Time-staged decomposition and related algorithms for stochastic mixed-integer programming

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Yunwei Qi, B.S.

Graduate Program in Department of Integrated Systems and Engineering

The Ohio State University

2012

Dissertation Committee:

Suvrajeet Sen, Advisor

Simge Küçükyavuz

Theodore Allen

Abstract

This dissertation focuses on solving two-stage stochastic mixed integer programs (SMIPs) with general mixed integer variables in both stages. Our setup allows randomness in all data elements influencing the recourse problem, and moreover, general integer variables are allowed in both stages. We develop a time-staged decomposition algorithm that uses multi-term disjunctive cuts [1] to obtain convex approximation of the second-stage mixed-integer programs. We prove that the proposed method is finitely convergent. Among the main advantages of our decomposition scheme is that the subproblems are approximated by successive linear programming problems, and moreover these can be solved in parallel. Several variants of an SMIP example in the literature are included to illustrate our algorithms. To the best of our knowledge, the only previously known time-staged decomposition algorithm to address the two-stage SMIP in such generality used operations that are computationally impractical (e.g. requiring exact value functions of MIP subproblems). In contrast, our decomposition algorithm allows partially solving the subproblems. Following the studies of our decomposition algorithm, we proceed with computational studies related to some of the key ingredients of our decomposition algorithm. First, we investigate how well multi-term disjunctions can approximate feasible sets associated with stochastic mixed-integer programming problems. This part of our study is experimental in nature and we investigate both "wait-and-see" as well as "here-and-now" formulations of stochastic programming problems.

[1] Disjunctive cut is the cut generated out of a set of disjunctions

In order to study the performance for the former class of problems, we use test problems from the integer programming literature (e.g. various versions of MIPLIB), whereas for the latter class of problems, we use the SSLP series of instances. Another important nugget of our decomposition algorithm is the use of multi-term disjunctions. Since the effectiveness of our scheme depends on this feature, we also investigate ways to improve the performance of cutting plane tree (CPT) algorithm for mixed integer programming problems. We compare different variable splitting rules in the computational experiment. A set of algorithms for solving multi-term CGLPs are also included and computational experiments with instances from MIPLIB are performed.

To my parents, Pichao Qi, Ying Liu and my wife Lili Zhuang

Acknowledgments

I would like to express my deep and sincere gratitude to my advisor, Dr. Suvrajeet Sen, for his inspirational and patient guidance, general support and unselfish help. Without his encouragement and inclusiveness, I would never finish my dissertation successfully.

I want to thank Dr. Küçükyavuz Simge for her guidance and insight, especially on Chapter 4. Also I want to thank her and Dr. Theodore Allen for serving on my committee. Many thanks to Dr. Julie L Higle for her guidance and help throughout my PhD studies. My gratitude also goes to Dr. Marc E. Posner, Dr. Radu Herbei and Dr. Srinivasan Parthasarathy for your constructive comments on my research. My gratitude also extends to Cedric, Mike, Tammy, Clarice, Kristen, Judy, Pam, and Candi in the Integrated Systems Engineering department for their help and services in all aspects.

Special thanks go to my friends and colleagues Yang Yuan, Binyuan Chen, Yifan Liu, Dinakar Gade, Shugang Kang, Harsha Gangammanavar, Paraneeth AVS and many others who gave me so much kindly help and happy time during my time at OSU. I also want to thank Dr. Attila Lengyel for being supportive on my PhD.

Finally, I would like to thank my parents and my wife for their love, support and encouragement who always be there for me in all my pursuits. Special thanks to my uncle and aunt who make me feel less homesick.

Vita

December 21, 1983	Born - Dalian, China
2002-2006	B.S. Computer Science and Engieering, Dalian University of Technology
2006-2011	Graduate Research Associate, The Ohio State Univeristy

Fields of Study

Major Field: Department of Integrated Systems and Engineering

Table of Contents

Page

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

Stochastic programming (SP) is one branch in the mathematical optimization field which deals with mathematical models with random parameters. It provides one way to solve decision-making problems under uncertainty. In particular, stochastic linear programs(SLP) requires all decision variables to be continuous in the SP model and stochastic mixed integer programs(SMIPs) considers modeling with mixed-integer variables. This dissertation focuses on the latter class of models.

In SMIPs, discrete decisions need to be made both before and after the observation of the random variables. Applications include optimal choice of server locations under uncertainty, power generation planning under uncertainty, vaccine design and production, airline scheduling, data mining and many more. It has long been recognized as an important class of models for many practical operational problems (see e.g. [Wol80]). Despite its widespread applicability, algorithmic advances to solve SMIP models have lagged other forms of stochastic programming (SP). In addition to the standard difficulties associated with stochastic linear programming (e.g. designing scalable ways to approximate the expected recourse/value function), SMIP models with integer variables in the second stage generates a nonconvex and possibly discontinuous value function.

The progress to solve SMIP models during the past decade has been significant. A range of algorithms for different model characteristics are available. However, time-staged decomposition algorithm for SMIP models with general integer variable in both stages has not been well understood. The two main approaches for solving two-stage SMIP models (with general integers in both stages), amounts to solving a large deterministic equivalent problem (DEP) either using a general-purpose MIP solver, or combining the latter with Lagrangian relaxation for lower bounding computations of each scenario (Caroe and Schultz 1999). However, the SP literature has demonstrated that time-staged decomposition (e.g. L-shaped/Benders', regularized decomposition and similar methods) is more scalable than solving the DEP. Hence the motivation is to design a time-staged decomposition algorithm for SMIP.

1.2 Literature Review

The decomposition algorithm for solving SMIP can be traced back to 1993 when the integer L-shaped method was proposed in [LL93] for pure binary variables in the master problem and mixed-integer variable in the subproblem. It is a "L-shaped" method which deals with the problem with same structure as Benders' decomposition allowing discrete decision variables in both stages. In the same year, [LvdV93] showed a decomposition algorithm with simple integer recourse. Following these two papers, there has been significant progress with algorithms for SMIP. In 1998, the work by [CT98] presented a decomposition algorithm based on mixed integer programming (MIP) duality, and while it was conceptually applicable to SMIP with general mixed-integer recourse decisions, the algorithm was not easily realizable because it required calculations involving MIP value functions. Subsequently, most authors addressed some sub-class of these problems. For instance, the global optimization algorithm

of [ATS04] assumed fixed tenders (i.e. deterministic technology matrix in subproblems). Others have addressed other sub-classes which either focus on binary mixed-integer recourse decisions, or pure integer recourse decisions. Back in 1993, [LvdV93] first utilize the simple recourse structure to solve SMIP with continuous variables in the first stage and general integer variables in the second stage. Then decomposition-based cutting plane algorithms step up. For example, the disjunctive decomposition(D^2) in [SH05] for pure binary first stage and mixed-binary second stage, decomposition based branch-and-bound using RLT cuts in [SZ06] for mixed-binary in both stages. Novel branch-and-bound methods have been proposed in [EGMP07] for mixed-binary in the first stage and mixed-binary in second stage, decomposition-based branch-and-cut ([SS06]) for pure binary first stage and mixed-integer second stage, and search-based approaches using certain IP value function characterizations for pure integer second stage ([KSH06], [TPS12]). As a consequence of the sharper focus, these algorithms have made significant progress with SMIP algorithms for their specialized classes of SMIP. To better illustrate the scope of these algorithms, a table (shown in Table 1.1) summarizes the problem structure that each algorithm can handle. Here column B means the set of stages in which the algorithm allows having binary variables and column C denotes the set of stages with continuous variables and column D is the set of stages for discrete(general integer) variables. Columns T, W, g and r present whether randomness is allowed in the corresponding part of the second stage problem.

The major road-block for developing a time-staged decomposition algorithm for SMIP may be attributed to the fact that the value function of a mixed-integer linear program (MILP-G) is non-convex and discontinuous. Moreover, simply evaluating the second-stage value requires the solution of multiple NP-hard problems. We will overcome the first hurdle by approximating the set of second-stage MIP feasible solutions by polyhedra that depend

Algorithm	B	C	D	T	W	g	r
[LvdV93]	$\{2\}$	$\{1\}$	$\{2\}$	fixed	fixed[1]	fixed	random
[LL93]	$\{1,2\}$	$\{2\}$	$\{2\}$	random	fixed	random	random
[CS98]	$\{1,2\}$	$\{1,2\}$	$\{1,2\}$	random	random	random	random
[ATS04]	$\{1,2\}$	$\{1\}$	$\{2\}$	fixed	random	random	random
[SH05]	$\{1,2\}$	$\{2\}$	$\{\}$	random	fixed	fixed	random
[SS06]	$\{1,2\}$	$\{2\}$	$\emptyset$	random	fixed	fixed	random
[SZ06]	$\{1,2\}$	$\{1,2\}$	$\emptyset$	random	random	fixed	random
[KSH06]:	$\{1,2\}$	$\{\}$	$\{1,2\}$	fixed	fixed	fixed	random
[NS07]:	$\{2\}$	$\{1,2\}$	$\emptyset$	random	fixed	random	random
[EGMP07]:	$\{1\}$	$\{1,2\}$	$\emptyset$	random	random	fixed	random
Algorithm in Chapter 2:	$\{1,2\}$	$\{1,2\}$	$\{1,2\}$	random	random	random	random

Table 1.1: Summary of Problem Structure allowed for Decomposition Algorithms in the Literature

parametrically on the first stage solution (x). The second hurdle (solving multiple NP-hard problems in each iteration) will be overcome by obtaining a sequence of approximations that are optimal as iterations proceed, but not necessarily optimal for every iteration. In order to accomplish these goals, we will rely heavily on multi-term disjunctive cuts, as well as local Benders' cuts.

