^ω for each scenario $\omega \in \Omega$, with corresponding scenario probability p_ω. The goal is to maximize the total expected

edge weight in these matchings. The SM problem formulation can be stated as follows:

$$\text{Max} \sum_{e \in E} c_e x_e + \sum_{\omega \in \Omega} p_\omega \sum_{e \in E} d_e^\omega y_e^\omega \tag{7.1a}$$

$$\text{s.t.} \sum_{e \in \delta(v)} x_e + \sum_{e \in \delta(v)} y_e^\omega \leq 1, \forall v \in V, \forall \omega \in \Omega, \tag{7.1b}$$

$$x_e \in \{0,1\}, y_e^\omega \in \{0,1\}, \forall e \in E, \forall \omega \in \Omega. \tag{7.1c}$$

Constraints (7.1a) dictate that a matching is made either in the first-stage or the second-stage, while constraints (7.1c) are the binary restrictions on first-stage and second-stage decision variables. Formulation (7.1) is the large-scale DEP formulation and can be decomposed into two-stage SMIP as follows:

$$\underset{x_e \in \{0,1\}, \forall e \in E}{\text{Max}} c_e x_e + E[f(x, \tilde{\omega})], \tag{7.2a}$$

where the vector $x = \{x_e\}_{e \in E}$ and for each scenario realization ω of $\tilde{\omega}$ we have:

$$f(x, \omega) = \text{Max} \sum_{e \in E} d_e y_e, \tag{7.3a}$$

$$\text{s.t.} \sum_{e \in \delta(v)} y_e \leq 1 - \sum_{e \in \delta(v)} x_e, \forall v \in V, \tag{7.3b}$$

$$y_e \in \{0,1\}, \forall e \in E. \tag{7.3c}$$

The SM problem arises in many business, investment and industrial applications. For example Hauskrecht and Upfal (2001) study a stochastic *contract* matching problem with two decision stages. In the first-stage an allocation problem deciding which contracts to buy and sell has to be solved. The second-stage is to solve a matching problem to decide the best coverage of sell contracts after observing the actual failure configuration. They formulate their problem as a stochastic linear program (SLP) with the ultimate goal of maximizing expected profits and give a specific example application of trading communication bandwidth through

unreliable satellite and/or ground transmission equipment and its channels. The goal is to find the best combination of lease (buy) and sell contracts maximizing the expected profit. They take into account the probability of failures, flexibility of equipment coverage, profits/costs for selling/buying respective contracts and penalties for breaching sell contracts.

7.2 Strategic Supply Chain Planning Under Uncertainty

We consider the two-stage stochastic programming (SP) approach for SSCh as presented in Alonso-Ayuso et al. (2003). Other recent work in this area include that of Escudero et al. (1996), MirHassani et al. (2000) and Ahmed et al. (2003). The essence of supply chain planning consists of determining the plant location, plant sizing, product selection, product allocation among plants and vendor selection for raw materials. The uncertain parameters include product net price and demand, raw material supply cost and production cost. The objective is to maximize the expected profit over a given time horizon for the investment depreciation and operations costs.

The two-stage stochastic supply chain planning problem (Alonso-Ayuso et al., 2003) we consider has the strategic decisions made in the first-stage while the operational or tactical decisions are made in the second-stage. The first-stage is devoted to strategic decisions about plants sizing, product allocation to plants and raw materials vendor selection. The second-stage deals with making tactical decisions about the raw material volume supply from vendors, product volume to be processed in plants, and stock volume of product and raw materials to be stored in plants and warehouses. Further, tactical decisions include component volume to be shipped from plants to market sources at each time period along the time horizon. All the tactical decisions are made based on the supply chain topology decided in the first-stage.

In making the strategic decisions in the first-stage it is assumed that the

information on the strategic decision costs and constraints is known. However, the information on the tactical decision costs/revenue and constraints is not known *a priori*. For example there may be randomness in the cost of product/raw materials and in the demand at different markets for selling the final products. The model given by Alonso-Ayuso et al. (2003) can be summarized as follows:

$$\text{Max } c^\top x + \sum_{\omega \in \Omega} p_\omega \left(q^\top y_\omega + q_\omega^\top z_\omega \right) \tag{7.4a}$$

$$\text{s.t. } Ax = b, \tag{7.4b}$$

$$Tx + W_1 y_\omega + W_2 z_\omega = r(\omega), \forall \omega \in \Omega, \tag{7.4c}$$

$$x \in \{0,1\}^{n_1},\ y_\omega \in \{0,1\}^{\bar{n}_2},\ z_\omega \geq 0,\ \forall \omega \in \Omega, \tag{7.4d}$$

where c, q, and q_ω are vectors of the objective function coefficients, x, y_ω, and z_ω are the decision variables, and $\omega \in \Omega$ denotes a scenario with Ω being the collection of all the scenarios. The x represents the 0-1 first-stage strategic decision variables and the y_ω and z_ω represent the 0-1 second-stage strategic decision variables and the continuous second-stage tactical decision variables for a scenario $\omega \in \Omega$, respectively. A and T are the first-stage and second-stage matrices related to the x variables, respectively; W_1 and W_2 are the matrices related to the y_ω and z_ω variables, respectively; b and $r(\omega)$ are the right-hand side vectors for the first-stage and second-stage for scenario $\omega \in \Omega$, respectively. All the parameters are appropriately dimensioned. The first-stage constraints (7.4c) include restrictions on the number of allowable plants in the supply chain and investment budgetary constraints, while the second-stage constraints (7.4d) include capacity expansion constraints and operation related constraints.

Formulation (7.4) is the DEP model for the stochastic supply chain planning problem. Alonso-Ayuso et al. (2003) follows the *scenario analysis* approach and decomposes (7.4) *scenario-wise* and proposes a branch-and-fix coordination (BFC) algorithm presented in Alonso et al. (2000) to solve the problem. In order to be commensurate with our approach, we will instead decompose the problem *stage-wise*

into a two-stage SMIP as follows:

$$\text{Max } c^\top x + E[f(x,\tilde{\omega})] \tag{7.5a}$$

$$\text{s.t. } Ax = b, \tag{7.5b}$$

$$x \in \{0,1\}^{n_1}, \tag{7.5c}$$

where for each scenario realization ω of $\tilde{\omega}$ we have the second-stage subproblem

$$f(x,\omega) = \text{Max } q^\top y_\omega + q_\omega^\top z \tag{7.6a}$$

$$\text{s.t. } W_1 y + W_2 z = r(\omega) - Tx, \tag{7.6b}$$

$$y \in \{0,1\}^{\bar{n}_2},\ z \geq 0. \tag{7.6c}$$

We again remind the reader that although the second-stage (recourse) variables y and z continue to depend on the outcome ω, this dependence is not explicitly indicated here since the subproblem for each outcome ω is decoupled from all other outcomes once a vector x is given.

7.3 Computational Experiments

We now report on our computational experience in applying the D^2 method to solving large scale two-stage SMIP problem instances of the two models from the literature. We compare some of our computational results with those obtained by the general-purpose programming system CPLEX 7.0 (ILOG, 2000) applied to the DEP formulation. All the experiments were conducted on a Sun 280R with 2 UltraSPARC-III+ CPUs running at 900 MHz. The problem instances were run to optimality or stopped when a CPU time limit of 10,800 seconds (3hrs) was reached. The large CPU times are indicative of the large-scale nature and difficulty of solving these problems. The CPU time reported for each problem instance is an average of three runs. The CPLEX MIP solver is applied to the large scale DEP formulation for each of the two-stage problem instances if possible as a bench mark. To get the

best CPU times the CPLEX parameters were set at the following values as in the SSLP experiments: "set mip emphasis 1" (emphasizes looking for feasible solutions), "set mip strategy start 4" (uses barrier at the root), and "branching priority order on x" (first branches on any fractional component of x before branching on y).

7.3.1 Stochastic Matching

The SM test set consists of a collection of nine randomly generated two-stage stochastic bipartite matching problem instances supplied by Kong and Schaefer (2004). These nine problem instances are divided into three groups. Each problem instance in the first two groups has 10 vertices in each side of the bipartition, while each instance in the last group has 20. The first-stage and second-stage edge weights are normally distributed. All weights in the problem instances in the first and third groups are restricted to be integral, whereas those in the second are unrestricted. Table 7.1 gives the characteristics of the problem instances. Columns "NS", "EW1" and "EW2" refer to the number of scenarios, the distribution of the first-stage and second-stage edge weights, respectively. Columns "Rows", "Cols", "Nonz" refer to the number of rows, columns, and non-zero elements in the DEP formulation (7.1).

Table 7.1: Stochastic Matching Problem Instance Dimensions

Instance				DEP		
Name	NS	EW1	EW2	Rows	Cols	Nonz
SM11	100	$\mathcal{N}(10, 15^2)$	$\mathcal{N}(10, 15^2)$	2,000	10,100	40,000
SM12	200	$\mathcal{N}(10, 15^2)$	$\mathcal{N}(10, 15^2)$	4,000	20,100	80,000
SM13	300	$\mathcal{N}(10, 15^2)$	$\mathcal{N}(10, 15^2)$	6,000	30,100	120,000
SM21	100	$\mathcal{N}(10, 15^2)$	$\mathcal{N}(10, 15^2)$	2,000	10,100	40,000
SM22	200	$\mathcal{N}(10, 15^2)$	$\mathcal{N}(10, 15^2)$	4,000	20,100	80,000
SM23	300	$\mathcal{N}(10, 15^2)$	$\mathcal{N}(10, 15^2)$	6,000	30,100	120,000
SM31	100	$\mathcal{N}(10, 10^2)$	$\mathcal{N}(10, 10^2)$	4,000	40,400	160,000
SM32	200	$\mathcal{N}(10, 10^2)$	$\mathcal{N}(10, 10^2)$	8,000	80,400	320,000
SM33	300	$\mathcal{N}(10, 10^2)$	$\mathcal{N}(10, 10^2)$	12,000	120,400	480,000

Computational results for the SM problem instances are given in Table 7.2. The table headings are as follows: "D^2 Iters" is the number of algorithmic iterations for the D^2 algorithm and "D^2 Cuts" is the number of D^2-cuts generated and added to each scenario subproblem LP relaxation. The next two columns give the CPU time in seconds for the D^2 algorithm and the CPLEX MIP solver applied on the corresponding DEP formulation for the two-stage problem instance, respectively. The last column gives the optimality % gap for the DEP after the time limit has been reached and the CPLEX MIP solver prematurely terminated.

Table 7.2: Computational Results for Stochastic Matching Problem Instances

			CPU(secs)		Gap
Instance	D^2 Iters	D^2 Cuts	D^2	DEP	DEP
SM11	40	0	6.38	6.34	
SM12	28	0	7.88	12.83	
SM13	17	0	7.21	19.30	
SM21	51	0	8.54	8.64	
SM22	38	0	10.50	13.84	
SM23	71	0	15.92	29.00	
SM31	87	0	136.60	777.60	
SM32	98	0	224.24	7688.98	
SM33	102	0	272.15	> 10800	28.45%

As can be seen in the table, the D^2 algorithms solves all the problem instances to optimality. The CPLEX MIP solver could not solve the last instance to optimality within the given time limit. Nevertheless, its performance on the smallest instances, SM11 and SM21, is comparable to that of the D^2 algorithm. Decomposing relatively smaller size problems may not be as beneficial due to the overhead involved in the decomposition process. However, we see that for the larger instances the D^2 algorithm makes substantial gains. We also observe that the performance of the D^2 algorithm scales very well with an increase in the number of scenarios in the problem instances, a behavior that was revealed even for the SSLP instances. On the contrary, solving the DEP with an increased number of scenarios results in a significant increase in the computation times.

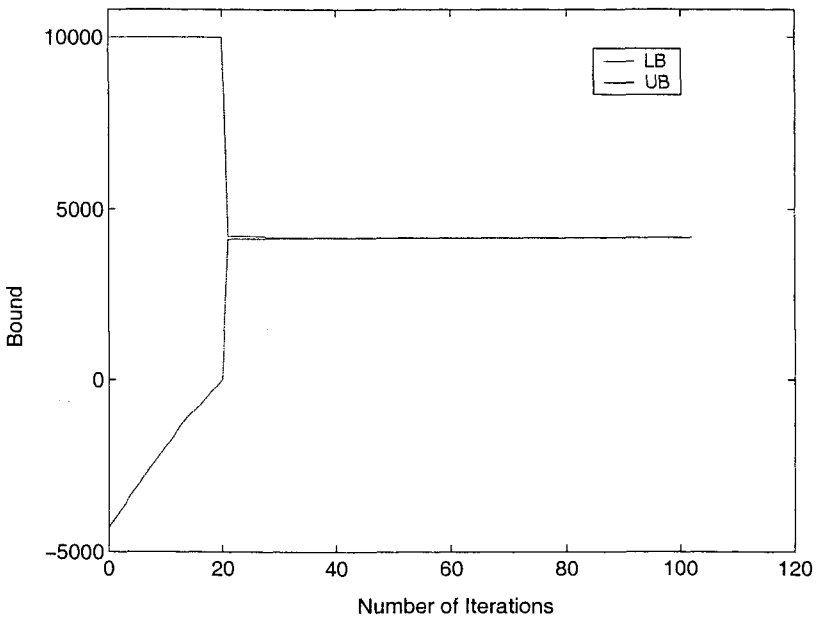

Figure 7.1: Convergence of the D^2 Algorithm for problem instance SM33

A typical graph of convergence of upper and lower bound is shown in Figure 7.1 for problem instance SM33. We note that the bounds have been translated so that they are nonnegative. As can be seen in the figure, the D^2 algorithm gets close to the optimal value (less than 1% optimality gap) just after about 20 iterations but fully closes the optimality gap and terminates after 102 iterations. The convergence of the algorithm is attainable for all the problem instances. Finally, we note that no D^2-cuts were generated in all the problem instances. This is due to the fact that scenario subproblem LP relaxations of the SM problem provided integral solutions due to the unimodularity of the recourse matrix and the integral second-stage right-hand side. Thus one can also apply a Benders' type algorithm to solve these instances. In this sense, this set of problems do not really test the power of D^2.