For mixed-binary linear programs (MILP-B), disjunctive and lift-and-project cuts ([Bal79]; [BCC93]), semidefinite relaxations ([LS91]) and reformulation-linearization technique (RLT) ([SA90],[She94]) have provided alternative approaches to generate cutting planes that define the convex hull of feasible points of a MILP-B. For pure integer programs, we can have a finite representation using Gomory cuts ([Gom63]). When it comes to the general mixed integer, [AS05] provided a generalization of the RLT methodology to the case of MILP-Gs using Lagrange Interpolation Polynomials to compute the bound factors in the RLT process. Recently, [CKS11] uses a cutting plane tree to manage partitions needed to generate multi-term disjunctive cuts and in [CKS12], computational experiment is conducted for this algorithm with different normalization in the CGLP.

[1]simple recourse structure

We should mention that there have been some efforts to speed up the solution of CGLP. In [PB01a], the authors proposed methods using simplex tableau methods to solve two-term CGLP. In [BP02], the authors proposed four methods to solve multi-term CGLP and they suggest Benders' decomposition algorithm is faster over the other three methods. But the experiments reported in [BP02] were not intended to test the effectiveness of multi-term disjunctions in solving MILP-G instances.

In this dissertation, we study a new finitely convergent decomposition algorithm to solve SMIP with general integer variables in two stages. We investigate different ways to obtain value function approximation for subproblems (a MILP with general integer variables) using multi-term disjunctive cuts.

1.3 Problem Statement

In this dissertation, we consider two stage SMIP problem with bounded mixed-integer variables in both stages. Assuming relatively complete recourse and there are finitely many scenarios, this dissertation addresses following questions.

- Is there a convergent time-staged decomposition algorithm for solving SMIP with general integer variables?
- Is it possible to obtain value function approximation from partial or fully optimized MILP using Branch and Bound (B&B) method? How does it compare with value function approximation using cutting plane tree method?
- Are there ways to enhance the computational performance of CPT algorithm? How does the multi-term disjunctive cut compare with the standard two-term disjunctive cut?

1.4 Organization of the Dissertation

The dissertation contains five chapters. The organization of the rest of this dissertation is as follows. In Chapter 2, the decomposition algorithm for solving two stage SMIP with general integer variables in both stages is presented. A convergence proof is included and that is followed by illustrative examples. In Chapter 3, methods to obtain a polyhedral approximation of MILP are discussed and computational experiments are conducted to compare their performances in different stochastic programming settings. In Chapter 4, we compare various variable splitting rules as well as ways to implement the CPT algorithm. We report our results using multiple variants of the CPT algorithm. Finally, a summary of the research contribution and future research direction along the line of work is shown in Chapter 5.

Chapter 2: Multi-term Disjunctive Decomposition for Mixed-Integer Recourse Decisions in Stochastic Programming

2.1 Introduction

Stochastic mixed-integer programs (SMIPs) have long been recognized as an important class of models for many practical operational problems (see e.g. [Wol80]). However, algorithmic advances to solve SMIP models have lagged other forms of stochastic programs (SP). In addition to the standard difficulties associated with stochastic linear programming (e.g. designing scalable ways to approximate the expected recourse/value function), SMIP formulations with mixed-integer recourse decisions in the second stage encounter value functions that are possibly non-convex and discontinuous. For this reason, certain decomposition algorithms for SMIP models were designed as scenario decomposition methods (e.g. [CT98], [LS04]) which were designed to take advantage of special structured problems associated with each scenario. For instance, [CT98] use the structure of unit-commitment models, whereas, [LS04] use the structure of batch-sizing models. However, many practitioners (e.g. O'Neill et al) have made a strong case for avoiding the dependence of decomposition algorithms on highly specialized scenario problems because one either loses the algorithmic advantages once the special-structure is violated, or, the user loses flexibility in modeling situations that were

not considered at the inception of the algorithm. For these reasons, time-staged decomposition (e.g. Benders' decomposition) may be preferable because, although, the non-convexity of feasible sets, and the attendant non-convexity or discontinuity of value functions poses algorithmic challenges. In this chapter, we will study time-staged decomposition methods that allow significantly more structures than available for SMIP today.

We are interested in designing time-staged decomposition algorithms for solving two-stage SMIPs in which mixed-integer decisions appear in both stages. In other words, we return to the class of models addressed in [CT98]. Fortunately, due to significant algorithmic advances in the interim, we are able to draw upon new approximations that will not only ensure finite convergence, but also avoid operations with intractable functions (such as the MIP value function). The SMIP problem formulation is stated in (2.1) and (2.2).

$$\min_{x \in X \cap Q_1} c^{\mathsf{T}} x + \mathsf{E}[f(x, \tilde{\omega})] \tag{2.1}$$

where

$$X = \{x \mid Ax \leq b, x_i \geq 0, \forall i \in I_1 = \{1...n_1\}$$

$$x_i \text{ is integer}, \forall i \in I_2 \subseteq I_1\}$$

$$Q_1 = \{x \mid l_1 \leq x \leq u_1\}$$

Here objective coefficient $c \in \mathbb{R}^{n_1}$, constraint matrix $A \in \mathbb{R}^{m_1 \times n_1}$ and right hand side $b \in \mathbb{R}^{m_1}$. Also $\tilde{\omega}$ denotes a random variable, and for each scenario (realization) ω of $\tilde{\omega}$, we define the recourse function by

$$\begin{aligned} f(x, \omega) = \min \quad & g(\omega)^{\top} y \\ \text{s.t.} \quad & W(\omega) y \geq r(\omega) - T(\omega) x \\ & y \in Y \cap Q_2 \end{aligned} \tag{2.2}$$

where

$$Y = \{y \mid y_j \geq 0, \forall j \in J_1 = \{1...n_2\}$$
$$y_j \text{ is integer}, \forall j \in J_2 \subseteq J_1\} \subseteq \mathcal{R}^{n_2}$$
$$Q_2 = \{y \mid l_2 \leq y \leq u_2\}$$

Both X and Y are assumed to be non-empty mixed-integer sets. We assume variables x and y are bounded by constraints defining Q_1 and Q_2. For subproblem (2.2), the decision variable $y \in \mathbb{R}^{n_2}$, objective coefficient $g(\omega) \in \mathbb{R}^{n_2}$, constraint matrix $W(\omega) \in \mathbb{R}^{m_2 \times n_2}$, $r(\omega) \in \mathbb{R}^{m_2}$ and $T(\omega) \in \mathbb{R}^{m_2 \times n_1}$. We assume the random variable $\tilde{\omega}$ can be discretized with each scenario ω having non-zero probabilities $p(\omega), \forall \omega \in \Omega$. The algorithm that we propose below will impose the following assumptions on the model.

A1 The integer variables in both stages are bounded.

A2 The random variable $\tilde{\omega}$ in the problem can be discretized into a finite number of scenarios, $\omega \in \Omega$, each with an associated probability of occurrence $p(\omega), \forall \omega \in \Omega$.

A3 For any $x \in X \cap Q_1$, set defined by $\{y | W(\omega)y \geq r(\omega) - T(\omega)x, y \in Y \cap Q_2\}$ is feasible for all $x \in X \cap Q_1$ and all $\omega \in \Omega$.

Due to assumption A2, (2.1) can be rewritten as:

$$\min_{x \in X \cap Q_1} c^{\mathsf{T}} x + \sum_{\omega \in \Omega} p(\omega) f(x, \omega) \tag{2.3}$$

For the rest of the chapter, we first introduce the overall architecture of multi-term disjunctive decomposition algorithms. Subsequently, we present two closely related approximation methods, one based on the cutting plane tree method ([CKS12]), and another we refer to as branch-and-bound (B&B)-based convexification method. Either of these methods can be used to convexify the second-stage, which in turn leads to a value function approximation

for $f(x, \omega)$. Finally, several variants of an SMIP example in the literature illustrates the applicability of the method under different settings (or structures) for SMIP models. In addition we present preliminary evidence that this approach is more scalable than using the deterministic equivalent problem (3) on a standard commercial MIP solver. Overall, our approach provides the most comprehensive time-staged decomposition approach to date.

2.2 Multi-term Disjunctive Decomposition

If we denote decision variables y under scenario ω as $y(\omega)$, the deterministic equivalent problem (DEP) formulation for (2.3) is

$$\min_{x,\{y(\omega)\}} \quad c^\top x + \sum_{\omega\in\Omega} p(\omega) \sum g(\omega)^\top y(\omega) \tag{2.4a}$$

$$\text{s.t.} \quad T(\omega)x + W(\omega)y(\omega) \geq r(\omega), \forall \omega \in \Omega \tag{2.4b}$$

$$x \in X \cap Q_1, y(\omega) \in Y \cap Q_2 \tag{2.4c}$$

When x are restricted to pure binary variables, [SF02] showed that solving problem (2.4) is equivalent to solving (2.5) provided $x = \bar{x}$ is facial with respect to Q_1.

$$\min_{x\in X} \quad c^\top x + \sum_{\omega\in\Omega} p(\omega) \sum g(\omega)^\top y(\omega) \tag{2.5a}$$

$$\text{s.t.} \quad (x, \{y(\omega)\}) \in \mathbf{conv}\{(x, \{y(\omega)\}) | T(\omega)x + W(\omega)y(\omega) \geq r(\omega),$$

$$x \in Q_1, y \in Y \cap Q_2\}, \forall \omega \in \Omega \tag{2.5b}$$

$$x \in X \cap Q_1, \tag{2.5c}$$

When the vector x is allowed to be continuous or has some elements that are restricted to be general integers, $x = \bar{x}$ may not always be facial with respect to Q_1 and as a result, pure cutting plane algorithms may not be able to solve the general SMIP model (2.4). Instead, we

plan to design an algorithm which implements a successive Benders' decomposition scheme associated with nodes of a first-stage search tree. In order to accomplish this, our approach combines three basic ingredients.

a) We use B&B in the first stage to ultimately enforce the facial property (as in [SZ06]).

b) We use ideas from disjunctive decomposition to ensure a finite number of successive approximations (as in [SH05]).

c) We use multi-term disjunctions to approximate the convex hull of mixed-integer sets (as in [CKS11]).

Let $Q_1^{t(k)}$ denote the bounding constraint of node $t(k)$ in the B&B tree in the first stage such that the k^{th} first-stage iterate $x^k \in Q_1^{t(k)}$, and $Q_1^{t(k)} \subseteq Q_1$. Then define

$$\mathcal{Y}(Q_1^{t(k)}, \omega) \equiv \{(x, y(\omega)) \mid T(\omega)x + W(\omega)y \geq r(\omega), x \in X \cap Q_1^{t(k)}, y \in Y \cap Q_2\}. \tag{2.6}$$

Also define $\mathcal{Y}_L(Q_1^{t(k)}, \omega)$ as a convex polyhedron such that

$$\mathcal{Y}_L(Q_1^{t(k)}, \omega) \supseteq \mathbf{conv}\{\mathcal{Y}(Q_1^{t(k)}, \omega)\} \tag{2.7}$$

While the convex hull operation above is well defined, obtaining $\mathbf{conv}\{\mathcal{Y}(Q_1^{t(k)}, \omega)\}$ is an onerous task. Instead, the cutting plane tree (CPT) provides a constructive mechanism to obtain $\mathcal{Y}_L(Q_1^{t(k)}, \omega)$ and the resulting $\mathcal{Y}_L(Q_1^{t(k)}, \omega)$ satisfies $f(x^k, \omega) = f_L^k(x^k, \omega)$ for fixed $x^k \in \mathbf{vert}(X \cap Q_1^{t(k)})$ where $f_L^k(x, \omega)$ is defined as

$$f_L^k(x, \omega) = \min\{g(\omega)^\top y : \ (x, y) \in \mathcal{Y}_L(Q_1^{t(k)}, \omega)\} \tag{2.8}$$

We will exploit this new capability for the class of algorithms studied in this chapter. Thus appealing to the CPT algorithm ([CKS11]), suppose that we are able to obtain a convex

representation as follows

$$\mathcal{Y}_L(Q_1^{t(k)},\omega) = \{(x,y) \mid T(\omega)x+W(\omega)y \geq r(\omega), \Gamma^k(\omega)x+\Pi^k(\omega)y \geq \Pi_0^k(\omega), x \in X \cap Q_1^{t(k)}, y \geq 0\}. \tag{2.9}$$

where $\Gamma^k(\omega)x + \Pi^k(\omega)y \geq \Pi_0^k(\omega)$ are the set of cuts added during iterations $1,\ldots,k$.

Proposition 1. *For any $\omega \in \Omega$ and fixed $x = \bar{x} \in X \cap Q_1^{t(k)}$, we have $f(x,\omega) \geq f_L^k(x,\omega)$. Furthermore, if $\bar{x} \in \mathbf{vert}(X \cap Q_1^{t(k)})$, where $\mathbf{vert}(X \cap Q_1^{t(k)})$ denote vertices of $X \cap Q_1^{t(k)}$, we have $f(x,\omega) = f_L^k(x,\omega)$.*

Proof. Since $\mathcal{Y}_L(Q_1^{t(k)},\omega) \supseteq \mathcal{Y}(Q_1^{t(k)},\omega)$, we have $f(x,\omega) \geq f_L^k(x,\omega)$. For fixed $\bar{x} \in \mathbf{vert}(X \cap Q_1^{t(k)})$, no matter whether $\bar{x}$ is integer or binary, as long as $\bar{x} \in \mathbf{vert}(X \cap Q_1^{t(k)})$, the restriction $x = \bar{x}$ is facial respect to $\mathcal{Y}_L(Q_1^{t(k)},\omega))$. Hence following ([SF02]), we have $f(x,\omega) = f_L^k(x,\omega)$. □

Proposition 1 helps us derive the decomposition algorithm for a general SMIP problem. For fixed x^k and $Q_1^{t(k)}$, the second-stage convexification process will yield the set $\mathcal{Y}_L(Q_1^{t(k)},\omega)$. Let the matrix $W^k(\omega)$ denote the entire collection of cuts generated for outcome ω. Given any subset $Q_1^{t(k)}$, not all cuts listed in $W^k(\omega)$ are valid. Let σ denote a subset of all cuts that are valid for $Q_1^{t(k)}$. Of course, σ depends on $t(k)$, but in the interest of notational brevity, we will use $W_\sigma^k(\omega)$ as the submatrix of $W_k(\omega)$ to denote the cuts that are valid for $Q_1^{t(k)}$ and $\sigma \in \Sigma(\omega, t(k))$ Then, and a Benders'-type lower bounding subproblem to approximate the integer recourse (or value) function may be obtained by solving the following problem.

$$f_L^k(x^k,\omega) = \min \quad \sum g(\omega)^\top y \tag{2.10a}$$

$$\text{s.t.} \quad W_\sigma^k(\omega)y \geq r_\sigma^k(\omega) - T_\sigma^k(\omega)x^k \tag{2.10b}$$

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

where $W^k_\sigma(\omega) = \begin{bmatrix} W(\omega) \\ \Pi^k_\sigma(\omega) \end{bmatrix}$, $r^k_\sigma(\omega) = \begin{bmatrix} r(\omega) \\ \Pi^k_{0\sigma}(\omega) \end{bmatrix}$, $T^k_\sigma(\omega) = \begin{bmatrix} T(\omega) \\ \Gamma^k_\sigma(\omega) \end{bmatrix}$. When the branch and bound (B&B) method is used to solve master problem, x^k may only be a vertex for local node $t(k)$. So it is important to note that cuts derived for one node with bounds Q^t_1 may not be valid for other nodes with bounds $Q'_1 \subsetneq Q^t_1$. On the other hand, the cuts generated using Q^t_1 can be applied to all subsets $Q'_1 \subseteq Q^t_1$. Let s denote the index of the Benders' cut generated for Q^t_1 in iteration k. As with cuts in the second-stage, the index s also depends on $t(k)$. However, letting $S(t(k))$ denote the entire collection of Benders' cuts for $Q^{t(k)}_1$, the most recent cut will be given as

$$\eta_{t(k)}(\omega) \geq \Theta^k(\omega)^\top (r^k_\sigma(\omega) - T^k_\sigma(\omega)x) \tag{2.11}$$

where $\Theta^k(\omega)$ is the vector of optimal dual multipliers associated with (2.10b). (2.11) can be derived for each $\omega \in \Omega$ in parallel. Such a Benders' cut for the master problem can be written as

$$\eta_{t(k)} \geq \xi_s - \zeta_s^\top x \tag{2.12}$$

where

$$\begin{aligned} \xi_s &= \sum_{\omega\in\Omega} p(\omega)\Theta^k(\omega)^\top r^k \sigma(\omega) \\ \zeta_s &= \sum_{\omega\in\Omega} p(\omega)\Theta^k(\omega)^\top T^k_\sigma(\omega). \end{aligned} \tag{2.13}$$

The process of deriving (2.12) is considered as approximating the subproblem value function for a given $x^k \in Q^{t(k)}_1$, and for all ω. Note that both (2.10) and (2.12) are valid only for $x \in Q^{t(k)}_1$.

The master problem is solved by a B&B method. Suppose the set of active (non-fathomed) nodes for 1st stage is denoted as $\mathcal{T}^k_1$, and let Q^t_1 denote the bounding constraints for $t \in \mathcal{T}_1$. Let $S^k(t)$ denote a collection of Benders' cuts generated from subproblems for

$x \in Q_1^t$ prior to (and including) iteration k. Then the lower bounding master problem for node t at iteration k is as follows.

$$\min \quad c^\top x + \eta_t \tag{2.14a}$$

$$\text{s.t.} \quad \eta_t \geq \xi_s - \zeta_s x, \forall s \in S^k(t) \tag{2.14b}$$

$$\eta_t \geq -M, \tag{2.14c}$$

$$x \in X_L \cap Q_1^t, \tag{2.14d}$$

where $-M$ is a valid lower bound on second stage value function, and

$$X_L = \{x \mid Ax \leq b, x_i \geq 0\} \subseteq \mathcal{R}^{n_1} \tag{2.15}$$

The entire algorithm starts from iteration $k = 1$ with $\mathcal{T}_1^k = \{o\}$ where the root node o has bounding constraint $Q_1^o \equiv Q_1$ and we initialize $S^k(t)$ as an empty set. At iteration k, a B&B method is used to solve the master problem (2.14) until mixed-integer optimum is found. In particular, problem (2.14) is solved for each $t \in \mathcal{T}_1^k$ and for our breadth-first approach, we choose the node with the lowest objective value as the node to branch on. Suppose this node is $\bar{t}$. Following a variable selection rule such as choosing the fractional variable with smallest index or the most relative fractional variable as shown in (2.16) and (2.17),

$$\theta_i = \min\{x_i^t - l_{1i}^t, u_{1i}^t - x_i^t\} \tag{2.16}$$

$$p \in \text{argmax}_{i=1,\ldots,n}\{\frac{\theta_i}{u_{1i}^t - l_{1i}^t}\} \tag{2.17}$$

we can select variable x_p and split its bounding constraint $[l_{1p}, u_{1p}]$ into

$$[l_{1p}, \lfloor x_p \rfloor] \text{ and } [\lceil x_p \rceil, u_{1p}], \text{ if } p \in I_2 \tag{2.18}$$

The Benders' cut inherited from $S^k(\bar{t})$ are now associated with two new nodes. Again the node with lowest objective value among $\mathcal{T}_1^k$ is selected and this process repeats until the mixed-integer optimum is found.

When the master problem's solution is found (denoted as x^k), we identify the first stage node to which x^k belongs, and refer to it as $Q_1^{t(k)}$. Given x^k and $Q_1^{t(k)}$, we approximate the second stage value function for all ω as described in (2.12). With value function indexed by the cuts in $S^k(t(k))$ updated, the master problem continues on finding new integer solutions and updating the node's value function until the node is fathomed, or the algorithm stops. A summary of the process is shown in Algorithm 1, and we refer to it as **M-D²** (for multi-term disjunctive decomposition).