7.3.2 Strategic Supply Chain Planning Under Uncertainty

The stochastic SSCh test set consists of seven of the ten problem instances reported in Alonso-Ayuso et al. (2003), where they apply the BFC approach to the problem instances. The instances have the following dimensions: 6 plant/warehouses, 3 capacity levels per plant, 12 products, 8 subassemblies, 12 raw materials, 24 vendors, 2 markets per product, 10 time periods, and 23 scenarios. We refer the reader to the given reference for further details on the problem instances. For completeness, we restate the characteristics of the deterministic model problem instances in Table 7.3 as reported in Alonso-Ayuso et al. (2003). The columns of the table are as follows: "Constrs" is the number of constraints, "Bins" is the number of binary decision variables, "Cvars" is the number of continuous decision variables, and "Dens(%)" is constraint matrix density.

The dimensions for the first-stage and second-stage are given in Table 7.4. As shown in the table the SSCh model has a lot of continuous decision variables in

Table 7.3: Stochastic SSCh Deterministic Model Dimensions

Case	Constrs	Bins	Cvars	Dens(%)
c1	3,388	107	2,937	0.103
c2	3,458	108	3,068	0.100
c3	3,145	103	2,663	0.112
c4	3,405	105	3,065	0.099
c6	3,145	103	2,663	0.112
c8	3,894	114	3,634	0.087
c10	3,101	103	2,533	0.114

Table 7.4: Stochastic SSCh First and Second Stage Model Dimensions

	FIRST-STAGE		SECOND-STAGE		
Case	Constrs	Bins	Constrs	Bins	Cvars
c1	73	71	3315	36	2,937
c2	73	72	3385	36	3,068
c3	70	67	3075	36	2,663
c4	70	69	3335	36	3,065
c6	70	67	3075	36	2,663
c8	79	78	3815	36	3,634
c10	66	67	3035	36	2,533

the second-stage. The dimensions of the stochastic SSCh DEP model (7.4) for the 23 scenarios are given in Table 7.5. As can be seen in the table, the problem instances have thousands of constraints and continuous variables and hundreds of binary variables.

Continuous artificial variables were added to the instances with high penalty costs (10^{12}) in the objective function in order to induce relatively complete recourse as required by the D^2 approach. However, inducing relatively complete recourse for problem cases c5, c7 and c9 was not possible. Table 7.6 shows the main results of our computational experience. The table headings "Z_{IP} BFC" and "% Diff" give the best objective value as determined by the BFC method (Alonso-Ayuso et al., 2003) and the percentage difference between the best objective values determined by the D^2 algorithm and the BFC algorithm, respectively. The D^2 algorithm was terminated when the percent gap between the lower and upper bounds was below 5%

Table 7.5: Stochastic SSCh DEP Instance Dimensions

Case	Constrs	Bins	Cvars	Total Vars
c1	76,318	899	67,551	68,450
c2	77,928	900	70,564	71,464
c3	70,795	895	61,249	62,144
c4	76,775	897	70,495	71,392
c6	70,795	895	61,249	62,144
c8	87,824	906	83,582	84,488
c10	69,871	895	58,259	59,154

and the lower bound remained relatively constant for several consecutive iterations. This was done because there was no further improvement in the lower bound even after running the algorithm for additional iterations.

Table 7.6: Computational Results for Strategic SSCh Problem Instances

Case	D^2 Z_{IP}	Z_{IP} BFC	% Diff	D^2 Iters	D^2 Cuts	CPU	Gap
c1	184439.00	178366.79	3.29	184	177	4558.29	4.139%
c2	0.00	0.00*	0.00	68	57	1342.34	0.000%
c3	230268.10	224564.20	2.48	92	85	1179.48	4.461%
c4	201454.00	197487.36	1.97	160	149	3265.06	4.070%
c6	231368.93	226578.02	2.07	114	109	1642.74	4.650%
c8	100523.00	89607.39	10.86	186	180	9650.11	3.234%
c10	139738.36	139738.36*	0.00	87	81	1083.00	2.364%

*Optimality has been proven by (Alonso-Ayuso et al., 2003)

As can be seen in the table, the D^2 algorithm solves the problem instances to below 5% optimality gap. Note that it was futile to even attempt to solve the DEP instances using the CPLEX MIP solver due to the size of the instances. As Alonso-Ayuso et al. (2003) points out, the problem instances have large percent gaps between the LP relaxation and the integer solution values and coupled with the extremely high dimensions of the problem instances, it makes it unrealistic to pretend to prove solution optimality. Nonetheless, our algorithm obtains relatively improved solution values compared to the ones reported in Alonso-Ayuso et al. (2003), even up to 10% gain in the case of c8. For cases c2 and c10 our algorithm achieves optimality, which has been proven for these two cases by Alonso-Ayuso

et al. (2003). However, the algorithm could not close the optimality gap for c10 completely. Also note that the computation time for case c8 is very large, probably an indication of problem instance difficulty.

In essence, our computational results for the given problem instances confirm the computational experience of Alonso-Ayuso et al. (2003). Finally, let us mention that these authors have actually justified the use of the SP model for SSCh under uncertainty for the problem instances considered. They have shown that it is always beneficial to use the SP model instead of obtaining strategic decisions based on the average scenario parameters.

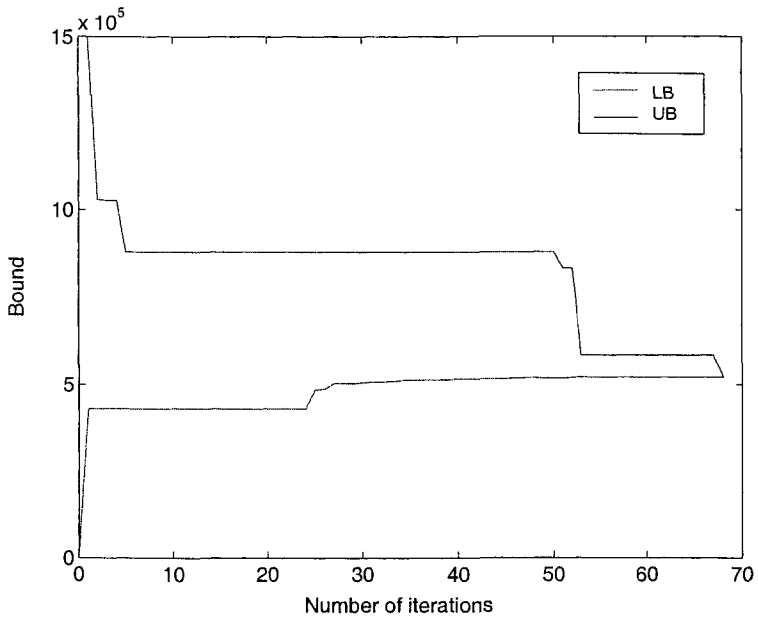

Figure 7.2: Convergence of the D^2 Algorithm for problem instance c2

Figures 7.2 and 7.3 show typical graphs of convergence of upper and lower bounds when applying the D^2 method to instances c1 and c2, respectively. Again the bounds have been translated so that they are nonnegative. As can be seen in Figure 7.2, the lower bound increases close to the optimal value in less than half the total number of iterations. However, good upper bounds are calculated

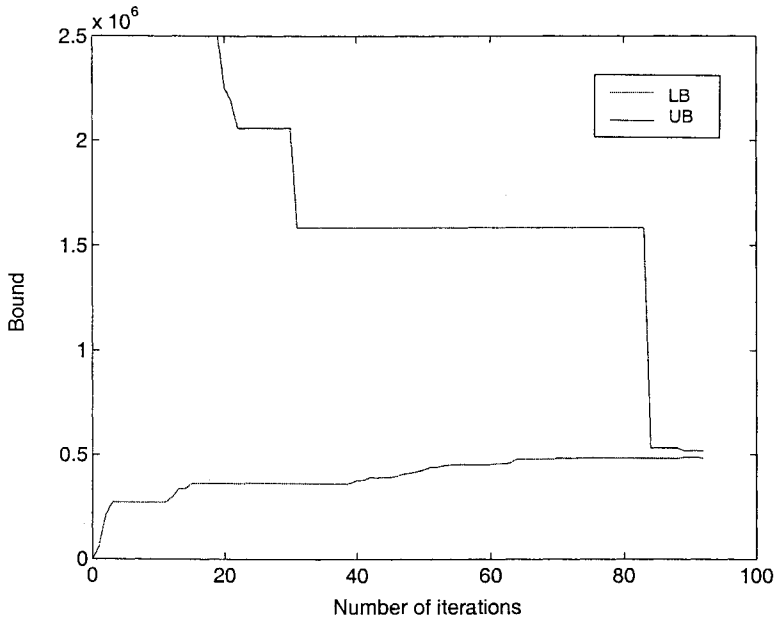

Figure 7.3: Convergence of the D^2 Algorithm for problem instance c3

only after first-stage solutions stabilize and this causes the method to continue for the remaining iterations without changing the lower bound significantly. Once no changes are detected in the first-stage solution, a good upper bound is calculated by solving the MIP subproblems. Figures 7.3 shows similar results. Even though the gap could not be fully closed for c1, the generally fast convergence of upper and lower bounds for the algorithm is attractive.

7.4 Summary

A computational study of the application of the D^2 method to stochastic matching and stochastic strategic supply chain planning has been presented. The study demonstrates the effective performance of this approach towards solving large-scale problem instances from these two application areas in which the second stage has continuous decision variables. Like the SSLPs, SM and SSCh problems result in large-scale problem instances which are comprised of loosely coupled subsystems, amenable to the divide-and-conquer paradigm of decomposition-coordination methods such as the D^2 method. The computational results show that the performance of the D^2 method is quite attractive.

CHAPTER 8

DISJUNCTIVE DECOMPOSITION FOR STOCHASTIC MIXED-INTEGER PROGRAMMING WITH CONTINUOUS FIRST-STAGE

The current D^2 method requires that the first-stage solutions be extreme points of the first-stage feasible set. This chapter extends the D^2 approach to SMIP problems where this requirement is not necessary. In particular, a branch-and-cut method for two-stage SMIP with continuous first-stage is derived and its convergence proved. Finally, a simple example to illustrate the proposed method is provided.

Throughout this chapter we consider the following two-stage SMIP problem:

$$\underset{x \in X}{\text{Min}}\, c^\top x + E[f(x, \tilde{\omega})], \tag{8.1}$$

where c is a known vector in $\Re^{n_1}$, $X \subseteq \Re^{n_1}$ is a set of feasible first-stage decisions, $E[.]$ is the usual mathematical expectation operator with $E[f(x,\tilde{\omega})] = \sum_{\omega \in \Omega} p_\omega f(x,\omega)$, $\tilde{\omega}$ is a multi-variate discrete random variable and for any scenario (realization) ω

$$f(x,\omega) = \text{Min } q^\top y, \tag{8.2a}$$

$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x, \tag{8.2b}$$

$$y \geq 0,\ y_j \text{ binary},\ j \in J_2. \tag{8.2c}$$

In problem formulation (8.2), q is the cost vector in $\Re^{n_2}$ and J_2 is an index set consisting of some or all of the recourse decision variables $y \in \Re^{n_2}$. We consider the SMIP case in which the first-stage solutions are continuous or mixed-binary and

are therefore not restricted to be extreme points of the set X. We maintain that assumptions A1-A3 (Chapter 2, Section 1.2) hold and that all integer variables in the second-stage are binary. Hence, all second-stage scenario subproblems satisfy the facial disjunctive property and thus, sequential convexification is possible for any scenario subproblem. In addition, we require the following assumption to hold:

(A4) The LP relaxation of (8.2) has a positive objective value for all (x, ω).

Since X is assumed to be compact, it is not too restrictive to assume that a uniform lower bound on the second-stage can be easily calculated. Hence without loss of generality, the value function can be appropriately translated to satisfy the assumption.

Before we present the new method, let us first consider one complication that is brought by the requirement that the first-stage solution x be continuous instead of an extreme point of X as is the case in the current D^2 and D^2-BAC algorithms. Let X_c be the following set:

$$X_c = \{x_1, x_2 \mid -x_1 \geq -2, \ -x_2 \geq -2, \ x_1, x_2 \geq 0\}$$

and suppose that the first-stage solutions be extreme points of this set $vert(X_c)$. Then this set has only four extreme points $x = [x_1, x_2]^\top$: $x = [0,0]^\top$, $x = [0,2]^\top$, $x = [2,0]^\top$ and $x = [2,2]^\top$. Therefore, in the context of the D^2 algorithm only this finite set of extreme points would be generated in the first-stage and forwarded to the second-stage in the worst-case. Also, the optimal solution of the SMIP must be at least one of the four points.

Consider a case in which the first-stage is continuous and the set X_c is as defined earlier. Now instead of having only the four extreme points, we have all the points $\{0 \leq x_1 \leq 2\} \times \{0 \leq x_2 \leq 2\}$ to consider (potentially an infinite number of points). Since we develop convexifications of the second-stage using inputs from the first-stage, the introduction of continuous variables in a decomposition method makes it

more challenging.

The next section provides the theoretical bridge between the current D^2 method and the one we propose. Section 8.2 derives the basic theory for the new approach and describes the algorithmic setting. Section 8.3 presents the new branch-and-cut algorithm and Section 8.4 gives a detailed example illustration of the new approach. Finally, Section 8.5 gives some extension perspectives.

8.1 Background: Bridging the Gap

In the D^2 algorithm sequential convexification is achieved via the C^3 theorem (Sen and Higle, 2000), which allows for a D^2 cut generated for one scenario subproblem to be easily translated into a cut that is valid for another scenario subproblem. The D^2 cut has the form $\pi^\top y \geq \pi_c(x,\omega)$, where π is the common-cut-coefficient vector and the right-hand side is a convexification of the function $\pi_0(x,\omega)$. Both π and $\pi_0(x,\omega)$ are generated by forming and solving the C^3-SLP (Problem 3.10, Ch. 3). The solution of this SLP provides the π as well as the coefficient vector multipliers that define $\pi_0(x,\omega)$ for each $\omega \in \Omega$. The convexification of $\pi_0(x,\omega)$ to $\pi_c(x,\omega)$ is achieved by solving the RHS-LP (Problem 3.11, Ch. 3) for each $\omega \in \Omega$ derived via a strategy from reverse convex programming in which disjunctive programming is used to provide facets of the convex hull of reverse convex sets (Sen and Sherali, 1987). Consequently, the LP relaxation of (8.2) in the K-th algorithmic iteration of the D^2 algorithm has the form

$$f_c^K(x,\omega) = \text{Min } q^\top y, \tag{8.3a}$$

$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x, \tag{8.3b}$$

$$(\pi^k)^\top y \geq \pi_c^k(x,\omega),\ k \in \Theta_K, \tag{8.3c}$$

$$y \geq 0, \tag{8.3d}$$

for all $\omega \in \Omega$. The set Θ_K is the set of algorithmic indices k at which a D^2 cut is generated. The scenario subproblem LP (8.3) can be rewritten in compact form as follows:

$$f_c^K(x,\omega) = \text{Min } q^\top y, \tag{8.4a}$$

$$\text{s.t. } W^K y \geq \rho^K(x,\omega), \tag{8.4b}$$

$$y \geq 0. \tag{8.4c}$$

where, the recourse matrix W augmented by $\{(\pi^k)^\top\}_{k\in\Theta_K}$ is denoted by W^K. Similarly, the right-hand side vector $r(\omega) - T(\omega)x$ augmented by $\{\pi_c^k(x,\omega)\}_{k\in\Theta_K}$, is denoted by $\rho^K(x,\omega)$.

Figure 8.1 shows a two-dimensional graphical illustration of the epigraph of functions $\pi_0(x,\varpi)$ and $\pi_c(x,\varpi)$ for a fixed scenario ϖ. In the figure L and U are the lower and upper bounds, respectively, on the decision variable x and $X = \{L \leq x \leq U\}$. The function $\pi_0(x,\varpi)$ is a piecewise linear concave function of the first argument. Note that for an extreme point $x^e \in X$, $\pi_0(x^e,\varpi) = \pi_c(x^e,\varpi)$.

We shall first state the theorem that addresses the identification of an optimal solution to the SMIP problem under the assumption that the master program solutions are extreme points of the first-stage feasible set X, for example SMIP problems with purely binary first-stage decision variables. The theorem provides a "bridging point" towards the derivation of a convexification process for the continuous first-stage case.

Theorem 1. *Suppose that $X = \{x \in \Re_+^{n_1} \mid Ax \geq b\}$ and assumptions A1-A4 hold. Suppose the D^2 algorithm identifies extreme point solutions of the C^3-SLP (Problem 3.10, Ch. 3). Then there exists a $\bar{K} < \infty$ such that for all $k > \bar{K}$, $f_c^k(x^k,\omega) = f(x^k,\omega)$ for all $\omega \in \Omega$ whenever x^k is an extreme point of X.*

Theorem 1 shows that there are finitely many second-stage scenario subproblem

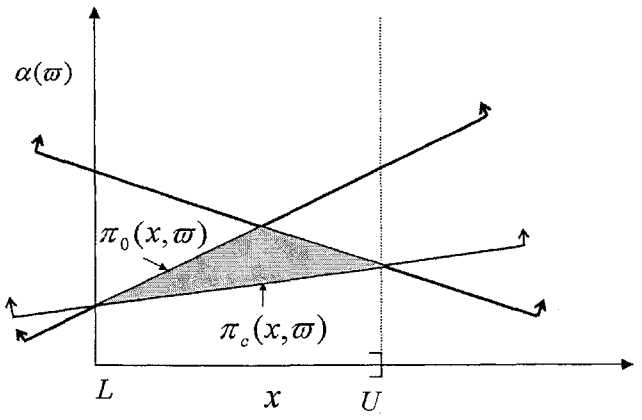

Figure 8.1: Graphical illustration of the functions $\pi_0(x,\varpi)$ and $\pi_c(x,\varpi)$

polyhedra of the form:

$$\pi_j \geq \lambda_{0,1}^\top W_j^k - I_j^k \lambda_{0,2}, \ \forall j \tag{8.5a}$$

$$\pi_j \geq \lambda_{1,1}^\top W_j^k + I_j^k \lambda_{1,2}, \ \forall j \tag{8.5b}$$

$$-1 \leq \pi_j \leq 1, \forall j \tag{8.5c}$$

$$\lambda_{0,1}, \lambda_{0,2}, \lambda_{1,1}, \lambda_{1,2} \geq 0 \tag{8.5d}$$

Furthermore, these polyhedra can have only finitely many extreme points. So let $\{(\lambda_0^e, \lambda_1^e)\}_e^M$ denote the collection of such extreme points from all the possible polyhedra, where $\lambda_0^e = [\lambda_{0,1}^e; \lambda_{0,2}^e]^\top$, and $\lambda_1^e = [\lambda_{1,1}^e; \lambda_{1,2}^e]^\top$. Then for any pair $(\lambda_0^e, \lambda_1^e)$ we can associate the pair $(\bar{\nu}_0^e(\omega), \bar{\gamma}_0^e(\omega))$ and $(\bar{\nu}_1^e(\omega), \bar{\gamma}_1^e(\omega))$ for all $\omega \in \Omega$ as suggested in Corollary 4 of Sen and Higle (2000). Therefore, for all $e \in \{1, ..., M\}$ we have

$$\pi_0^e(x,\omega) = \text{Min}\{\bar{\nu}_0^e(\omega) - \bar{\gamma}_0^e(\omega)^\top x, \ \bar{\nu}_1^e(\omega) - \bar{\gamma}_1^e(\omega)^\top x\}.$$

Using the sequential convexification property for facial disjunctive programs, it

follows that one can recover the closure of the convex hull of the second-stage mixed-binary points by appending the constraints $(\pi^e)^\top y \geq \pi_0^e(x,\omega)$ for all $e \in \{1,...,M\}$. Consequently the second-stage subproblem LP relaxation takes the following form:

$$f_0(x,\omega) = \text{Min } q^\top y, \tag{8.6a}$$

$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x, \tag{8.6b}$$

$$(\pi^e)^\top y \geq \pi_0^e(x,\omega),\ e = 1,...,M, \tag{8.6c}$$

$$y \in \Re_+^{n_2}. \tag{8.6d}$$

In general the number of extreme points of (8.5) given by M can be very large. Hence Sen and Higle (2000) derive the D^2 algorithm based on limited and sequentially revealed information.

In order to convexify $\pi_0^e(x,\omega)$ Sen and Higle (2000) use disjunctive programming to provide facets of the convex hull of reverse convex sets and define the epi-reverse polar denoted $\Pi_X^\dagger(\omega)$ whose extreme points can be denoted by $\{(\sigma_0^\tau(\omega), \sigma^\tau(\omega), \delta^\tau(\omega))\}_{\tau\in\mathcal{T}}$. Note that this defines the feasible set for the *RHS*-LP (Problem 3.11, Ch. 3). Letting $\nu_\tau(\omega) = \frac{\delta(\omega)}{\sigma_0(\omega)}$ and $\gamma_\tau(\omega) = \frac{\sigma^\top(\omega)}{\sigma_0(\omega)}$, they define $\pi_c : X \times \Omega \to \Re$, where

$$\pi_c(x,\omega) = \underset{\tau\in\mathcal{T}}{\text{Max}}\{\nu_\tau(\omega) - \gamma_\tau(\omega)^\top x\}. \tag{8.7}$$

The function $\pi_c(x,\omega)$ is referred to as the convex hull approximation of $\pi_0(x,\omega)$. This is because $\{(\alpha,x) \mid x \in X, \alpha \geq \pi_c(x,\omega)\}$, the epigraph of $\pi_c(x,\omega)$ restricted to $x \in X$, agrees with clconv($\Pi_X(\omega)$). Thus $\pi_0(x,\omega) = \pi_c(x,\omega)$ whenever x is an extreme point of X. This property always holds for example, for purely 0-1 first-stage, but may not hold for continuous first-stage in general. The *violation* of this property will guide us towards the derivation of a branch-and-cut algorithm for the continuous first-stage case.

8.2 Algorithmic Approach

Let us now consider an iteration K of the D^2 algorithm applied to problem (8.1-8.2). Then the algorithm has iteratively identified $\{\pi^k, \pi_0^k(x,\omega), \pi_c^k(x,\omega)\}_{k\in\Theta_K}$ and the master program has the following form:

$$\text{Min } c^\top x + \eta \tag{8.8a}$$

$$\text{s.t. } Ax \geq b, \tag{8.8b}$$

$$\beta_k{}^\top x + \eta \geq \alpha_k, \ k = 1, ..., K, \tag{8.8c}$$

$$x \geq 0, \eta \geq 0. \tag{8.8d}$$

The variable η provides a piecewise linear approximation of the subproblem expected objective function $E[f(x,\tilde{\omega})]$. Constraints (8.8c) are the Benders-type (Benders, 1962) optimality cuts, and constraints (8.8d) are the nonnegativity requirements on x and η. In constraint (8.8c) the right-hand side $\alpha_k = E[\psi^k(\tilde{\omega})^\top r^k(\tilde{\omega})]$ and $\beta_k = E[\psi^k(\tilde{\omega})^\top T^k(\tilde{\omega})]$ for $k = 1, ..., K$, where $\psi^k(\omega)$ denotes an appropriately dimensioned vector of optimal dual multipliers associated with constraints (8.4b) for scenario $\omega \in \Omega$ in iteration k.

For each algorithmic iteration $k \in \Theta_K$ we have a pair $\{\pi_0^k(x,\omega), \pi_c^k(x,\omega)\}$, where

$$\pi_0^k(x,\omega) = \text{Min}\{\bar{\nu}_0^k(\omega) - \bar{\gamma}_0^k(\omega)^\top x, \ \bar{\nu}_1^k(\omega) - \bar{\gamma}_1^k(\omega)^\top x\}$$

and from the optimal solution of the RHS-LP (Problem 3.11, Ch. 3) we get

$$\pi_c^k(x,\omega) = \nu^k(\omega) - \gamma^k(\omega)^\top x.$$

Since x is continuous it follows that x is not necessarily an extreme point of X and therefore, the condition $\pi_0(x,\omega) = \pi_c(x,\omega)$ may not hold in general even when the solution of (8.4) satisfies the binary restrictions. Consequently, the D^2 algorithm may not converge to an optimal solution of problem (8.1-8.2) in general. Therefore, our approach will involve recording the pairs $\{\pi_0^k(x,\omega), \pi_c^k(x,\omega)\}$ during

the execution of the D^2 algorithm and then using these data to sequentially identify those pairs that violate the condition $\pi_0^k(x,\omega) = \pi_c^k(x,\omega)$ for a given ω and k. The goal will be to create a branch-and-bound (B&B) tree by partitioning the first-stage continuous feasible set X based on the violated pairs.

A major issue in applying a B&B algorithm over continuous domains is that the resulting approach may not be finite. Our approach will however be guided by *disjunction variables* in the second-stage, whose total number is finite. Consequently, we will have finitely many subdivisions to consider, leading to the finiteness of our B&B method.

8.2.1 Foundations of the Branch-and-Bound Approach

Let some first-stage solution $\bar{x}$ be given. Then we can identify a scenario $\varpi \in \Omega$ and iteration index κ that violates the condition $\pi_0^k(\bar{x},\omega) = \pi_c^k(\bar{x},\omega)$. The idea is to find the pair (ϖ,κ) for which the condition is both violated the most and scenario ϖ has a high probability of occurrence p_ω. This can be accomplished by the following criterion:

$$(\varpi,\kappa) \in \underset{\omega\in\Omega,\ k\in\Theta_K}{\operatorname{argmax}} \left\{ p_\omega \left(\pi_0^k(\bar{x},\omega) - \pi_c^k(\bar{x},\omega) \right) \right\} \tag{8.9}$$

Proposition 2. *Let* $\bar{\gamma}_{q0}^{\kappa}(\varpi) = \bar{\gamma}_1^{\kappa}(\varpi) - \bar{\gamma}_0^{\kappa}(\varpi)$, $\bar{\nu}_{q0}^{\kappa}(\varpi) = \bar{\nu}_1^{\kappa}(\varpi) - \bar{\nu}_0^{\kappa}(\varpi)$, *and* $\bar{\gamma}_{q1}^{\kappa}(\varpi) = -\bar{\gamma}_{q0}^{\kappa}(\varpi)$ *and* $\bar{\nu}_{q1}^{\kappa}(\varpi) = -\bar{\nu}_{q0}^{\kappa}(\varpi)$. *Also let* $X_{q0} = \{x \mid \bar{\gamma}_{q0}^{\kappa}(\varpi)^\top x \geq \bar{\nu}_{q0}^{\kappa}(\varpi), x \geq 0\}$ *and* $X_{q1} = \{x \mid \bar{\gamma}_{q1}^{\kappa}(\varpi)^\top x \geq \bar{\nu}_{q1}^{\kappa}(\varpi), x \geq 0\}$. *Suppose that* X^q *denotes some subset of the first stage decisions* X. *Then* X^q, *can be given as* $X^q = \mathcal{P}_{q0} \cup \mathcal{P}_{q1}$, *where*

$$\mathcal{P}_{q0} = X_q \cap X_{q0} \tag{8.10}$$

and

$$\mathcal{P}_{q1} = X_q \cap X_{q1} \tag{8.11}$$

Proof. Condition (8.9) identifies a pair (ϖ, κ) such that $\pi_0^\kappa(\bar{x}, \varpi) > \pi_c^\kappa(\bar{x}, \varpi)$. Since

$$\pi_0^\kappa(x, \varpi) = \text{Min}\{\bar{\nu}_0^\kappa(\varpi) - \bar{\gamma}_0^\kappa(\varpi)^\top x,\ \bar{\nu}_1^\kappa(\varpi) - \bar{\gamma}_1^\kappa(\varpi)^\top x\} \tag{8.12}$$

is a piece-wise linear concave function, it follows that either

$$\bar{\nu}_0^\kappa(\varpi) - \bar{\gamma}_0^\kappa(\varpi)^\top x \geq \bar{\nu}_1^\kappa(\varpi) - \bar{\gamma}_1^\kappa(\varpi)^\top x \tag{8.13}$$

or

$$\bar{\nu}_0^\kappa(\varpi) - \bar{\gamma}_0^\kappa(\varpi)^\top x \leq \bar{\nu}_1^\kappa(\varpi) - \bar{\gamma}_1^\kappa(\varpi)^\top x. \tag{8.14}$$

By intersecting the half-space defined by (8.13) with X_q and that defined by (8.14) with X_q the result follows. ■

Figure 8.2 shows a two-dimensional graphical illustration of the branching constraints and the epigraph of the functions $\pi_0(x, \varpi)$ and $\pi_c(x, \varpi)$ for a fixed scenario ϖ. In the figure L and U are the lower and upper bounds, respectively, on the decision variable x and $X = \{L \leq x \leq U\}$. The two sets $\mathcal{P}_{q0}$ and $\mathcal{P}_{q1}$ are shown in the figure. Note that the branching constraints pass through the intersection of the two affine/linear pieces comprising the function $\pi_0(x, \varpi)$.