Proposition 2. *Assuming the process to derive (2.12) is a finite procedure and can obtain $f(x,\omega) = f_L(x^k,\omega)$ for each $\omega \in \Omega$, then algorithm **M-D²** (run with $\epsilon = 0$) terminates finitely with the incumbent solution x^* being optimal to problem (2.1),(2.2).*

Proof. (2.1) are assumed to have bounded variables, thus there are only finitely many nodes for the $B\&B$ tree to enumerate. In the worst case, all possible combination of values for variables in I_2 are enumerated which take finitely many steps. If deriving (2.12) is a finite process and for each $\omega \in \Omega$ we have $f(x^k,\omega) = f_L(x^k,\omega)$, from Proposition 1. At least, the Benders' cut derived is precise on x^k. As long as there are finite scenarios in Ω, it takes finite steps for a decomposition algorithm using Benders' cut to converge. Thus, it takes finitely many steps for the algorithm to evaluate each node in the B&B tree. So algorithm **M-D²** terminates finitely. □

Algorithm 1 M-D^2

Initialize: Set iteration $k = 1$, objective value upper bound $V = \infty$, lower bound $v = -\infty$, first stage active nodes $\mathcal{T}_1^k = \{o\}$ with $Q_1^o \equiv \{x | l_1 \leq x \leq u_1\}$. Set $S^k(o) = \emptyset$. Let ϵ denote the stopping tolerance and $(x^*, y^*(\omega))$ for $\omega \in \Omega$ the incumbent solutions. Solve problem (2.14) at node o and get its objective value v^o

while true **do**

Update v and denote the node with the lowest objective value as $t(k)$, and update the lower bound:

$$v \leftarrow \min_{t \in \mathcal{T}_1^k} \{v^t\}.$$

if $V - v \leq \epsilon$ **then STOP**.

end if

if $x_i^{t(k)}$ for $i \in I_2$ are integers **then**

Derive (2.12) and update $S^k(t(k))$.

Update V and incumbent $(x^*, y^*(\omega))$ if $y^k(\omega)$ satisfies mixed-integer restrictions for all $\omega \in \Omega$.

Update v^t for $t \in \mathcal{T}_1^k$ by re-solving problem (2.14) with $S^k(t)$ updated.

else

Choose variables to split and replace node $t(k)$ with two new nodes $t^1(k)$,$t^2(k)$.

Solve problem (2.14) for the two new node and obtain $v^{t^1(k)}$ and $v^{t^2(k)}$.

end if

Fathom nodes for which $v^t \geq V$, the upper bound V: $\mathcal{T}_1^{k+1} \leftarrow \mathcal{T}_1^k \setminus \{t \mid t \in \mathcal{T}_1^k, V - v^t \geq \epsilon\}$.

$k \leftarrow k + 1$

end while

2.3 Successive value function approximations

In order to ensure that the decomposition algorithm is effective, objective function approximations generated in one iteration should be made available in subsequent iterations. This philosophy was adopted in designing the D^2 algorithm ([SH05]) where disjunctive cuts defining the convex approximation of $Y \cap Q_2$ were made reusable for subsequent iterations by convexifying the function defining the right hand side of the cuts as a relatively simple function of the first stage decision x. The promising results from the D^2 algorithm ([YS09]) for binary SMIPs motivates us to extend that algorithm into more general cases considered in this chapter.

2.3.1 Cutting plane tree algorithm

It has been shown that only using traditional two-term disjunctive cuts ([BCC93]) to solve MILP may lead to infinite steps ([Bal79], [OM01]). One way to overcome this is to use the multi-term disjunctive cuts. It has been proved in [CKS11] to use multi-term disjunctive cuts (also called CPT cuts) to solve general MILP problems in finitely many iterations. We can apply this algorithm to solve subproblems (2.2). Similar to the D^2 algorithm, we will create the CPT cuts in a manner that they will be re-usable for different x.

Suppose that at the k-th iteration of **M-D^2**, we have a fixed $x^k \in X \cap Q_1^{t(k)}$, and to approximate $f(x^k, \omega)$ we execute the cutting plane tree (CPT) algorithm for a few iterations. To initialize a sequence of subproblem approximations of the CPT algorithm, we start with the subproblem LP relaxation $f_L(x, \omega)$ (as shown in (2.19)) is solved with $x = x^k$. In general, this relaxation will not yield a mixed-integer feasible point, and we will build approximations of the mixed-integer set using the CPT cuts which will delete non-integer solutions as and when we encounter them. For iteration d of the CPT algorithm, suppose that a non-integer

solution is denoted $y^d(\omega)$, where d is the iteration number for the subproblem algorithm.

$$\begin{aligned} f_L(x,\omega) = \min \quad & g(\omega)^\top y \\ \text{s.t.} \quad & W(\omega)y \geq r(\omega) - T(\omega)x \\ & y \in Y_L \cap Q_2 \\ & Y_L = \{y \geq 0, y \in \mathcal{R}^{n_2}\} \end{aligned} \tag{2.19}$$

Let $\mathcal{T}_2^d$ denote an index set of sets that partition Q_2. Q_2^t is a bounding constraint for the variable y as shown in (2.20).

$$Q_2^t = \{x | l_2^t \leq y \leq u_2^t\} \text{ for } \forall t \in \mathcal{T}_2^d \tag{2.20}$$

$\mathcal{T}_2^d$ provides a disjunctive description of $Y \cap Q_2$. And condition (2.21) is maintained

$$\bigcup_{t \in \mathcal{T}_2^k} (Y_L \bigcap Q_2^t) \supseteq Y \cap Q_2 \tag{2.21}$$

To cut off $(x^k, y^d(\omega))$, a disjunctive set is constructed such that (2.22) is satisfied.

$$(x^k, y^d(\omega)) \notin \bigcup_{t \in \mathcal{T}_2^d} \{(x, y(\omega)) \mid x \in X \cap Q_1^{t(k)}, y(\omega) \in Y_L \cap Q_2^t\} \tag{2.22}$$

As shown in [CKS12], $\mathcal{T}_2^d$ is maintained by a tree structure (called cutting plane tree). $\mathcal{T}_2^d$ contains all the nodes that do not have children nodes and is initialized with one node with defining constraint Q_2. If the solution $y^d(\omega)$ does not satisfy the mixed-integer restrictions, then the algorithm walks through cutting plane tree to locate the deepest node from the root node that contains $y^d(\omega)$. If this node is in $\mathcal{T}_2^d$ (meaning it does not have any children node), two nodes are created as its children nodes and the node is removed from $\mathcal{T}_2^d$. If the node is not in $\mathcal{T}_2^d$, no new node is created. In this way, $\mathcal{T}_2^d$ is updated such that conditions (2.21) and (2.22) are satisfied (see also Algorithm 2).

Assuming that (2.22) is satisfied, we will use a cut generation linear program (CGLP) to derive a valid inequality. While the specific form of the CGLP is not critical to the

proof of convergence f the methodology, a linear programming structure of the CGLP is important. The form of CGLP shown in (2.23) maximizes the depth of cut while restricting the one-norm of the coefficients to be 1 (2.23e). This version is similar to the CGLP used by ([CKS12]), where the right-hand-side was fixed to be 1, and the one-norm of cut coefficients was minimized.

Using (2.20), the cut generating linear program (CGLP) shown in (2.23) can be formulated to derive multi-term disjunctive cuts.

$$\max \quad \pi_0(\omega) \tag{2.23a}$$

$$\text{s.t.} \quad \pi_{2j}(\omega) = W_j(\omega)^\top \lambda_{2t} + \mu_{2jt} - \nu_{2jt} \quad \forall j \in J, t \in \mathcal{T}_2^d \tag{2.23b}$$

$$\pi_{1i}(\omega) = T_i(\omega)^\top \lambda_{2t} + A_i{}^\top \lambda_1 + \mu_{1i} - \nu_{1i} \quad \forall i \in I, t \in \mathcal{T}_2^d \tag{2.23c}$$

$$r(\omega)^\top \lambda_{2t} + b^\top \lambda_1 + l_2^t{}^\top \mu_{2t} + l_1^{t(k)\top} \mu_1 - u_2^t{}^\top \nu_{2t} - u_1^{t(k)\top} \nu_1$$
$$\geq \pi_1(\omega)^\top x^k + \pi_2(\omega)^\top y^d(\omega) + \pi_0(\omega) \quad t \in \mathcal{T}_2^d \tag{2.23d}$$

$$\sum_{i \in I} \alpha_{1i} + \sum_{j \in J} \alpha_{2j} = 1 \tag{2.23e}$$

$$-\alpha_1 \leq \pi_1(\omega) \leq \alpha_1, -\alpha_2 \leq \pi_2(\omega) \leq \alpha_2 \tag{2.23f}$$

$$\alpha_1, \alpha_2 \geq 0, \lambda_1, \lambda_{2t} \geq 0, \nu_1, \nu_{2t} \geq 0, \mu_1, \mu_{2t} \geq 0, \text{for } \forall\ t \in \mathcal{T}_2^d \tag{2.23g}$$

The cut derived is shown in (2.24).

$$\pi_1(\omega)^\top (x - x^k) + \pi_2(\omega)^\top (y(\omega) - y^d(\omega)) \geq \pi_0(\omega) \tag{2.24}$$

Note that (2.24) is specific to scenario ω. We can derive a cut for each scenario $\omega \in \Omega$. Since (2.23) includes inequalities that are valid for $X \cap Q_1^{t(k)}$, this CGLP combines the convexification of x and y simultaneously.

Proposition 3. *Cutting plane (2.24) is valid for feasible set* $\{(x, y(\omega)) \mid T(\omega)x + W(\omega)y(\omega) \geq r(\omega), x \in X \cap Q_1^{t(k)}, y(\omega) \in Y \cap Q_2\}$.