Proposition 2 allows us to divide the first-stage feasible set into two sets and therefore, optimization of the original problem can be carried out over each set. This then enables us to devise a B&B procedure for solving problem (8.1-8.2), whereby we can further divide each resulting set if necessary following the same logic.

Let Q denote the set of nodes of the B&B tree for the first-stage and let $q \in Q$ denote a node of the B&B tree. At the root node of the tree we have the initial SMIP problem (8.15 - 8.16). We can then apply the D^2 algorithm to this problem and after each iteration k we check the condition $\pi_0^k(x^k, \omega) = \pi_c^k(x^k, \omega)$. If it is violated we perform "branching" on the first-stage by invoking Proposition 2 and

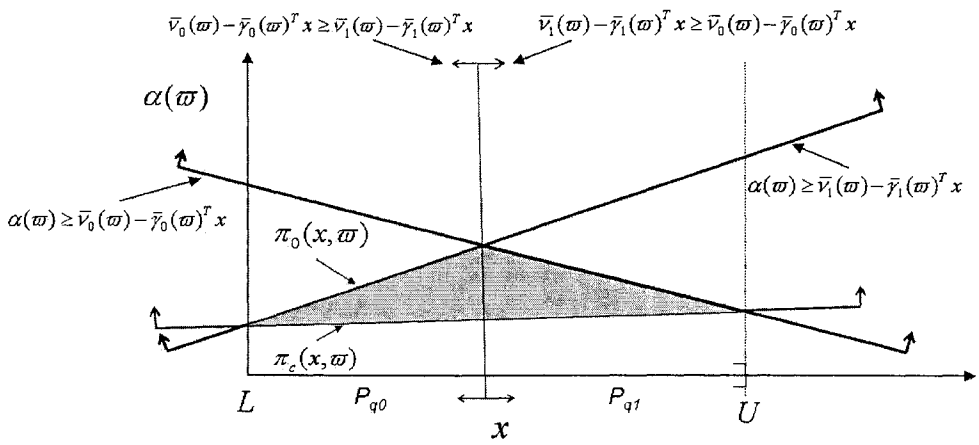

Figure 8.2: Graphical illustration of the branching constraints

generating "branching" constraints (8.13) and (8.14). By branching at node q we create two *sibling nodes* q_0 and q_1 from the *parent node* q, and for each sibling node we have the parent master program with the corresponding branching constraint added, resulting in the set P_{q_h} of the first-stage feasible set, where , $h \in H$ and $H = \{0, 1\}$. Then the D^2 algorithm can be applied to the problem at each *sibling node* and the process repeated.

Let k_q denote the algorithmic iteration for the problem at a node $q \in Q$. With q we can associate a sequence of nodes that trace a unique path from the node q to

the root node. Let B_q denote this sequence of nodes. For each element $\tau \in B_q$, we have all algorithmic indices $k(\tau)$ as well as indices $\kappa(\tau)$ that identify the iteration κ and scenario ϖ used in the branching process. Thus for each $k \in \kappa(\tau)$ we have a pair (ϖ, κ) with the associated branching constraint coefficients $\bar{\gamma}^k_{q_h} = \bar{\gamma}^\kappa_{q_h}(\varpi)$ and $\bar{\nu}^k_{q_h} = \bar{\nu}^\kappa_{q_h}(\varpi)$ for $h \in H$. Then the master program (8.8) at a node $q \in Q$ takes the following form:

$$\text{Min } c^\top x + \eta \tag{8.15a}$$

$$\text{s.t. } Ax \geq b, \tag{8.15b}$$

$$\beta_k{}^\top x + \eta \geq \alpha_k, \ k \in k(\tau), \ \tau \in B_q \tag{8.15c}$$

$$(\bar{\gamma}^k_{q_h})^\top x \geq \bar{\nu}^k_{q_h}, \ h \in H, \ k \in \kappa(\tau), \ \tau \in B_q, \tag{8.15d}$$

$$x \geq 0, \eta \geq 0. \tag{8.15e}$$

Constraints (8.15c) are the Benders's type optimality cuts derived up to iteration k_q for node q. Constraints (8.15d) are the branching constraints added to the master program for node q. The nodal master program (8.15) may become infeasible for some node $q \in Q$ beyond the first branch in the B&B tree. In this case, the node is fathomed, and we backtrack.

Let us now turn to updating the right-hand side for the D^2 cut on which branching is performed. Since branching is carried out based on this D^2 cut, its right-hand side $\pi^\kappa_c(x, \varpi)$ must be updated to reflect the branching.

Corollary 3. *Let (ϖ, κ) be defined as in (8.9). Consider the D^2 cut at which (ϖ, κ) is determined. Then the RHS $\pi^\kappa_c(x, \varpi)$ for the D^2 cut must be replaced by $\pi^\kappa_0(x, \omega) = \bar{\nu}^\kappa_1(\varpi) - \bar{\gamma}^\kappa_1(\varpi)^\top x$ for the branch $\bar{\gamma}^\kappa_{q_0}(\varpi)^\top x \geq \bar{\nu}^\kappa_{q_0}(\varpi)$, and by $\pi^\kappa_0(x, \omega) = \bar{\nu}^\kappa_0(\varpi) - \bar{\gamma}^\kappa_0(\varpi)^\top x$ for the branch $\bar{\gamma}^\kappa_{q_1}(\varpi)^\top x \geq \bar{\nu}^\kappa_{q_1}(\varpi)$.*

Proof. From Proposition 2, it follows that for the branch $\bar{\gamma}^\kappa_{q_0}(\varpi)^\top x \geq \bar{\nu}^\kappa_{q_0}(\varpi)$, the right-hand side of the cut must be the minimum of the two affine functions. Hence

$\pi_0^\kappa(x,\omega) = \bar{\nu}_1^\kappa(\varpi) - \bar{\gamma}_1^\kappa(\varpi)^\top x$. Similarly, the right-hand side of the cut for the other branch is also justified. ■

The master program (8.15) in iteration k_q provides a first-stage solution x^{k_q} to feed to the second-stage LP for $\omega \in \Omega$ and node $q \in Q$, which takes the form:

$$f_c^{k_q}(x^{k_q},\omega) = \text{Min } q^\top y, \tag{8.16a}$$

$$\text{s.t. } Wy \geq r(\omega) - T(\omega)x^{k_q}, \tag{8.16b}$$

$$(\pi^k)^\top y \geq \pi_c^k(x^{k_q},\omega),\ k \in k(\tau) \setminus \kappa(\tau),\ \tau \in B_q \tag{8.16c}$$

$$(\pi^k)^\top y \geq \pi_0^k(x^{k_q},\omega),\ k \in \kappa(\tau),\ \tau \in B_q \tag{8.16d}$$

$$y \geq 0. \tag{8.16e}$$

Constraints (8.16c) are the D^2 cuts generated at each node along the path from the root node to node q whose RHS has not been updated, while constraints (8.16d) have the RHS updated (due to branching). From here on, we shall refer to problem (8.15-8.16) as the *nodal problem* P_q for node q. We shall denote by v_q and V_q the lower and upper bounds of the nodal problem as determined by the D^2 algorithm.

We now note that the D^2 cuts generated at a node $q \in Q$ are valid for all the descendant nodes of q. However, they are generally not valid for all the other nodes in Q due to the fact that the right-hand side $\rho_c^k(x,\omega) = r^k(\omega) - T^k(\omega)x$ may be different for scenario ϖ and iteration index κ due to the update performed after branching according to Proposition 2.

8.3 A Branch-and-Cut Algorithm

We shall now present a formal statement of the basic D^2 with branch-and-cut algorithm for problem (8.1-8.2). We shall refer to the algorithm as the D^2-CBAC algorithm, since it is based on the basic D^2 algorithm and "C" stands for "continuous" and "BAC" for "branch-and-cut". The critical operations in the

algorithm are italicized and are discussed in the following subsection. We use the following notation in the description.

Notation:

$\mathcal{L}$: List of open nodal problems

v: Lower bound on the optimum

V: Upper bound on the optimum

v_q: Lower bound on node q problem optimum

V_q: Upper bound on node q problem optimum

k: Iteration index for root node problem

k_q: Iteration index for node q problem

x^*: Optimal solution

The Basic D^2-CBAC Algorithm

Step 0. Initialization:

Let $\epsilon > 0$ and $x^1 \in X$ be given. Set $k \leftarrow 0$ and initialize $V = \infty$, $v = -\infty$, and add the root nodal problem (8.15-8.16) to the list $\mathcal{L}$ of open nodal problems.

Step 1: Termination

If $\mathcal{L} = \emptyset$, terminate with solution x^*.
Otherwise *select* a nodal problem P_q from the list $\mathcal{L}$ of currently open nodal problems, and set $\mathcal{L} \leftarrow \mathcal{L} \setminus P_q$ and $k \leftarrow k + 1$.

Step 2. Apply the D^2 Algorithm to the Nodal Problem

Apply one iteration of the D^2 algorithm to the nodal problem P_q.
Store the multipliers $\{\lambda^k_{0,1}; \lambda^k_{0,2}\}$, and $\{\lambda^k_{1,1}; \lambda^k_{1,2}\}$.
These multipliers define $(\bar{v}^k_0(\omega), \bar{\gamma}^k_0(\omega))$, $(\bar{v}^k_1(\omega), \bar{\gamma}^k_1(\omega))$
and the corresponding $\pi^k_0(x,\omega)$ and $\pi^k_c(x,\omega)$ for all k and all $\omega \in \Omega$.
At the end of the D^2 algorithm iteration one of the following
must hold:
(i) Nodal master program (8.15) becomes infeasible;
(ii) $V_{k_q} - v_{k_q} < \epsilon$;
(iii) $V_{k_q} - v_{k_q} \geq \epsilon$.

If condition (i) is true

Fathom this node by infeasibility. Go to Step 1,

else if condition (ii) is true, update incumbent:

If $V_{k_q} < V$, then $V \leftarrow V_{k_q}$, $v \leftarrow v_{k_q}$ and $x^* \leftarrow x_{k_q}$.

Fathom this node by optimality.

Fathom the node list, that is, $\mathcal{L} \leftarrow \mathcal{L} \setminus \{P_q \mid v_q \geq V\}$.

Otherwise fathom this node by bound.

Go to Step 1,

else if condition (iii) is true,

go to Step 3.

Step 3: Branching

Apply Proposition 2 and perform *branching* to create two nodal
problems P_{q_0} and P_{q_1} of the form (8.15 -8.16) and add to the
list $\mathcal{L}$ of open nodal problems, that is, set $\mathcal{L} \leftarrow \mathcal{L} \cup \{P_{q_0}, P_{q_1}\}$.
For node *selection* purposes determine and record v_{q_0} and v_{q_1},
the objective value of the corresponding
nodal master program (8.15) after branching.
Go to Step 1.

8.3.1 Algorithm Convergence

We now prove finite convergence of the D^2-CBAC algorithm.

Lemma 4. *There exists a finite number t such that after t divisions of the first-stage feasible set X, the branch-and-bound process described in Section 8.3 finds a node that can be fathomed.*

Proof. A node is fathomed if we either encounter $v_q = V_q$ or an infeasible nodal master program. Let $Y(x) = \{y \mid Wy \geq r(\omega) - T(\omega)x, y \geq 0, -y_j \geq -1, j \in J_2\}$ and let I_q denote the set of indices defining the disjunction variables for B_q. Because we have the right-hand side equal to $\pi_0^k(x,\omega)$ for all $k \in \kappa(\tau), \tau \in B_q$, the cuts used in the second-stage are facets of $Y(x) \cap (\{y_j \leq 0\} \cup \{y_j \geq 1\})_{j \in I_q}$. Since the feasible set of the second-stage can be described as a facial disjunctive set, the sequential convexification process yields the convex hull of the set as proved by Balas (1979). Therefore, after finitely many divisions (t), the branch-and-bound process finds a node with either $v_q = V_q$ or an infeasible nodal master program 8.15. ■

Theorem 5. *Suppose that assumptions (A1-A4) hold. The proposed D^2-CBAC algorithm terminates with an optimal solution to problem (8.1-8.2) after finitely many steps.*

Proof. In the proposed D^2-CBAC algorithm branching can only be done finitely many times. This follows from the fact that there are finitely many disjunction variables in the second-stage, leading to finitely many right-hand sides $\{\pi_0^k(x,\omega)\}_{k \in \Theta_K}$ for some $K < \infty$ for all $\omega \in \Omega$. Consequently, there are finitely many divisions of the continuous first-stage feasible set to consider. Then by Lemma 4 each node q will be fathomed in the course of the algorithm. The optimality of the

solution follows from the validity of the lower and upper bounding procedures used. ■

Comments on the critical operations of the D^2-CBAC algorithm are now in order. The *branching* part of the algorithm requires that the multipliers that define $\pi_0^k(x,\omega)$ and $\pi_c^k(x,\omega)$ for all k and $\omega \in \Omega$ be recorded. However, this data may be large and therefore, we suggest writing them to a file if necessary for computer memory considerations.

In Step 1 of the algorithm we need to *select* a nodal problem from the list $\mathcal{L}$. We propose selecting a nodal problem with the least lower bound v_q. That is, selecting a problem $P_{\bar{q}}$ such that $v_{\bar{q}} = \min_{q\in\mathcal{L}}\{v_q\}$. This follows the *best-node first strategy* (see e.g (Wolsey, 1998)) and leads to a bound improving selection operation as required for the convergence of a B&B algorithm (Horst and Tuy, 1996).

Since the D^2 cuts generated at a given node are not in general valid for all nodes in the B&B tree, care must be taken to curtail the proliferation of these cuts. We suggest avoiding storing all the cuts explicitly for each node. Instead, the cuts can be stored in some data structure that contains each cut with the corresponding algorithmic iteration and node number at which the cut was generated. Thus each cut would be recorded only once and used for each nodal problem as required. When a node is *fathomed* (by infeasibility, bound or optimality) all the D^2 cuts that were generated at that node can be deleted from the list. This also applies to the branching constraints and optimality cuts generated at a node and added to the master program. Next we provide an example to illustrate the proposed algorithm.

8.4 Example Illustration

Consider the following simple SMIP problem with two scenarios:

$$\begin{aligned} \text{Min} \quad & -2x + E[f(x,\omega)] \\ \text{s.t.} \quad & -x \geq -2 \\ & x \geq 0 \end{aligned}$$

where,

$$\begin{aligned} f(x,\omega) = \text{Min} \quad & -4y \\ \text{s.t.} \quad & -6y \geq r(\omega) + 5x \\ & y \text{ binary.} \end{aligned}$$

The first-stage variable is x, while the second-stage variable is y. There are two scenarios, ω_1 with $r(\omega_1) = -10$, and ω_2 with $r(\omega_2) = -12$. The scenarios have equal probability of occurrence, that is, $p_{\omega_1} = p_{\omega_2} = 0.5$. Note that the problem satisfies assumptions A1-A3. In this problem we have the following data: $c = -2$, $A = [-1]$, $b = [-2]$, $q = [-4]$, $W = [-6]$, $T(\omega_1) = T(\omega_1) = [-5]$, $r(\omega_1) = -10$ and $r(\omega_2) = -12$. Note that the first-stage solutions are not necessarily extreme points of $X = \{x : Ax \geq b, x \geq 0\}$. We can now start the D^2-CBAC algorithm as follows.

Iteration 1 ($k = 1$)

Step 0: Initialization

The algorithm is initialized with the following root node (Node 0) problem P_0 whose

master program is:

$$\begin{aligned} \text{Min } & -2x + \eta \\ \text{s.t. } & -x \geq -2 \\ & x, \eta \geq 0. \end{aligned}$$

Set $W^1 = W$, $T^1(\omega) = T(\omega)$, and $r^1(\omega) = T(\omega)$ for both scenarios and initialize the list open nodal problems $\mathcal{L} \leftarrow \mathcal{L} \cup \{P_0\}$. The overall upper and lower bounds are initialized as $V = \infty$ and $v = -4$, respectively.

Step 1: Termination

List $\mathcal{L} \neq \emptyset$, so we select the initial nodal problem. The initial master program yields $x^1 = 2$ and $\eta = 0$. The nodal upper and lower bounds are initialized as $V_0 = \infty$ and $v_0 = -4$, respectively.

Step 2: Apply one iteration of the D^2 Algorithm to the Nodal Problem

Step (i)

For Step 1 of the algorithm we use $x^1 = 2$ and solve the LP relaxation of the second-stage subproblem for ω_1 and ω_2, which we shall call LP_1 and LP_2, respectively:

$$\begin{aligned} LP_1 : f_1(2, \omega_1) = \text{Min } & -4y \\ \text{s.t. } & -6y \geq 0 \\ & -y \geq -1 \\ & y \geq 0. \end{aligned}$$

and

$$LP_1 : f_2(2,\omega_2) = \text{Min } -4y$$
$$\text{s.t. } -6y \geq -2$$
$$-y \geq -1$$
$$y \geq 0.$$

The optimal solution for LP_1 is $y(\omega_1) = 0$ and for LP_2 is $y(\omega_2) = 0.3333$

Step (ii)

Since $y(\omega_2)$ does not satisfy the integer restrictions, we choose y as the "disjunction" variable and create the disjunction $-y \geq 0$ or $y \geq 1$ for LP_2. We formulate the C^3-LP (Problem 3.10, Ch. 3), which yields the vector π^1 for updating W^1 and the data for the RHS-LP (Problem 3.11, Ch. 3) whose optimal solution is used to update the right-hand side of the second-stage constraints. An optimal solution for the C^3-LP is $\pi^1 = -1, \lambda_{0,1}^\top = [0,0], \lambda_{0,2} = 1, \lambda_{1,1}^\top = [0.25,0]$, and $\lambda_{1,2} = 0.5$.

We obtain W^2 by appending π^1 to W^1: $W^2 = \begin{bmatrix} [W^1] \\ -1 \end{bmatrix}$. From the C^3-LP solution we obtain $\bar{\nu}_0^1(\omega_1) = 0$, $\bar{\nu}_1^1(\omega_1) = -2$, $\bar{\gamma}_0^1(\omega_1)^\top = [0]$, $\bar{\gamma}_1^1(\omega_1)^\top = [-1.25]$ and $\bar{\nu}_0^1(\omega_2) = 0$, $\bar{\nu}_1^1(\omega_2) = -2.5$, $\bar{\gamma}_0^1(\omega_2)^\top = [0]$, $\bar{\gamma}_1^1(\omega_2)^\top = [-1.25]$. In order to maintain $\pi_0^k(x,\omega) \geq 0$, we translate $\bar{\nu}_0^1(\omega)$ and $\bar{\nu}_1^1(\omega)$ by $+2$ and $+2.5$, respectively. We then have:

$$\pi_0^1(x,\omega_1) = \min\{2 - 0x, 0 + 1.25x\}$$

and

$$\pi_0^1(x,\omega_2) = \min\{2.5 - 0x, 0 + 1.25x\}.$$

Using the above data we formulate the RHS-LP (Problem 3.11, Ch. 3) for both ω_1 and ω_2. The optimal solution for ω_1 is $\delta(\omega_1) = 0$, $\sigma_0(\omega_1) = 0.55556$, $\sigma(\omega_1) = -0.555556$ and for ω_2 is $\delta(\omega_1) = 0$, $\sigma_0(\omega_1) = 0.5$, $\sigma(\omega_1) = -0.625$. We therefore

have $\nu^2(\omega_1) = 0$, $\gamma^2(\omega_1) = 1$, $\nu^2(\omega_2) = 0$ and $\gamma^3(\omega_1) = 1.25$. Translating back we obtain

$$\pi_c^1(x, \omega_1) = -2 + x$$

and

$$\pi_c^1(x, \omega_2) = -2.5 + 1.25x.$$

Note that since $x^1 = 2$ is an extreme point of X, we have that $\pi_0^1(2, \omega_1) = \pi_c^1(2, \omega_1) = 2$ and $\pi_0^1(2, \omega_2) = \pi_c^1(2, \omega_2) = 2.5$. Based on the RHS-LP (Problem 3.11, Ch. 3) solution we make the following updates: $r^2(\omega_1) = \begin{bmatrix} [r^1(\omega_1)] \\ -2 \end{bmatrix}$, $T^2(\omega_1) = \begin{bmatrix} [T^1(\omega_1)] \\ -1 \end{bmatrix}$. Similarly we update $r^2(\omega_2) = \begin{bmatrix} [r^1(\omega_2)] \\ -2.5 \end{bmatrix}$, $T^2(\omega_2) = \begin{bmatrix} [T^1(\omega_2)] \\ -1.25 \end{bmatrix}$. This completes Step 2 of the algorithm.

Step (*iii*)

Re-optimizing the updated scenario subproblems we obtain $y(\omega_1) = 0$ with $f_1(2, \omega_1) = 0$, and $y(\omega_2) = 0$ with $f_2(2, \omega_2) = 0$. The dual solutions are $d^\top(\omega_1) = [0, 0, 4]$ and $d^\top(\omega_1) = [0, 0, 4]$. Since both scenarios satisfy integral requirements on y we have an incumbent solution $x = 2$ and we update the upper bound $V_0 = \min\{V_0, -2 * 2 + 0.5 * (0) + 0.5 * (0) + 4\} = 0$.

Step (*iv*)

Using the dual solution for each scenario subproblem LP from Step (*iii*), we formulate the Benders-type optimality cuts. The resulting cuts are $-4x + \theta(\omega_1) \geq -8$ and $-5x + \theta(\omega_2) \geq -10$. Since the two scenarios are equally likely, the expected values associated with the cut coefficients yield $-4.5x + \theta \geq -9$. Applying the translation $\eta = \theta + 4$ we get $-4.5x + \eta \geq -5$ as the optimality cut to add to the

master program:

$$\begin{aligned} \text{Min } & -2x + \eta \\ \text{s.t. } & -x \qquad \geq -2 \\ & -4.5x + \eta \geq -5 \\ & x, \eta \geq 0. \end{aligned}$$

Solving the master program we get $x^2 = 1.111$, $\eta = 0$ and an objective value of -2.222. Therefore, the lower bound $v_0 = -2.222$. This completes this iteration of the D^2 algorithm. Since $V_0 > v_0$, $k \leftarrow 2$, and begin the next step of the D^2-CBAC algorithm.

Step 3: Branching

We now need to apply Proposition 2 with $x^2 = 1.111$ (use the translated functions $\pi_0^1(x, \omega)$ and $\pi_c^1(x, \omega)$):

For ω_1 we have:

$$\begin{aligned} \pi_0^1(1.111, \omega_1) &= \min\{2 - 0 * 1.5, 0 + 1.25 * 1.111\} \\ &= \min\{2, 1.389\} \\ &= 1.389 \end{aligned}$$

$$\begin{aligned} \pi_c^1(1.111, \omega_1) &= 0 + 1 * 1.111 \\ &= 1.111 \end{aligned}$$

Thus

$$\begin{aligned} p_\omega(\pi_0^1(1.111, \omega_1) - \pi_c^1(1.111, \omega_1)) &= 0.5 * (1.389 - 1.111) \\ &= 0.1389. \end{aligned}$$

For ω_2 we have:

$$\begin{aligned}\pi_0^1(1.111,\omega_2) &= \min\{2.5-0*1.111, 0+1.25*1.111\}\\ &= \min\{2.5, 1.389\}\\ &= 1.389\end{aligned}$$

$$\begin{aligned}\pi_c^1(1.111,\omega_2) &= 0+1.25*1.111\\ &= 1.389\end{aligned}$$

Thus

$$\begin{aligned}p_\omega(\pi_0^1(1.111,\omega_2)-\pi_c^1(1.111,\omega_2)) &= 0.5*(1.389-1.389)\\ &= 0.\end{aligned}$$

In this case we select $(\varpi,\kappa) = (\omega_1, 1)$ since Since $0.1389 > 0$. The branching constraint is therefore,

$$\begin{aligned}&\bar{\nu}_0^1(\omega_1)-\bar{\gamma}_0^1(\omega_1)x \geq \bar{\nu}_1^1(\omega_1)-\bar{\gamma}_1^1(\omega_1)x\\ \Rightarrow\quad &2-0x \geq 0+1.25x\\ \Rightarrow\quad &-x \geq -1.6\end{aligned}$$

So now we have for the first branch $-x \geq -1.6$ with the corresponding D^2 cut in the subproblem updated to $\pi^1 y \geq \bar{\nu}_1^1(\omega_1) - \bar{\gamma}_1^1(\omega_1)x \Rightarrow -1y \geq -2 + 1.25x$. For the second branch we have $x \geq 1.6$ with the corresponding D^2 cut updated to $\pi^1 y \geq \bar{\nu}_1^1(\omega_1) - \bar{\gamma}_1^1(\omega_1)x \Rightarrow -1y \geq 0 - 0x$ or $-y \geq 0$. Note that for this branch y is fixed at 0. We now create two nodal problems (P_1 and P_2) whose master programs are as follows.

Node 1 Master Program:

$$\begin{aligned} \text{Min } & -2x + \eta \\ \text{s.t. } & -x \geq -2 \\ & -4.5x + \eta \geq -5 \\ & -x \geq -1.6 \\ & x, \eta \geq 0. \end{aligned}$$

with optimal solution $x^2 = 1.111$, $\eta = 0$ and an objective value of -2.222. Therefore, the lower bound $v_1 = -2.222$.

Node 2 Master Program:

$$\begin{aligned} \text{Min } & -2x + \eta \\ \text{s.t. } & -x \geq -2 \\ & -4.5x + \eta \geq -5 \\ & x \geq 1.6 \\ & x, \eta \geq 0. \end{aligned}$$

with optimal solution $x^2 = 1.6$, $\eta = 2.2$ and an objective value of -1. Therefore, the lower bound $v_2 = -1$. Set $\mathcal{L} \leftarrow \mathcal{L} \cup \{P_1, P_2\}$ to the list of unexplored problems $\mathcal{L}$.