Proof. To show that cut (2.24) is valid, it is equivalent to show that $\pi_1(\omega)^\top x + \pi_2(\omega)^\top y \geq \pi_1(\omega)^\top x^k + \pi_2(\omega)^\top y^d(\omega) + \pi_0(\omega)$ is valid. For any $\{(x, y(\omega)) \mid T(\omega)x + W(\omega)y(\omega) \geq r(\omega), x \in X \cap Q_1^{t(k)}, y(\omega) \in Y \cap Q_2\}$,

$$\pi_1^\top x + \pi_2^\top y(\omega) \tag{2.25a}$$

$$= (\lambda_{2t}^\top T(\omega)x + \lambda_1^\top Ax + \mu_1^\top x - \nu_1^\top x) + (\lambda_{2t}^\top W(\omega)y(\omega) + \mu_{2t}^\top y(\omega) - \nu_{2t}^\top y(\omega)) \tag{2.25b}$$

$$\geq r(\omega)^\top \lambda_{2t} + b^\top \lambda_1 + {l_2^t}^\top \mu_{2t} + {l_1^{t(k)}}^\top \mu_1 - {u_2^t}^\top \nu_{2t} - {u_1^{t(k)}}^\top \nu_1 \tag{2.25c}$$

$$\geq \pi_1(\omega)^\top x^k + \pi_2(\omega)^\top y^d(\omega) + \pi_0(\omega) \tag{2.25d}$$

□

The cut (2.24) is included in the second-stage formulation, leading to a stronger approximation of the value function, which will be denoted $f_L^d(x, \omega)$. In general, after d iterations of the CPT algorithm for the subproblem, we have

$$f_L^d(x, \omega) = \min \quad g(\omega)^\mathsf{T} y \tag{2.26a}$$

$$\text{s.t.} \quad W(\omega)y \geq r(\omega) - T(\omega)x \tag{2.26b}$$

$$\Pi_2^d(\omega)y \geq \Pi_0^d(\omega) - \Pi_1^d(\omega)x \tag{2.26c}$$

$$y \in Y_L \cap Q_2 \tag{2.26d}$$

$$Y_L = \{y \geq 0, y \in \mathcal{R}^{n_2}\} \tag{2.26e}$$

where constraints (2.26c) accumulate cuts generated by ancestor nodes, as well as solutions in $Q_1^{t(k)}$. Cuts generated from first-stage solutions that do not belong to $Q_1^{t(k)}$ or one of its ancestors are not included in (2.26c). d is the number of cuts added since the approximation algorithm starts. At any iteration, we allow at most D number of cuts to be added for each

run of second-stage approximation algorithm. After the new cut is added, the strengthened LP is re-solved, and the approximation $f_L^d(x,\omega)$ is obtained. The algorithm continues until either a mixed-integer optimum is found or $d = D$. Once the process of solving the subproblem stops, we form a Benders' cut as in (2.11) and return to the master problem.

Since cuts (2.24) are only valid in the $(x, y(\omega)) \in (X_L \cap Q_1^{t(k)}) \times (\mathcal{Y}_L \cap Q_2)$ space, they could be reused at any subsequent iteration $k+\tau$, $\tau \geq 1$ as long as $x^{k+\tau}$ satisfies $x^{k+\tau} \in X \cap Q_1^{t(k)}$. Recall that we let $\Sigma(\omega, t(k))$ denote the collection of cuts generated from node t's ancestor or itself. It is used to populate constraints (2.26c) at $d = 1$. Once optimal solution is found, $\Sigma(\omega, t(k))$ is updated to incorporate newly generated constraints.

Given $x^k \in X \cap Q_1^{t(k)}$, the algorithm to solve subproblems ω with at most D cuts added is shown as Algorithm-2.

Algorithm 2 CPT-D

Initialize $d \leftarrow 1$, set CPT tree leaves set $\mathcal{T}_2^d(\omega) \leftarrow \{o\}$ where o is the root node with bounding constraint $Q_2^o \leftarrow Q_2$.
while $d \leq D$ **do**
  Populate $\Pi_1^d(\omega)$, $\Pi_2^d(\omega)$, $\Pi_0^d(\omega)$ using cuts in $S(\omega, t)$. Solve $f_L^d(x,\omega)$ and get $y^d(\omega)$.
  if $y^d(\omega)$ satisfies mixed-integer restrictions **then**, STOP and $y^k(\omega) \leftarrow y^d(\omega)$
  else
    Find the deepest node σ that contains $y^d(\omega)$ in $\mathcal{T}_2^d$.
    if σ is not leaf node **then**,
      Make splits on σ and use updated $\mathcal{T}_2^d$ and first stage node bounds $Q_1^{t(k)}$ to formulate and solve (2.23) and obtain cut (2.24).
    else
      No splits are needed. Use σ and leaf nodes that don't belong to subtree of σ and $Q_1^{t(k)}$ to formulate and solve (2.23) and obtain cut (2.24).
    end if
    Add the cut into $\Sigma(\omega, t(k))$ and update $\Pi_1^d(\omega)$, $\Pi_2^d(\omega)$, $\Pi_0^d(\omega)$ with the new cut
  end if
  $d \leftarrow d + 1$
end while

Proposition 4. *Assume that the cuts generated from (2.23) correspond to its extreme point solutions. Then there exist finite $D < \infty$, such that algorithm **CPT-D** solves subproblems (2.2) at scenario ω to optimal with x fixed at x^k and also: $f_L^k(x^k, \omega) = f(x^k, \omega)$.*

Proof. When D is large enough, algorithm **CPT-D** is the same as using CPT algorithm to solve the subproblem except the CGLP is different. From the finiteness proof of CPT algorithm in [CKS11], to prove that algorithm **CPT-D** can obtain mixed-integer optimum in finitely many steps, we only need to show that for fixed $\mathcal{T}_2^k$ and $Q_1^{t(k)}$, only finitely many constraints can be generated out of the new CGLP (2.23) as $y^d(\omega)$ changes. To show that, we know the dual of CGLP has $y^d(\omega)$ only in the objective function. Since dual constraint set stays the same as $y^k(\omega)$ changes, there are only finitely many extreme points to enumerate, thus finitely many constraint to generate out of CGLP (2.23). Thus there exists iteration $D < \infty$ such that (2.2) is solved at scenario ω with x fixed at x^k. Since the mixed-integer solution is found, $f_L^k(x^k, \omega) = f(x^k, \omega)$. □

We use algorithm **CPT-D** to solve subproblem for each $\omega \in \Omega$ and derive constraint (2.12). In algorithm **M-D²** , we initialize $D \leftarrow 2$ and at each iteration, increment $D \leftarrow D+2$. Therefore, in the beginning, $f_L^k(x^k, \omega) = f(x^k, \omega)$ is not guaranteed. Still, we have $K < \infty$, such that for $k > K$, D is large enough such that all subproblem is solved and we will have $f_L^k(x^k, \omega) = f(x^k, \omega)$. Compared with D^2 algorithm, **CPT-D** algorithm requires a separate CPT tree for each scenario to manage the multi-term disjunctions. And the multi-term disjunctive cut derived in **CPT-D** algorithm is only valid for $x \in Q_1^{t(k)}$ and it is special for each ω. But in D^2 algorithm, the disjunctive cut works for all $\{x \mid 0 \leq x \leq 1\}$. Although the right hand side changes with different ω, the left hand side of D^2 cuts derived are fixed for all $\omega \in \Omega$.

2.3.2 Branch-and-Bound based Convexification

Branch and Bound (B& B) method [LD60] is a popular method for solving MILP problems and when combined with cutting planes, these methods form the backbone for most state-of-the-art commercial solvers. The CPT algorithm in previous section is a pure cutting plane algorithm. The fact that it also utilizes a tree structure to manage the disjunctive sets inspires a way that transforms the B&B tree obtained from an MILP solve to create a polyhedral approximation that gives the same IP optimal value as obtained from the B&B method. However, the potential size of the disjunctions encountered in B&B method makes it non-trivial to obtain a polyhedral approximation from it. For the remainder of this section, we describe such an algorithm and prove that this approximation can be obtained in finitely many steps.

Suppose for fixed $x^k \in X \cap Q_1^{t(k)}$, subproblem $f(x^k, \omega)$ is solved by a B&B method and its optimal solution is $y^k(\omega)$. Also we have the set of leaf nodes $\mathcal{T}_2 = \mathcal{T}_2^{\text{remain}} \cup \mathcal{T}_2^{\text{fathom}}$ where $\mathcal{T}_2^{\text{remain}}$ are the remaining leaf nodes in the B&B tree and $\mathcal{T}_2^{\text{fathom}}$ are the leaf nodes that have been fathomed. Suppose the constraint set for $f_L(x^k, \omega)$ is

$$\mathcal{Y}_L(x^k, \omega) = \{y(\omega) | W(\omega)y(\omega) \geq r(\omega) - T(\omega)x^k, y(\omega) \in Y_L \cap Q_2\} \tag{2.27}$$

and

$$\mathcal{Y}(x^k, \omega) = \mathcal{Y}_L(x^k, \omega) \cap Y. \tag{2.28}$$

Since $Q_2^t, \ \forall \ t \in \mathcal{T}_2$ are disjoint from each other, $\mathcal{T}_2$ provides us a disjunctive relaxation in the space of $(x, y(\omega))$

$$\{X \cap Q_1^{t(k)}\} \times \mathcal{Y}_D(x^k, \omega) = \{X \cap Q_1^{t(k)}\} \times \bigcup_{t \in \mathcal{T}_2} (\mathcal{Y}_L(x^k, \omega) \bigcap Q_2^t), \tag{2.29}$$

Thus, the same CGLP (2.23) can be used to derive multi-term disjunctive cuts by replacing $\mathcal{T}_2^k$ with $\mathcal{T}_2$ in the formulation. The rest of algorithm is similar to **CPT-D**. There are two

phase in the algorithm. The first phase is to use a B&B method to either solve the second-stage MIP, or to use a few nodes of the B&B tree to obtain an disjunctive approximation. In either case, we obtain $\mathcal{T}_2$. The second phase starts from solving $f_L^d(x^k, \omega)$ with $d = 1$. At iteration d, if the optimal solution of $f_L^d(x^k, \omega)$ is fractional (denoted as $y^d(\omega)$), (2.23) is formulated based on $\mathcal{T}_2$ and Q_1^t to cut off $y^d(\omega)$. The new cut is added into $S(t(k))$. Then $f_L^d(x^k, \omega)$ is re-solved and this process continues until $y^d(\omega)$ falls into $\mathcal{Y}_D$. The method is shown in Algorithm 3.