Step 1: Termination

List $\mathcal{L} \neq \emptyset$. Since $v_1 < v_2$ we select P_1 and set $\mathcal{L} \leftarrow \mathcal{L} \setminus \{P_1\}$.

Step 2: Apply one iteration of the D^2 Algorithm to the Nodal Problem

Step (i)

For Step 1 of the algorithm we use $x^2 = 1.111$ and solve the LP relaxation of the

second-stage subproblem for ω_1 and ω_2:

$$\begin{aligned} LP_1 : f_1(1.5, \omega_1) = \text{Min} \; & -4y \\ \text{s.t.} \; & -6y \geq -4.44445 \\ & -y \geq -1 \\ & -y \geq -0.61111 \\ & y \geq 0. \end{aligned}$$

and

$$\begin{aligned} LP_1 : f_2(2, \omega_2) = \text{Min} \; & -4y \\ \text{s.t.} \; & -6y \geq -6.44445 \\ & -y \geq -1 \\ & -y \geq -1.11111 \\ & y \geq 0. \end{aligned}$$

The optimal solution for LP_1 is $y(\omega_1) = 0.61111$ and for LP_2 is $y(\omega_2) = 1$

Step (*ii*)

Since $y(\omega_1)$ does not satisfy the integer restrictions, we choose y as the "disjunction" variable and create the disjunction $-y \geq 0$ or $y \geq 1$. We formulate the C^3-SLP (Problem 3.10, Ch. 3), which yields the vector π^2 for updating W^2 and the data for the RHS-LP (Problem 3.11, Ch. 3) whose optimal solution is used to update the right-hand side of the second-stage constraints. An optimal solution for the C^3-LP is $\pi^2 = -0.777778, \lambda_{0,1}^\top = [0,0,0], \lambda_{0,2} = 0.777778, \lambda_{1,1}^\top = [0.5,0,0]$, and $\lambda_{1,2} = 2.222223$.

We obtain W^3 by appending π^2 to W^2: $W^3 = \begin{bmatrix} [W^3] \\ -0.777778 \end{bmatrix}$. From the C^3-LP solution we obtain $\bar{\nu}_0^2(\omega_1) = 0$, $\bar{\nu}_1^2(\omega_1) = -2.77778$, $\bar{\gamma}_0^2(\omega_1)^\top = [0]$, $\bar{\gamma}_1^2(\omega_1)^\top = [-2.5]$

and $\bar{\nu}_0^2(\omega_2) = 0$, $\bar{\nu}_1^2(\omega_2) = -3.77778$, $\bar{\gamma}_0^2(\omega_2)^\top = [0]$, $\bar{\gamma}_1^2(\omega_2)^\top = [-2.5]$. In order to maintain $\pi_0^k(x,\omega) \geq 0$, we translate $\bar{\nu}_0^2(\omega)$ and $\bar{\nu}_1^2(\omega)$ by +2.77778 and +3.77778, respectively. We then have:

$$\pi_0^2(x,\omega_1) = \min\{2.77778 - 0x, 0 + 2.5x\}$$

and

$$\pi_0^2(x,\omega_2) = \min\{3.77778 - 0x, 0 + 2.5x\}.$$

Using the above data we formulate the *RHS*-LP (Problem 3.11, Ch. 3) for both ω_1 and ω_2. The optimal solution for ω_1 is $\delta(\omega_1) = 0$, $\sigma_0(\omega_1) = 0.642857$, $\sigma(\omega_1) = -0.892857$ and for ω_2 is $\delta(\omega_1) = 0$, $\sigma_0(\omega_1) = 0.569620$, $\sigma(\omega_1) = -1.075949$. We therefore have $\nu^2(\omega_1) = 0$, $\gamma^2(\omega_1) = 1.38889$, $\nu^2(\omega_2) = 0$ and $\gamma^3(\omega_1) = 1.88889$. Translating back we obtain

$$\pi_c^2(x,\omega_1) = -2.77778 + 1.38889x$$

and

$$\pi_c^2(x,\omega_2) = -3.77778 + 1.88889x.$$

Note that since $x^2 = 1.111$ is NOT an extreme point of X, we have that $\pi_0^2(1.111,\omega_1) > \pi_c^2(2,\omega_1)$ and $\pi_0^2(1.111,\omega_2) > \pi_c^1(1.111,\omega_2)$. Based on the *RHS*-LP (Problem 3.11, Ch. 3) solution we make the following updates: $r^3(\omega_1) = \begin{bmatrix} [r^2(\omega_1)] \\ -2.77778 \end{bmatrix}$, $T^3(\omega_1) = \begin{bmatrix} [T^2(\omega_1)] \\ -1.38889 \end{bmatrix}$. Similarly we update $r^3(\omega_2) = \begin{bmatrix} [r^2(\omega_2)] \\ -3.77778 \end{bmatrix}$, $T^3(\omega_2) = \begin{bmatrix} [T^2(\omega_2)] \\ -1.88889 \end{bmatrix}$. This completes Step 2 of the algorithm.

Step (*iii*)

Re-optimizing the updated scenario subproblems we obtain $y(\omega_1) = 0.6111$ and $y(\omega_2) = 1$. Note that the fractional solution for ω_1 has not been cut off by the D^2 cut. The dual solutions are $d^\top(\omega_1) = [0,0,4,0]$ and $d^\top(\omega_1) = [0,4,0,0]$. Since scenario ω_1 does not satisfy the integral requirements V_1 remains unchanged.

Step (*iv*)

Using the dual solution for each scenario subproblem LP from Step (*iii*), we formulate the Benders-type optimality cuts. The resulting cuts are $-5x + \theta(\omega_1) \geq -8$ and $0x + \theta(\omega_2) \geq -4$. Since the two scenarios are equally likely, the expected values associated with the cut coefficients yield $-2.5x + \theta \geq -6$. Applying the translation $\eta = \theta + 4$ we get $-2.5x + \eta \geq -2$ as the optimality cut to add to the master program:

$$\begin{aligned} \text{Min} \quad & -2x + \eta \\ \text{s.t.} \quad & -x \geq -2 \\ & -4.5x + \eta \geq -5 \\ & -x + \eta \geq -1.6 \\ & -2.5x + \eta \geq -2 \\ & x, \eta \geq 0. \end{aligned}$$

Solving the master program we get $x^2 = 0.8$, $\eta = 0$ and an objective value of -1.6. Therefore, the lower bound $v_1 = -1.6$. This completes this iteration of the D^2 algorithm. Since $V_1 > v_1$, $k \leftarrow 3$, and begin the next step of the D^2-CBAC algorithm.

Step 3: Branching

We now need to apply Proposition 2 with $x^2 = 0.8$:

For ω_1 we only have $\pi_0^2(x, \omega_1)$ to consider since $\pi_0^1(x, \omega_1)$ has already been branched on:

$$\begin{aligned} \pi_0^2(0.8, \omega_1) &= \min\{2.77778 - 0 * 0.8, 0 + 2.5 * 0.8\} \\ &= \min\{2.77778, 2\} \\ &= 2 \end{aligned}$$

$$\begin{aligned}\pi_c^2(0.8,\omega_1) &= 0 + 1.38889 * 0.8 \\ &= 1.1111\end{aligned}$$

Thus

$$\begin{aligned}p_\omega(\pi_0^2(0.8,\omega_1) - \pi_c^2(0.8,\omega_1)) &= 0.5 * (2 - 1.1111) \\ &= 0.4444.\end{aligned}$$

For ω_2 we have:

$$\begin{aligned}\pi_0^1(0.8,\omega_2) &= \min\{2.5 - 0 * 0.8, 0 + 1.25 * 0.8\} \\ &= \min\{2.5, 1\} \\ &= 1\end{aligned}$$

$$\begin{aligned}\pi_c^1(0.8,\omega_2) &= 0 + 1.25 * 0.8 \\ &= 1\end{aligned}$$

Thus

$$\begin{aligned}p_\omega(\pi_0^1(0.8,\omega_2) - \pi_c^1(0.8,\omega_2)) &= 0.5 * (1 - 1) \\ &= 0.\end{aligned}$$

$$\begin{aligned}\pi_0^2(0.8,\omega_2) &= \min\{3.77778 - 0 * 0.8, 0 + 2.5 * 0.8\} \\ &= \min\{3.77778, 2\} \\ &= 2\end{aligned}$$

$$\begin{aligned}\pi_c^2(0.8,\omega_2) &= 0 + 1.8889 * 0.8 \\ &= 1.5111\end{aligned}$$

Thus

$$\begin{aligned}p_\omega(\pi_0^2(0.8,\omega_2) - \pi_c^2(0.8,\omega_2)) &= 0.5 * (2 - 1.5111) \\ &= 0.24444.\end{aligned}$$

In this case we select $(\varpi, \kappa) = (\omega_1, 2)$ since Since $0.4444 > 0.24444 > 0$. The branching constraint is therefore,

$$\begin{aligned} & \bar{\nu}_0^2(\omega_1) - \bar{\gamma}_0^2(\omega_1)x \geq \bar{\nu}_1^2(\omega_1) - \bar{\gamma}_1^2(\omega_1)x \\ \Rightarrow \quad & 2.77778 - 0x \geq 0 + 2.5x \\ \Rightarrow \quad & -x \geq -1.1111 \end{aligned}$$

So now we have for the first branch $-x \geq -1.111$ with the corresponding D^2 cut in the subproblem updated to $\pi^2 y \geq \bar{\nu}_1^2(\omega_1) - \bar{\gamma}_1^2(\omega_1)x \Rightarrow -0.777778y \geq -2.77778 + 2.5x$. For the second branch we have $x \geq 1.111$ with the corresponding D^2 cut updated to $\pi^2 y \geq \bar{\nu}_1^2(\omega_1) - \bar{\gamma}_1^2(\omega_1)x \Rightarrow -0.77778y \geq 0 - 0x$ or $-y \geq 0$. Note that for this branch y is fixed at 0. We now create two nodal problems $P_3 and P_4$ whose master programs are as follows.

Node 3 Master Program:

$$\begin{aligned} \text{Min} \quad & -2x + \eta \\ \text{s.t.} \quad & -x \geq -2 \\ & -4.5x + \eta \geq -5 \\ & -x \geq -1.6 \\ & -2.5x + \eta \geq -2 \\ & -x \geq -1.111 \\ & x, \eta \geq 0. \end{aligned}$$

with optimal solution $x^2 = 0.8$, $\eta = 0$ and an objective value of -1.6. Therefore, the lower bound $v_3 = -1.6$.

Node 4 Master Program:

$$\begin{aligned} \text{Min } & -2x + \eta \\ \text{s.t. } & -x \geq -2 \\ & -4.5x + \eta \geq -5 \\ & -x \geq -1.6 \\ & -2.5x + \eta \geq -2 \\ & x \geq 1.111 \\ & x, \eta \geq 0. \end{aligned}$$

with optimal solution $x^2 = 1.111$, $\eta = 0.7775$ and an objective value of -1.4445. Therefore, the lower bound $v_4 = -1.4445$. Set $\mathcal{L} \leftarrow \mathcal{L} \cup \{P_3, P_4\}$ to the list of unexplored problems, which is now $\mathcal{L} = \{P_2, P_3, P_4\}$ with $v_2 = -1$, $v_3 = -1.6$ and $v_4 = -1.4445$.

Step 1: Termination

List $\mathcal{L} \neq \emptyset$. Since $v_3 < v_2$ and $v_3 < v_4$ we select P_3 and set $\mathcal{L} \leftarrow \mathcal{L} \setminus \{P_3\}$.

Step 2: Apply one iteration of the D^2 Algorithm to the Nodal Problem

Step (i)

For Step 1 of the algorithm we use $x^2 = 0.8$ and solve the LP relaxation of the

second-stage subproblem for ω_1 and ω_2:

$$\begin{aligned} LP_1 : f_1(0.8, \omega_1) = \text{Min } & -4y \\ \text{s.t. } & -6y \geq -6 \\ & -y \geq -1 \\ & -y \geq -1 \\ & -0.77778y \geq -0.77778 \\ & y \geq 0. \end{aligned}$$

and

$$\begin{aligned} LP_1 : f_2(0.8, \omega_2) = \text{Min } & -4y \\ \text{s.t. } & -6y \geq -8 \\ & -y \geq -1 \\ & -y \geq -1.5 \\ & -0.77778y \geq -2.66668 \\ & y \geq 0. \end{aligned}$$

The optimal solution for LP_1 is $y(\omega_1) = 1$ with $f_2(0.8, \omega_1) = -4$, and for LP_2 is $y(\omega_2) = 1$ with $f_2(0.8, \omega_2) = -4$. Since the solutions for both scenarios satisfy integral requirements we have an incumbent solution: $x = 1.6$ and we update the upper bound $V_4 = \min\{V_1, -2 * 0.8 + 0.5 * (-4) + 0.5 * (-4) + 4\} = -1.6$ (Note that $V_1 = 0$ from the first iteration of the D^2-CBAC algorithm).

The dual solutions are $d^\top(\omega_1) = [0, 0, 0, 5.142856]$ and $d^\top(\omega_1) = [0, 0, 0, 4]$. We now go to step (iv) of the D^2 algorithm.

Step (iv)

Using the dual solution for each scenario subproblem LP from Step (ii), we formulate the Benders-type optimality cuts. The resulting cuts are $-12.8571x +$

$\theta(\omega_1) \geq -14.2857$ and $7.5556x + \theta(\omega_2) \geq -15.1111$. Since the two scenarios are equally likely, the expected values associated with the cut coefficients yield $-10.2064x + \theta \geq -14.6984$. Applying the translation $\eta = \theta + 4$ we get $-10.2064x + \eta \geq -10.6984$ as the optimality cut to add to the master program:

$$\begin{aligned}
\text{Min } & -2x + \eta \\
\text{s.t. } & -x \geq -2 \\
& -4.5x + \eta \geq -5 \\
& -x \geq -1.6 \\
& -2.5x + \eta \geq -2 \\
& -x \geq -1.111 \\
& -10.2064x + \eta \geq -10.6984 \\
& x, \eta \geq 0.
\end{aligned}$$

Solving the master program we get $x^4 = 0.8$, $\eta = 0$ and an objective value of -1.6. Therefore, the lower bound $v_3 = -1.6$. This completes this iteration of the D^2 algorithm. Since $V_3 = v_3$, We fathom this node by *optimality* and update the overall bound $V \leftarrow V_3$.

We now need to scan the problem on the list $\mathcal{L}$ of open problems. We have two unexplored nodal problems: $\mathcal{L} = \{P_2, P_4\}$ with $v_2 = -1$ and $v_4 = -1.4445$. Since $v_2 > V$ and $v_4 > V$ we fathom both nodes by *bound* and set $\mathcal{L} \leftarrow \mathcal{L} \setminus \{P_2, P_4\}$.

Step 1: Termination

List $\mathcal{L} = \emptyset$. Therefore, the algorithm terminates with $x = 0.8$ as the optimal solution with expected objective value $= -1.6 - 4 = -5.6$. And this concludes the example illustration.

8.5 Extensions

The derivation of the D^2-CBAC algorithm focused purely on SMIP with continuous first-stage. Next we need to extend this approach to mixed-binary, mixed-integer, and pure-integer cases. For the all these three cases if the master program is solved as an MIP then we can basically follow the approach outlined in this chapter. Whenever $\pi_0^k(\bar{x},\omega) \neq \pi_c^k(\bar{x},\omega)$ for some $\bar{x}$ and $\omega \in \Omega$ we can do branching in the first-stage based on Proposition 2. Nevertheless, a thorough analysis of such an approach is yet to be made.

On the other hand if the master program is solved as an LP relaxation, then the situation becomes a lot more complex. This would require not only branching on the continuous variables but also on the first-stage integer decision variables whose solution is fractional. The ideas presented in this chapter provides a starting point.

8.6 Summary

This chapter has extended the D^2 approach to SMIP with continuous or mixed-binary first-stage. In particular, a branch-and-cut method is derived and algorithm convergence proved. This approach is fairly unique in that branching is done on a continuous domain and is guided by the disjunction variables in the second-stage. A simple example problem that illustrates the proposed method is given. The ideas presented in this chapter should be best construed as conceptual.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

The research in this dissertation has demonstrated the potential application of the disjunctive decomposition (D^2) approach towards solving large-scale SCO problems. These problems result in very large scale instances which are comprised of loosely coupled subsystems. By taking advantage of the loosely coupled structure of SCO problems, it is shown that the divide-and-conquer paradigm of decomposition-coordination methods provide a highly effective algorithm, and surpasses the scalability of even the most efficiently implemented backtracking search algorithms.

Computer implementation of D^2 methods and computational experimentation with several instances of SCO problem instances are undertaken. The modeling aspect of the research is conducted by deriving a new model for the server location problem under uncertainty as well as experimenting with the models from the literature. The D^2 method is also applied to stochastic matching and stochastic strategic supply chain planning problems from the literature. The two models are of a large-scale nature, have continuous decision variables in the second stage, and are quiet different in how uncertainty is revealed in each one.

The appropriateness of the D^2 approach is determined through computational experiments on numerous large-scale problem instances from the three application areas. Comparisons of the computation time results with those obtained by a state-of-art optimization software applied to the DEP are done. The effect of various

aspects of the problem on computation time and algorithm convergence are also analyzed. Large-scale problem instances comprising of lots of discrete decision variables and constraints are solved within reasonable time. The results of the study reveal the promising potential of the D^2 approach towards solving large-scale SMIP problems.

In order for the current D^2 method to converge, the first-stage solutions are required to be extreme points of the first-stage feasible set. This research extends the D^2 approach to SCO problems where this requirement is not necessary. In particular, a branch-and-cut method for two-stage SMIP with continuous first-stage is derived.

9.2 Contributions of This Research

This research has made modeling and computational contributions to stochastic combinatorial optimization. These contributions include the following:

1. Computer implementation of the D^2 algorithm and the identification of the issues associated such an implementation. These issues will potentially translate to future algorithms that follow a similar approach.

2. Proposing a new model for server location under uncertainty (the SSLP) with potential use in a variety of application domains. Conducting a computational study of the SSLP and demonstrating that significant gains can be made by the application of the D^2 method to SSLP. As a by-product of this experiment, we have developed SSLP test problems that can be used to test the performance of other algorithms. These test problems will be made available via SIPLIB at http://www.isye.gatech.edu/~sahmed/siplib/.

3. Solving some of the largest SCO problem instances reported in the literature to date. Some of these problem instances have up to over a hundred thousand

constraints and over a million binary variables. Furthermore, this research has revealed that the convergence of the D^2 method on large-scale SCO problems is in fact attainable. In addition, the dissertation demonstrates the applicability of the D^2 approach to stochastic strategic supply chain planning and stochastic matching problem instances from the literature.

4. Finally, an extension of disjunctive decomposition to two-stage SMIP with continuous first-stage is made and a new branch-and-cut procedure is proposed.

9.3 Future Work

Although this research has contributed towards solving some of the largest SMIP problem instances to date, the challenge of solving general SMIP problems is far from over. In fact, we are still in the early stages of this field. A lot of research still remains to be done in order to meet the "grand challenge" imposed by SCO. The following are some of the research areas along this line of work that remain to be explored:

1. This research has illustrated, implemented and experimented with a sequential version of the D^2 algorithm. In order to get the most out of the decomposition algorithm, distributed/parallel implementation and computational experimentation of the algorithm is needed. In fact, distributed computing may provide the only way to conquer some of the most difficult and challenging large-scale SCO problems. Towards this goal, the conference paper by (Ntaimo and Yu, 2004) has laid the basic framework for a distributed implementation of the D^2 algorithm.

2. Chapter 8 provides a specific direction for future research. This chapter has extended disjunctive decomposition to SMIP problems with first-stage

solutions that are no longer required to be extreme points of X. The next task is to implement the ideas presented in Chapter 8 and conduct computational experiments to assess the performance of the proposed method.

3. A vast majority of important combinatorial optimization problems are NP-hard and in a lot of cases their solutions require *approximation* algorithms. Other than striving to solve SCO problems to optimality as in this dissertation, future research would involve settling for "good" approximate solutions. The design and analysis of approximations/heuristics for large scale SCO remains an open research area (Stougie and van der Vlerk, 2003).

4. Finally, in many real life problems decisions have to be made sequentially over time under uncertainty. Therefore, there is a need for algorithms that allow for optimal decisions to be made accordingly and this leads to the multistage case. Another research direction is to extend the D^2 approach to the multistage case. This will allow for the development of new algorithms for multistage SMIP.

REFERENCES

Ahmed, S. and R. Garcia (2003). Dynamic capacity acquisition and assignment under uncertainty. *Annals of Operational Research*. to appear.

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

Ahmed, S., M. Tawarmalani, and N. V. Sahinidis (2004). A finite branch and bound algorithm for two-stage stochastic integer programs. *To appear in Mathematical Programming*. http://www.isye.gatech.edu/so/publications/.

Albareda-Sambola, M., M. H. van der Vlerk, and E. Fernandez (2002). Exact solutions to a class of stochastic generalized assignment problems. Research Report 02A11, SOM, University of Groningen, The Netherlands, http://som.rug.nl.

Alonso, A., L. F. Escudero, and M. T. Ortuño (2000). A stochastic 0-1 program based approach for the air traffic flow management problem. *European Journal of Operations Research* **120**, 47–62.

Alonso-Ayuso, A., L. F. Escudero, A. Garín, M. T. Ortuño, and G. Perez (2003). An approach for strategic supply chain planning under uncertainty based on stochastic 0-1 programming. *Journal of Global Optimization* **26**, 97–124.

Baker, S., D. Morton, R. Rosenthal, and L. Williams (2002). Optimizing military airlift. *Operations Research* **50**, 582–602.

Balas, E. (1979). Disjunctive programming. *Annals of Discrete Mathematics* **5**, 3–51.

Balas, E., E. S. Ceria, and G. Cornuéjols (1993). A lift-and-project cutting plane algorithm for mixed 0-1 integer programs. *Mathematical Programming* **58**, 295–324.

Barahona, F., S. Bermon, O. Gunluk, and S. Hood (2001). Robust capacity planning in semiconductor manufacturing. IBM Research Report RC22196, IBM.

Beale, E. (1955). On minimizing a convex function subject to linear inequalities. *Journal of Royal Statistical Society, Series B* **17**, 173–184.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variable

programming problems. *Numerische Mathematic* **4**, 238–252.

Berman, O. and D. Simchi-Levi (1988). Finding the optimal a priori tour and location of a traveling salesman with nonhomogeneous customers. *Transportation Science* **22(2)**, 148–154.

Bertsimas, D. (1994). A mathematical programming approach to stochastic and dynamic optimization problems. Technical Report, Operations Research Center, MIT, Cambridge, MA.

Bertsimas, D. and M. Sim (2003). Robust discrete optimization and network flows. *Mathematical Programming Series B* **98**, 49–71.

Birge, J. R. (1985). Decomposition and partitioning methods for multi-stage stochastic linear programs. *Operations Research* **33**, 989–1007.

Birge, J. R. (1995). *Solution Methods for Stochastic Dynamic Linear Progams.* Stanford University, Stanford, CA: Ph.D. Dissertation. Technical Report SOL 81-29, Systems Optimization Lab, Stanford University.

Birge, J. R. (1997). Stochastic programming computation and applications: State of the art. *INFORMS Journal on Computing* **9(2)**, 111–133.

Birge, J. R. and F. V. Louveaux (1988). A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operations Research* **34**, 384–392.

Birge, J. R. and F. V. Louveaux (1997). *Introduction to Stochastic Programming.* New York: Springer.

Blair, C. (1995). A closed-form representation of mixed-integer program value functions. *Mathematical Programming* **71**, 127–136.

Blair, C. and R. Jeroslow (1982). The value function of an integer program. *Mathematical Programming* **23**, 237–273.

Carinõ, D., T. Kent, D. Meyers, C. Stacy, M. Stylvanus, A. Turner, K. Watanabe, and W. Ziemba (1994). The Russel-Yasuda Kasai Model: An asset/liability model for a Japanese insurance company using multistage stochastic programming. *Interfaces* **24**, 29–49.

Carøe, C. and R. Schultz (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters* **24**, 37–45.

Carøe, C. C. (1998). *Decomposition in Stochastic Integer Programming.* Dept. of Operations Research, University of Copenhagen, Denmark: Ph.D. thesis.

Carøe, C. C. and R. Schultz (1998). A two-stage stochastic program for unit commitment under uncertainty in a hydro-thermal power system.

Preprint 98-13, Echtzeit-Optimierung groer Systeme. http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/Preprint-98-13.html.

Carpenter, T., I. Lustig, and J. Mulvey (1991). Formulating stochastic programs for interior point methods. *Operations Research* **39**, 757–770.

Charnes, A. and W. Cooper (1959). Chance constrained programming. *Management Science* **5**, 73–79.

Cook, W. J., W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver (1998). *Combinatorial Optimization.* New York: John Wiley and Sons.

Cormican, K., D. Morton, and R. Wood (1998). Stochastic network interdiction. *Operations Research* **46**, 184–197.

Coy, P. (1996). The CFO goes 3D. *Business Week*. October 28.

Dantzig, G. (1955). Linear programming under uncertainty. *Management Science* **1**, 197–206.

Dantzig, G. and P. Glynn (1990). Parallel processors for planning under uncertainty. *Annals of Operations Research* **22**, 1–21.

Dantzig, G. and G. Infanger (1991). Large-scale stochastic linear programs - Importance sampling and Benders decomposition. In C. Brezinski and U. Kulisch (Eds.), *Computational and Applied Mathematics I (Dublin, 1991)*, pp. 111–120. North-Holland, Amsterdam.

Dantzig, G. and A. Madansky (1961). On the solution of two-stage linear programs under uncertainty. In J. Neyman (Ed.), *Proceedings of the 4th Berkeley Symposiom on Mathematical Statistics and Probability*, pp. 165–176. Berkeley, CA: University of California Press.

Doverspike, R. D. (2003). Private communication.

Dyer, M. and L. Stougie (2003). Computational complexity of stochastic programming problems. Spor-report 2003-09, Dept. of Mathematics and Computer Science, Eindhoven Technical University, Eindhoven.

Edirisinghe, N. and W. Ziemba (1996). Implementing bounds-based approximations in convex-concave two stage programming. *Mathematical Programming* **19**, 314–340.

Edmonds, J. (1965). Maximum matching and a polyhedron with 0-1 vertices. *Journal of research of the national bureau of standards* **69B**, 125–130.

Eppen, G., R. Martin, and L. Schrage (1989). A scenario approach to capacity planning. *Operations Research* **37**, 517–527.

Escudero, L., E. Galindo, E. Gómez, G. García, and V. Sabau (1996). SCHUMAN, a modeling framework for supply chain management under uncertainty. *European Journal of Operational Research* **119**, 13–34.

Ferguson, A. and G. Dantzig (1956). The allocation of aircraft to routes: An example of linear programming under uncertain demands. *Management Science* **3**, 45–73.

Fortnow, L. (1997). Counting complexity. In L. Hemaspaandra and A. Selman (Eds.), *Complexity Theory Retrospective II*, pp. 81–107. Berlin: Springer-Verlag.

Frantzeskakis, L. and W. Powell (1990). A successive approximation procedure for stochastic dynamic vehicle allocation problems. *Transportation Science* **24**, 40–57.

Frauendorfer, K. (1992). Stochastic two-stage programming. *Lecture Notes in Economics and Mathematical Systems* **392**.

Frauendorfer, K. (1994). Multistage stochastic programming: error analysis for the convex case. *Zeitschrift fur Operations Research* **39**, 93–122.

Gassmann, H. (1990). MSLiP: A computer code for the multistage stochastic linear programming problem. *Mathematical Programming* **47**, 407–423.