Algorithm 3 BB-D

Initialize iteration $d \leftarrow 1$. Populate $\Pi_1^d(\omega)$, $\Pi_2^d(\omega)$, $\Pi_0^d(\omega)$ using cuts in $S(t(k))$.
Solve subproblem: Solve $f(x^k, \omega)$ by B&B method for D iterations and get leaf nodes set $\mathcal{T}_2$ and solution $y^*(\omega)$.
while true **do**
  Populate $\Pi_1^d(\omega)$, $\Pi_2^d(\omega)$, $\Pi_0^d(\omega)$ using cuts in $S(\omega, t)$. Solve $f_L^d(x, \omega)$ and get $y^d(\omega)$.
  if $y^d(\omega) \in Y_D$ **then**, STOP and $y^k(\omega) \leftarrow y^d(\omega)$
  end if
  Use $\mathcal{T}_2$ and Q_1^t to formulate and solve (2.23) and obtain cut (2.24). Add the cut into $S(\omega, t)$ and update $\Pi_1^d(\omega)$, $\Pi_2^d(\omega)$, $\Pi_0^d(\omega)$ with the new cut
  $d \leftarrow d + 1$
end while

Define **clconv** as the operation to get closure of the convex hull of a set and the last iteration number of the algorithm is d_∞. Then we have the following theorem.

Theorem 1. *Algorithm 3 terminates in finitely many steps (i.e. $d_\infty < \infty$). And when the algorithm stops, we have $f(x^k, \omega) \geq f_L^{d_\infty}(x^k, \omega)$. If D is large enough to have integer optimal from B&B solve, then $f(x^k, \omega) = f_L^{d_\infty}(x^k, \omega)$*

Proof. Because $\mathcal{T}_2$ is fixed, as $y^d(\omega)$ changes, each time a new extreme point of the CGLP (2.23) is generated. Because there are finitely many extreme points of the CGLP and a

subset of these corresponds to facets of clconv$\{\mathcal{Y}_D(x^k,\omega)\}$. We will have $y^d(\omega) \in \mathcal{Y}_D(x^k,\omega)$ in finitely many iterations which is the stopping criterion. For the second part of the theorem, also from the algorithm's stopping rules and Proposition 4, there $\exists\ y^{d_\infty}(\omega)$ such that $y^{d_\infty}(\omega) \in \mathcal{Y}_D(x^k,\omega)$. If $y^{d_\infty}(\omega) = y^*(\omega)$, then obviously $f(x^k,\omega) \geq f_L^{d_\infty}(x^k,\omega)$. On the other hand, suppose that $y^{d_\infty}(\omega) \neq y^*(\omega)$. Since the B&B tree embodies the disjunction, it follows that $y^*(\omega) \in \mathcal{Y}_D(x^k,\omega)$. Hence $g(\omega)^\top y^{d_\infty}(\omega) \geq g(\omega)^\top y^*(\omega)$. But since every solution generated by the algorithm is based on a approximations strengthened by valid inequalities, it follows that $g(\omega)^\top y^{d_\infty}(\omega) \leq g(\omega)^\top y^*(\omega)$. It follows that $g(\omega)^\top y^{d_\infty}(\omega) = g(\omega)^\top y^*(\omega)$. While for relatively small D, when B&B method does not reach integer optimal, $f_L^{d_\infty}(x^k,\omega) = g(\omega)^\top y^{d_\infty}(\omega) = g(\omega)^\top y^*(\omega) \geq f(x^k,\omega)$. When D is large enough to have integer optimal, $f(x^k,\omega) = g(\omega)^\top y^*(\omega) = f_L^{d_\infty}(x^k,\omega)$

□

Algorithm 2 and Algorithm 3 both use multi-term disjunctive cuts to obtain value function approximation. The difference between the two algorithms is the way to construct disjunctive sets. Algorithm-3 uses the B&B nodes to construct the disjunctive set, whereas, Algorithm 2 iteratively builds up the disjunctive set.

2.4 Illustrative Examples and a Computational Prototype

We illustrate the workings of the algorithms of this chapter via a collection of examples. The following presentation is based on an example that has been used by several authors, under alternative problem structures (e.g. fixed recourse ([CS98], [ATS04]), binary recourse variables ([Sen03]). We present extensions of these instances to more general cases such as mixed-integer recourse, mixed-integer first-stage decisions, and random recourse matrices. In addition to illustrating the generality of our approach, we will also present results associated

with a prototypical implementation using Matlab. One should recognize that in general, Matlab scripts cannot be expected to compete with CPLEX. Nevertheless, we show that as the number of scenarios increase, the Matlab code associated with our implementation performs better than the commercial solver, thus demonstrating the potential for the class of algorithms presented in this chapter.

2.4.1 Illustration Examples

Example 1.1 is an extension of **Example 1.0** (see Appendix A) which has been used in the literature as an instance of binary recourse models. **Example 1.1** illustrates the performance of our algorithm when we include general integer variables in the second stage. **Example 1.1**.

$$\min \quad -1.5x_1 - 4x_2 + \sum_{\omega\in\Omega} p(\omega) f(x,\omega) \tag{2.30a}$$

$$\text{s.t.} \quad x_1, x_2 \text{ binary} \tag{2.30b}$$

where

$$f(x,\omega) = \min \quad -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \tag{2.31a}$$

$$\text{s.t.} \quad \begin{bmatrix} 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \\ 6y_1 + 1y_2 + 3y_3 + 2y_4 - R \end{bmatrix} \leq r(\omega) - T(\omega)x \tag{2.31b}$$

$$y_i \in \{0, 1...5\}, i = 1, ..., 4; R \geq 0 \tag{2.31c}$$

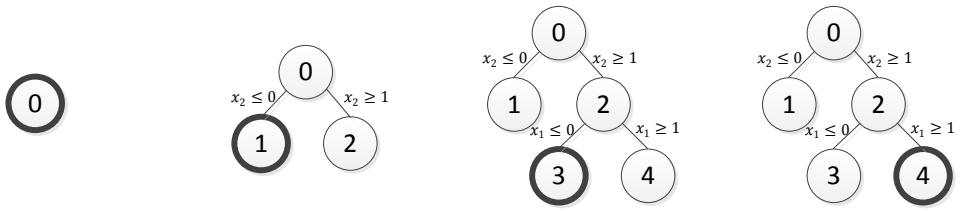
$\Omega = \{\omega_1, \omega_2\}$, $p(\omega_1) = p(\omega_2) = 0.5$, $r(\omega_1) = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$, $T(\omega_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $r(\omega_2) = \begin{bmatrix} 10 \\ 3 \end{bmatrix}$, $T(\omega_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

The summary of applying **M-D²** with **CPT-D** and **BB-D** on **Example 1.1** is shown in Table 2.1 and Table 2.2.

In Table (2.1) and (2.2), each row shows the information for one iteration of the algorithm. Column "Node No." denotes the number of active nodes in the B&B tree. "Cuts No" means

Iter	v	V	x	Node No.	$f(x,\omega_1)$	Cuts No.	$f(x,\omega_2)$	Cuts No.	Value function cut for Node
1	-M-5.5	Inf	(1,1)	1	-57	0	-76	1	$\eta \geq -73.7560 + 6.2292x_1 + 1.0268x_2$
2	-76.7292	-72	(0,1)	1	-57	0	-80.625	4	$\eta \geq -80.0714 + 0.8929x_1 + 11.2589x_2$
3	-73.7560	-72.8125	(0,0)	2	-60.2979	6	-80	5	$\eta \geq -70.1489 + 2.4734x_1 + 0.9468x_2$
4	-72.8125	-72	(0,1)	3	-57	0	-80	2	$\eta \geq -79 + 1.4583x_1 + 10.5x_2$
5	-72.5	-72.5	(0,1)	3					

Table 2.1: **M-D²** with **CPT-D** for Example 1.1

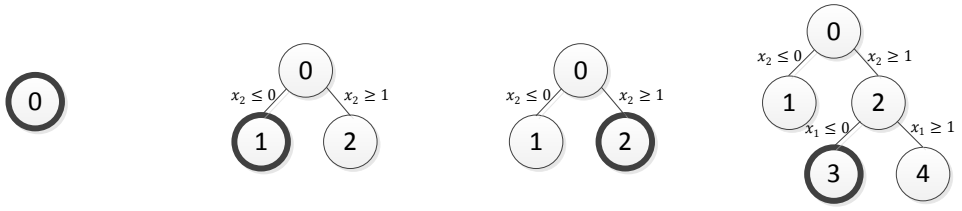


(a) Iteration 1,2 (b) Iteration 3 (c) Iteration 4 (d) Iteration 5

Figure 2.1: Master Problem B&B Tree for **M-D²** with **CPT-D**

Iter	v	V	x	Node No.	$f(x,\omega_1)$	Cuts No.	$f(x,\omega_2)$	Cuts No.	Value function cut for Node
1	-M-5.5	Inf	(1,1)	1	-57	0	-76	1	$\eta \geq -73.7560 + 6.2292x_1 + 1.0268x_2$
2	-76.7292	-72	(0,1)	1	-57	0	-80.1250	6	$\eta \geq -79.7232 + 1.5089x_1 + 11.1607x_2$
3	-73.7560	-72	(0,0)	2	-57	6	-80	4	$\eta \geq -68.5 + 2x_1$
4	-72.5625	-72	(0,1)	2	-57	0	-80	2	$\eta \geq -78.8571 + 1.4643x_1 + 10.3571x_2$
5	-72.5	-72.5	(0,1)	3					

Table 2.2: **M-D²** with **BB-D** for Example 1.1



(a) Iteration 1,2 (b) Iteration 3 (c) Iteration 4 (d) Iteration 5

Figure 2.2: Master Problem B&B Tree for **M-D²** with **BB-D**

the number of multi-term disjunctive cuts generated for that scenario. From the table, we can see that both algorithms generate more cuts in the subproblem than in **Example 1.0** but there is only small difference between **BB-D** and **CPT-D** for this example.