Hauskrecht, M. and E. Upfal (2001). A clustering approach to solving large stochastic matching problems. *Proceedings of the Seventeenth International Conference on Uncertainty in Artificial Intelligence*, 219–226. http://www.cs.pitt.edu/ milos/publications.html.

Higle, J., B. Rayco, and S. Sen (2002). Stochastic scenario decomposition for multistage stochastic programs. *submitted to Operations Research.*

Higle, J. and S. Sen (1991). Stochastic Decomposition: An algorithm for two-stage linear programs with recourse. *Mathematics of Operations Research* **16**, 83–112.

Higle, J. and S. Sen (1996). *Stochastic Decomposition: A Statistical Method for Large Scale Stochastic Linear Programming.* Dordrecht: Kluwer Academic Publishers.

Higle, J. and S. Sen (1999). Statistical approximations for stochastic linear programming problems. *Annals of Operations Research* **85**, 173–192.

Higle, J. and S. Sen (2002). Duality for multistage convex stochastic programs. *to appear in Annals of Operations Research.*

Horst, R. and H. Tuy (1996). *Global Optimization: Deterministic Approaches* (3rd ed.). Berlin: Springer-Verlag.

ILOG, I. (2000). *CPLEX 7.0 Reference Manual.* Incline Village, NV: ILOG CPLEX Division.

Infanger, G. (1991). Monte Carlo (importance) sampling within a Benders decomposition algorithm for stochastic linear programs; Extended version: Including results of large-scale problems. Technical report sol 91-6, Systems Optimization Lab, Dept. of Operations Research, Stanford University, Stanford, CA.

Jorjani, S., C. H. Scott, and D. L. Woodruff (1995). Selection of an optimal subset of sizes. Technical report, Univ. of California, Davis, CA.

Kall, P. (1976). *Stochastic Linear Programming.* Berlin: Springer-Verlag.

Kall, P. (1979). Computational methods for solving two-stage stochastic linear programming problems. *Journal of Applied Mathematics and Physics* **30**, 261–271.

Kall, P. and J. Mayer (1996). SLP-IOR: An interactive model management system for stochastic linear programs. *Mathematical Programming, Series B* **75**, 221–240.

Kall, P. and S. Wallace (1994). *Stochastic Programming.* Chichester England: John wiley & Sons.

Kelley, J. (1960). The cutting plane method for convex programs. *Journal of SIAM* **8**, 703–712.

Kenyon, A. S. and D. P. Morton (2003). Stochastic vehicle routing with random travel times. *Transportation Science* **37(1)**, 69–82.

Klein Haneveld, W., L. Stougie, and M. van der Vlerk (1995). On the convex hull of the simple integer recourse objective function. *Annals of Operations Research* **56**, 209–224.

Klein Haneveld, W., L. Stougie, and M. van der Vlerk (1996). An algorithm for construction of convex hulls in simple integer recourse programming. *Annals of Operations Research* **64**, 67–81.

Kong, N. and A. J. Schaefer (2004). A factor $\frac{1}{2}$ approximation algorithm for two-stage stochastic matching problems. *submitted to INFORMS Journal on Computing*. http://www.ie.pitt.edu/ schaefer/SIP.htm.

Kusy, M. and W. Ziemba (1986). A bank asset and liability management model. *Operations Research* **34**, 356–376.

Laporte, G. and F. V. Louveaux (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters* **1**, 133–142.

Laporte, G., F. V. Louveaux, and H. Mecure (1994). An exact solution for the a

priori optimization of the probabilistic traveling salesman problem. *Operations Research* **39**, 71–78.

Laporte, G., F. V. Louveaux, and L. van Hamme (2002). An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research* **50**, 415–423.

Linderoth, J. and S. J. Wright (2003). Decomposition algorithms for stochastic programming on a computational grid. *Computational Optimization and Applications* **24**, 207–250.

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

Lustig, I., R. Marsten, and D. Shanno (1991). Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra and its application* **152**, 191–222.

Lustig, I., R. Marsten, and D. Shanno (1994). Interior point methods for linear programming: Computational state of the art. *ORSA Journal on Computing* **6**, 1–14.

Markowitz, H. (1952). Portfolio selection. *Journal of Finance* **7**, 77–91.

MirHassani, S., C. Lucas, G. Mitra, and C. Poojari (2000). Computational solution of capacity planning model under uncertainty. *Parallel Computing Journal* **26(5)**, 511–538.

Morton, D., R. Rosenthal, and L. Weng (1996). Optimizion for military airlift. *Military Operations Research* **1**, 49–68.

Mulvey, J. and A. Ruszczynski (1995). A new scenario decomposition method for large-scale stochastic optimization. *Operations Research* **43(3)**, 477–489.

Norkin, V., Y. Ermoliev, and A. Ruszczyński (1998). On optimal allocation of indivisibles under uncertainty. *Operations Research* **46(3)**, 381–395.

Ntaimo, L. and L. Yu (2004). Distributed discrete optimization under uncertainty. *Proceedings of IIE Conference 2004*.

Ogryczak, W. and A. Ruszczynski (2002). Dual stochastic dominance and related mean-risk models. *SIAM Journal on Optimization* **13**, 60–78.

Pereira, M. and L. Pinto (1985). Stochastic optimization of a multireservior hydroelectric system - A decomposition approach. *Water Resources Research* **21**, 779–792.

Pereira, M. and L. Pinto (1991). Multistage stochastic optimization applied to

energy planning. *Mathematical Programming* **52**, 359–375.

Powell, W. (1988). A comparative review of alternative algorithms for the dynamic vehicle allocation program. In B. Golden and A. Assad (Eds.), *Vehicle Routing: Methods and Studies*. North-Holland.

Powell, W. (1990). Real-time optimization for truckload motor carriers. *OR/MS Today* **17**, 28–33.

Powell, W. (1996). A stochastic formulation of the dynamic assignment problem, with an application to truckload motor carriers. *Transportation Science* **3**, 195–219.

Powell, W. and D. Gittoes (1996). An approximate labeling algorithm for the dynamic assignment problem. In M. B. L. Bianco, P. Toth (Ed.), *Advanced Methods in Transportation Analysis*, pp. 547–584. Springer, New York.

Powell, W., W. Snow, and R. K.-M. Cheung (2004). Adaptive labeling algorithms for the dynamic assignment problem. *Transportation Science*.

Prékopa, A. (1971). Logarithmic concave measures with application to stochastic programming. *Acta Scientiarium Mathematiarum (Szeged)* **32**, 301–316.

Prékopa, A. (1995). *Stochastic Programming*. Dordrecht, The Netherlands: Kluwer.

Riis, M. and Schultz (2003). Applying the minimum risk criterion in stochastic recourse programs. *Computational Optimization and Applications* **16**, 267–288.

Riis, M., A. Skriver, and J. Lodahl (2004). Deployment of mobile switching centers in a telecommunications network: A stochastic programming approach. *to appear*. http://home.imf.au.dk/riis/.

Rockafellar, R. and S. Uryasev (2000). Optimization of conditional value-at-risk. *The Journal of Risk* **2(3)**, 21–41.

Rockafellar, R. and S. Uryasev (2002). Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance* **26(7)**, 1443–1471.

Rockafellar, R. and R.-B. Wets (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research* **16**, 119–147.

Rockafellar, R. and R.-B. Wets (1992). A dual strategy for the implementation of the aggregation principle in decision making under uncertainty. *Applied Stochastic Models and Data Analysis* **8**, 245–255.

Römish, W. and R. Schultz (1991). Distribution sensitivity in stochastic programming. *Mathematical Programming* **50**, 197–226.

Rosa, C. and A. Ruszczynski (1996). On augmented lagrangian decomposition methods for multistage stochastic programs. *Annals of Operations Research Letters* **64**, 289–309.

Ruszczynski, A. (1993). Parallel decomposition of multistage stochastic programs. *Mathematical Programming* **58**, 201–228.

Sand, G. and S. Engell (2003). A two-stage stochastic programming approach to real-time scheduling. In I. Grossmann and C. McDonald (Eds.), *Fourth International Conference on Foundations of Computer-Aided Process Operations*, pp. 347–350. Austin, TX : CACHE Corp. http://ametist.cs.utwente.nl/INTERNAL/PUBLICATIONS/DORTMUNDPublications/Dortmund_Publications.html.

Schultz, R., , L. Stougie, and M. van der Vlerk (1998). Solving stochastic programs with integer recourse by enumeration: a framework using Gröbner basis. *Mathematical Progamming* **83**, 71–94.

Schultz, R. (1993). Continuity properties of expectation functions in stochastic integer programming. *Mathematics of Operations Research* **18**, 578–589.

Schultz, R. (2003). Integers in stochastic programming. Preprint series: Smd, 543, Mathematik - Universitaet Duisburg-Essen.

Schultz, R., L. Stougie, and M. van der Vlerk (1996). Two-stage stochastic integer programming: A survey. *Statistica Neerlandica* **50**, 404–416.

Sen, S. (1992). Relaxations for probabilistically constrained programs with discrete random variables. *Operations Research* **18**, 81–86.

Sen, S. (2003). Algorithms for stochastic mixed-integer programming models. In K. Aardal, G. Nemhauser, and R. Weismantel (Eds.), *Stochastic Integer Programming Handbook*, Chapter 18. Dordrecht, The Netherlands. http://www.sie.arizona.edu/MORE/papers/SIPHbook.pdf.

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

Sen, S. and J. Higle (1999). An introductory tutorial on stochastic linear programming models. *Interfaces* **29(2)**, 33–61.

Sen, S. and J. Higle (2000). The C^3 theorem and a D^2 algorithm for large scale stochastic integer programming: Set convexification. *Stochastic E-Print Series*. http://dochost.rz.hu-berlin.de/speps/.

Sen, S., J. L. Higle, and L. Ntaimo (2002). A summary and illustration of disjunctive decomposition with set convexification. In D. L. Woodruff (Ed.), *Stochastic Integer Programming and Network Interdiction Models*, Chapter 6. Dordrecht,

The Netherlands: Kluwer Academic Press.

Sen, S. and H. Sherali (2004). Decomposition with branch-and-cut approaches for two stage stochastic mixed-integer programming. *submitted to Mathematical Programming*. http://www.sie.arizona.edu/MORE/papers/dbacs.pdf.

Sen, S. and H. D. Sherali (1987). Nondifferentiable reverse convex programs and facetial cuts via a disjunctive characterization. *Mathematical Programming* **37**, 169–183.

Sen, S. and H. D. Sherali (2003). Decomposition with branch-and-cut approaches for two stage stochastic mixed-integer programming. *to appear in Mathematical Programming*. http://www.sie.arizona.edu/MORE/papers/MayRevDbacs.pdf.

Sen, S., L. Yu, and T. Genc (2003). A stochastic programming approach to power portfolio optimization. *submitted*. http://tucson.sie.arizona.edu/MORE/papers.html.

Shapiro, A. (1991). Asymptotic analysis of stochastic programs. *Annals of Operations Research* **30**, 169–186.

Shapiro, A. and T. Homem de Mello (1998). A simulation-based approach to stochastic programming with recourse. *Mathematical Programming* **81**, 301–325.

Sherali, H. and W. Adams (1990). A hierarchy of relaxations between the continuous and convex hull representations for 0-1 programming problems. *SIAM J. on Discrete Mathematics* **3**, 411–430.

Sherali, H. and 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**, 319–342.

Sherali, H. and C. Shetty (1980). Optimization with disjunctive constraints. *Lecture Notes in Economics and Mathematical Systems* **181**, 411–430.

Simms, A. E. (Ed.) (1997). *A stochastic approach to modeling aviation security problems using the KNAPSACK problem.* Virginia Tech., Blacksburg, Virginia: M.S. Thesis. http://scholar.lib.vt.edu/theses/available/etd-53097-132255/unrestricted/asimms.pdf.

Stougie, L. (1985). Design and analysis of algorithms for stochastic integer programming. Ph.d. thesis, Center for Mathematics and Computer Science, Amsterdam, The Netherlands.

Stougie, L. and M. H. van der Vlerk (2003). Approximation in stochastic integer programming. Research Report 03A14, SOM, Univ. of Groningen, The Netherlands. http://www.ub.rug.nl/eldoc/som/a/03A14/03A14.pdf.

Strazicky, B. (1980). Some results concerning an algorithm for discrete recourse problem. In M. Dempster (Ed.), *Stochastic Programming*. New York: Academic Press.

Takriti, S. and S. Ahmed (2002). On robust optimization of two-stage systems. *To appear in Mathematical Programming*.

Takriti, S., J. Birge, and E. Long (1996). A stochastic model for the unit commitment problem. *IEEE Transactions on Power Systems* **11**, 1497–1508.

Uryasev, S. and P. M. Pardalos (2001). *Stochastic Optimization: Algorithms and Applications*. New York: Kluwer Academic Publishers.

Valiant, L. (1979). The complexity of enumeration and reliability problems. *SIAM Journal on Computing* **8**, 410–421.

van der Vlerk, M. (1995). Stochastic programming with integer recourse. Ph.d. thesis, Rijksuniversiteit Groningen, The Netherlands.

van der Vlerk, M. (2002). On the convex hull of the simple integer recourse objective function. Som research report 02a21, University of Groningen. also, Stochastic Programming E-Print Series 2002-10.

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

Wald, A. (1950). *Statistical Decision Functions*. Mew York: Wiley.

Wang, Q., E. Batta, and C. M. Rump (2003). Facility location models for immobile servers with stochastic demand. *Naval Research Logistics*. Submitted.

Wets, R. J.-B. (1974). Stochastic programs with fixed recourse: the equivalent deterministic problem. *SIAM Review* **16**, 309–339.

Wolsey, L. (1998). *Integer Programming*. New York: Wiley & Sons.

Woodruff, D. L. (Ed.) (2002). *Stochastic Integer Programming and Network Interdiction Models*. Dordrecht, The Netherlands: Kluwer Academic Press.