To better illustrate the algorithm, we also include two sets of figures (Figure 2.1 and Figure 2.2) which show the master problem B&B tree at each iteration of the algorithm. The node with black circle contains solution x^k and gets value function updated in the iteration. Similar to the result shown in the tables, there is only minor difference for the master problem B&B tree between the two algorithms.

Example 1.2 We extend **Example 1.1** by requiring general integer variables in the first stage as well. So instead of having x_1, x_2 to be binary, they are allowed to be integers and bounded by $0 \leq x_1, x_2 \leq 5$. The summaries of applying two algorithms are shown in Table 2.3 and Table 2.4. Compared to previous two examples, from Table-(2.3) and (2.4),

Iter	v	V	x	Node No.	$f(x,\omega_1)$	Cuts No.	$f(x,\omega_2)$	Cuts No.	Value function cut for Node
1	-M-27.5	Inf	(5,5)	1	100	0	-47	2	$\eta \geq -261 + 5x_1 + 52.5x_2$
2	-261	-1	(0,0)	1	-60.9545	4	-81.9048	4	$\eta \geq -71.4297 + 3.7564x_1 + 1.0352x_2$
3	-80.3241	-1	(0,3)	2	-19	0	-78.3824	6	$\eta \geq -89.4265 + 1.1912x_1 + 13.5784x_2$
4	-74.3945	-1	(0,1)	3	-57	0	-80	4	$\eta \geq -73 + 1.45x_1 + 4.5x_2$
5	-72.5	-72.5	(0,1)	5					

Table 2.3: **M-D^2** with **CPT-D** for Example 1.2

Iter	v	V	x	Node No.	$f(x,\omega_1)$	Cuts No.	$f(x,\omega_2)$	Cuts No.	Value function cut for Node
1	-M-27.5	Inf	(5,5)	1	100	0	-47	2	$\eta \geq -261 + 5x_1 + 52.5x_2$
2	-261	-1	(0,0)	1	-59.6667	5	-80	5	$\eta \geq -69.8333 + 2x_1 + 1.3333x_2$
3	-77.8333	-1	(0,3)	2	-19	0	-76	3	$\eta \geq -90.25 + 14.25x_2$
4	-73.6667	-59.5	(3,2)	5	-38	0	-61	5	$\eta \geq -74.1 + 1.8667x_1 + 9.5x_2$
5	-72.75	-62	(2,2)	5	-38	0	-66	6	$\eta \geq -81 + 5x_1 + 9.5x_2$
6	-72.5	-63	(0,1)	5	-57	0	-80	0	$\eta \geq -70.5 + 2x_1 + 2x_2$
7	-72.5	-72.5	(0,1)	5					

Table 2.4: **M-D^2** with **BB-D** for Example 1.2

we can see that with general integer variables in the first stage, more iterations are needed to solve the problem. Note also that using **BB-D** requires more iterations than **CPT-D** for the **M-D²** algorithm. However, the average number of cuts for **BB-D** is fewer than that used by **CPT-D**.

Example 1.3 The next extension is to allow randomness in the T matrix. So on top of **Example 1.2**, we set $T(\omega_1) = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.5 \end{bmatrix}$, $T(\omega_2) = \begin{bmatrix} 0.3 & 0.2 \\ 0 & 0.2 \end{bmatrix}$. The summary of applying two algorithms are shown in Table 2.5 and Table 2.6. From Table (2.5) and (2.6)'s

Iter	v	V	x	Node No.	$f(x,\omega_1)$	Cuts No.	$f(x,\omega_2)$	Cuts No.	Value function cut for Node
1	-M-27.5	Inf	(5,5)	1	-22.8333	2	-66.5	2	$\eta \geq -91.1061 + 0.95x_1 + 8.3379x_2$
2	-93.8561	Inf	(5,0)	1	-59.3929	4	-71.3750	4	$\eta \geq -74.6060 + 1.8444x_1 + 0.4550x_2$
3	-81.6960	Inf	(0,2)	2	-57	0	-79.36	6	$\eta \geq -79.2 + 0.84x_1 + 5.51x_2$
4	-80.8424	Inf	(5,3)	2	-38	4	-66.925	8	$\eta \geq -65.1930 + 2.3312x_1 + 0.3581x_2$
5	-79.48	Inf	(5,2)	3	-57	0	-67.7086	10	$\eta \geq -84.1132 + 2.3273x_1 + 5.0612x_2$
6	-78.6816	Inf	(4,2)	4	-57	0	-69.2180	12	$\eta \geq -75.4437 + 0.5315x_1 + 5.1043x_2$
7	-78.16	Inf	(3,2)	4	-57	0	-70.5438	14	$\eta \geq -82.267 + 2.7281x_1 + 5.1554x_2$
8	-77.8543	Inf	(5,2)	6	-57	0	-66.4930	16	$\eta \geq -71.2465 + 4.75x_2$
9	-77.8065	Inf	(1,1)	7	-57	3	-76	13	$\eta \geq -72.3252 + 5.8252x_2$
10	-77.69	-72	(0,1)	7	-57	1	-77.3333	20	$\eta \geq -67.1667$
11	-77.5	-72	(2,2)	7	-57	0	-76	0	$\eta \geq -79.2524 + 1.1355x_1 + 5.2407x_2$
12	-77.5	-77.5	(2,2)	7					

Table 2.5: **M-D²** with **CPT-D** for Example 1.3

Iter	v	V	x	Node No.	$f(x,\omega_1)$	Cuts No.	$f(x,\omega_2)$	Cuts No.	Value function cut for Node
1	-M-27.5	Inf	(5,5)	1	-22.8333	2	-65.6250	3	$\eta \geq -91.2652 + 1.0375x_1 + 8.3697x_2$
2	-93.5777	Inf	(5,0)	1	-57	4	-70.5	4	$\eta \geq -73.6806 + 1.9861x_1$
3	-81.6806	Inf	(0,2)	2	-57	0	-76	4	$\eta \geq -77.2950 + 5.3975x_2$
4	-80.4686	-74.5	(5,3)	2	-38	6	-64.6375	5	$\eta \geq -65.8580 + 2.7049x_1 + 0.3382x_2$
5	-79.7361	-74.5	(4,2)	3	-57	0	-67.5333	7	$\eta \geq -73.1165 + 0.2569x_1 + 4.9111x_2$
6	-79	-74.5	(3,2)	4	-57	0	-68.4810	10	$\eta \geq -81.0190 + 3.7595x_1 + 3.5x_2$
7	-77.5098	-74.5	(5,2)	5	-57	0	-66	4	$\eta \geq -69.8442 + 1.0174x_1 + 1.6287x_2$
8	-77.5	-77	(2,2)	5	-57	0	-76	0	$\eta \geq -76.0454 + 1.1008x_1 + 3.6719x_2$
9	-77.5	-77.5	(2,2)	5					

Table 2.6: **M-D²** with **BB-D** for Example 1.3

result, we can conclude that **BB-D** needs far fewer cuts on average for solving subproblems than **CPT-D**. In addition, the total number of iterations needed is also less.

2.4.2 Experiments with a Computational Prototype.

In this subsection we report experiments with a Matlab prototype to give the reader a sense of the potential for the methodology presented in this chapter. We designed a Matlab implementation of the **M-D²** algorithm with calls to the CLPEX LP solver whenever an LP solution was required. For all other purposes (e.g. managing the B&B tree) the Matlab script operated in the Matlab environment. Because Matlab is a scripting language, there are huge overheads in execution, and cannot be expected to compete with codes written in C or C++. Such handicaps notwithstanding, we conducted an experiment to see how our procedures scale with increases in the number of scenarios. Moreover, we wish to study how a commercial software like CPLEX might perform on the same instances.

Three sets of instances are generated based on **Example 1.1-1.3**. For each example, we create 4,9,36,121,441 scenarios by generating the right hand sides $r(\omega)$ from equidistant lattice points in $[5, 15] \times [5, 15]$ with equal probability assigned to each point (this methodology was borrowed from [ATS04]). For the four instances based on Example 1.3, besides random right hand side $r(\omega) = \begin{bmatrix} r_1(\omega) \\ r_2(\omega) \end{bmatrix}$, $T(\omega)$ is also random by setting $T(\omega) = \begin{bmatrix} 1/r_1(\omega) & 1 - 1/r_1(\omega) \\ 1/r_2(\omega) & 1 - 1/r_2(\omega) \end{bmatrix}$. Table 2.7 shows the result comparing **M-D²** with **CPT-D** and **BB-D** with deterministic equivalent solved by default setting of CPLEX 12.3 MILP solver (running on Windows with Intel i7-3770K 3.5GHz). In the table, Instance 1-3 corresponds to variations based on Example 1.1-1.3. **Obj** denotes the optimal objective value of the SMIP. **Var** and **Constr** denotes the number of variables and constraints in the **DEP**. The entries in column **M-D²(CPT)** and **M-D²(B&B)** denote Iterations (Master Nodes, Second-stage

Leaf Nodes, Running Time) which correspond to the total number of iterations , the total number of nodes in the B&B tree in the master problem, the maximum leaf nodes encountered during solving the subproblem and the total running time. The **DEP** column shows the solving time of **DEP** from default CPLEX MIP solver. The max cpu time allowed is 20 minutes. Again, we reiterate that our algorithms were coded in the Matlab environment.

From Table 2.7's result, between column **M-D^2(CPT)** and **M-D^2(B&B)**, the same result shows as in the examples that **M-D^2(B&B)** needs less number of iterations to solve the instance and as a result, it takes less running time for all the instances. Compared with commercial solver's running time on **DEP**, all three methods take longer time to solve the instances as the scenarios getting larger. And for instances with 441 scenarios, **M-D^2(B&B)** takes less than half of running time of **M-D^2(CPT)** to solve the three instances and the MILP solver has difficulties to get the optimal objective within 20 minutes. Based on the instances tested, algorithm **M-D^2** shows very stable results on the solving time relative to CPLEX default MILP solver on **DEP**.

	Scenarios	**Obj**	**Var**	**Constr**	**M-D^2(CPT)**	**M-D^2(B&B)**	**DEP**
Instance 1	4	-63.50	26	17	7 (4, 5, 0.27)	5 (4,5,0.14)	0.23
	9	-66.56	56	37	20 (15, 10, 1.9)	18 (15, 10, 0.98)	0.02
	36	-66.83	218	145	7 (2, 7, 1.34)	6 (2, 7, 1.01)	0.02
	121	-67.17	728	485	7 (1, 8, 4.01)	6 (1, 8, 2.96)	0.16
	441	-65.58	2648	1765	16 (3, 13, 41.90)	10 (2, 7, 15.27)	1.58[1]
Instance 2	4	-63.50	26	17	20 (14, 10, 0.5)	12 (14, 10, 0.3)	0.02
	9	-66.56	56	37	20 (15, 10, 1.72)	18 (15, 10, 0.94)	0.02
	36	-69.86	218	145	19 (16, 10, 6.93)	18 (16, 10, 4.54)	0.03
	121	-71.12	728	485	18 (16, 11, 22.46)	17 (16, 11, 13.51)	4.09
	441	-69.64	2648	1765	20 (16, 20, 147.17)	18 (15, 21, 58.97)	Failed(>20mins)
Instance 3	4	-63.50	26	17	18 (15, 15, 1.31)	18 (15,15,1.26)	0.00
	9	-64.22	56	37	25 (20, 20, 6.57)	22 (20, 20, 3.07)	0.13
	36	-65.94	218	145	28 (22, 23, 33.43)	30 (22, 23, 14.87)	4.18
	121	-66.52	728	485	31 (22, 25, 122.57)	32 (22, 25, 51.03)	Failed(>20mins)
	441	-68.30	2648	1765	31 (23, 25, 371.61)	31 (22, 28, 134.89)	Failed(>20mins)

Table 2.7: Comparison of **M-D^2** with DEP

2.5 Conclusion

We develop a decomposition algorithm to solve two-stage stochastic integer program. The algorithm allows general mixed-integer variables in both stages as well as scenario dependent matrices W and T. The algorithm uses multi-term disjunctive cuts to obtain value function approximation of the second-stage mixed-integer programs. In fact, the multi-term disjunctive cuts can be obtained via two alternative approaches: cutting plane tree method and branch and bound tree convexification. Both of them are proved to be finitely convergent. We proved that the proposed decomposition algorithm leads to finite convergence. To the best of our knowledge, this is the first time-staged decomposition method that allows us to partially solve subproblems with mixed integer variables ultimately guaranteeing that a mixed-integer optimum is obtained. Since the cutting plane generation for the second stage is separable between scenarios, the value function approximation of subproblems for each scenario in our decomposition scheme can be obtained in parallel.

[1]Obj=65.576 and the solution has numerical issues

Chapter 3: Solving Families of Mixed Integer Programs Arising in Stochastic Mixed Integer Programming

3.1 Introduction

As computational and theoretical advances continue, it is now possible to derive algorithms for solving stochastic mixed-integer programming problems with general integer variables (SMIP-G). It is not trivial to solve these potentially large number of NP-hard problems which are slightly different from each other depending on how random variables appear in the constraints. However, the solutions of families of instances are closely related. Our investigation starts by finding good approximations of the convex hull of mixed-integer solutions: IP convex polytope approximation that may be helpful for re-optimizing future LP-relaxations of IPs or IPs themselves.

Our study will be experimental in nature, and we will investigate both "wait-and-see" and "here-and-now" formulations of stochastic programming problems. In the case of wait-and-see problems, we consider situations in which the solution may have to be implemented in real-time, as data becomes available. The models are NP-hard problems, but we can compensate by setting up some approximations based on historical data in advance. This is common in many production-operations models in which weekly aggregate plans are made prior to observations of data, and once real-time data becomes available, the models are

updated, and there is a need to solve problems with actual data rapidly. For instance, suppose we have ample time to solve the problem with estimated cost vector c^* and extract any other information deemed useful. One can create strong LP approximations and once the true cost vector c becomes known, we must quickly, possibly in real time, verify if the solution to the problem with estimated cost c^* is still optimal, and, if not, produce a solution to the problem with true cost c. The other class of models in stochastic programming are often referred to as here-and-now models. For this class of models, some part of the plan must be made prior to observing the data, and other pieces of the plan can be implemented later. It is common to solve these problems by decomposing the planning process into two or more stages. When the second stage model is an MILP, it is again necessary to solve a family of mixed-integer linear programming (MILP) problems which may benefit from preparations that help create stronger LP approximations. In this way, the approximation used to solve one scenario with fixed first stage solution x can be treated as a warm-start point for other scenarios, and also used to solve scenarios with different first-stage solution x.

3.2 Methods for approximating convex hull of mixed-integer points

For a general MILP with n variables

$$\min_{x \in X} c^\top x \tag{3.1}$$

where $X = \{Ax \geq b, x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n-n_1}\}$ and the first n_1 variables are required to be integer, A is a $m \times n$ matrix and the bounding constraints $\{l_j \leq x_j \leq u_j \text{ for } j = 1...n\}$ (if they exist) are assumed to be included in X. We define X_{IP} as the convex hull of all the mixed-integer feasible points in X. Of course, obtaining it for a general MILP problem is a far-reaching goal. So we take a wider perspective and propose an outer approximation of X_{IP} which is

required to satisfy the following conditions

$$\min_{x \in X} c^\top x = \min_{x \in X_C} c^\top x \tag{3.2}$$

$$X \subseteq X_C \tag{3.3}$$

where X_C is a convex polyhedron. While X_{IP} is such a polyhedron, not all approximations that satisfy (3.2-3.3) are convex hulls of X.

Branch and bound (B& B) method [LD60] is a popular method for solving MILP problems and is included in most state-of-art MILP solvers. Recent advances in the cutting plane methods [CKS12] for solving MILP problems make it possible to transform the B&B tree obtained from an MILP solve to create a polyhedral approximation that gives the same IP optimal value as obtained from the B&B method. The disjunctive nature of leaf nodes used in the B&B method make it possible to obtain X_C from it. For the remainder of this section, we describe such an algorithm and prove that this approximation can be obtained in finitely many steps.

Let the LP relaxation of X be

$$X_L = \{Ax \geq b, x \in \mathbb{R}^n\}. \tag{3.4}$$

Suppose problem (3.1) is solved by the B&B method and its optimal solution is x^*. We also have the set of leaf nodes $\mathcal{Q} = \mathcal{Q}_{\text{remain}} \cup \mathcal{Q}_{\text{fathom}}$ where $\mathcal{Q}_{\text{remain}}$ are the remaining leaf nodes in the B&B tree and $\mathcal{Q}_{\text{fathom}}$ are the leaf nodes that are fathomed. Suppose there are T nodes in $\mathcal{Q}$ with each node having bounding constraints denoted as Q_t:

$$Q_t = \{x | L_t \leq x \leq U_t\}. \tag{3.5}$$

$Q_t, t \in \{1, ..., T\}$ are disjoint from each other. $\mathcal{Q}$ provides us a disjunctive relaxation of X:

$$X_D = \bigcup_{t \in T} (X_L \bigcap Q_t). \tag{3.6}$$

Our plan is to develop multi-term disjunctive cuts using X_D. The set of hyperplanes that characterizes the convex hull of X_D (3.6) forms a cone. To truncate this set and obtain a bounded solution, various normalization constraints have been studied in the literature ([Bal79], [BB09], [BP02] and [BCC93]). In [CKS12], the authors compared two normalization schemes for multi-term disjunctions. In this chapter, we use the minimum l_1 norm cut (M1NC) version of normalization although others are clearly possible [Cad10]. A cut generation LP with the M1NC normalization formulation is:

$$\begin{aligned} \textbf{Minimize} \quad & \sum_j |\pi_j| \\ \textbf{s.t.} \quad & \pi - A^\top \lambda_t - \mu_t + \nu_t = 0, \quad \text{for } t \in \{1, ..., T\} \\ & b^\top \lambda_t + L_t^\top \mu_t - U_t^\top \nu_t \geq \pi^\top x^k + 1, \quad t \in \{1, ..., T\} \\ & \lambda_t \geq 0, \mu_t \geq 0, \nu_t \geq 0 \quad t \in \{1, ..., T\} \end{aligned} \tag{3.7}$$

where x^k is a fractional point and satisfies $x^k \notin X_D$. The cut derived from this cut-generation LP has the following form:

$$\pi^\top (x - x^k) \geq 1. \tag{3.8}$$

Similar to the cutting plane tree algorithm ([CKS11]), cuts (3.8) are derived and added into X_L each time a fractional solution is encountered. The algorithm to obtain a polyhedral approximation has two phases. In the first phase, for each fractional solution encountered in the B&B method, a cut is derived and added into X_L. In the second phase, X_L is updated until no new fractional solution is encountered. The algorithm is shown in Algorithm 4.

Define **clconv** as the operation to get closure of the convex hull of a set. Then we have the following theorem.

Proposition 5. *Algorithm 4 terminates in finitely many steps (say $K < \infty$) and the resulting X_L^K satisfies conditions (3.2) and (3.3).*

Algorithm 4 A cutting plane tree algorithm to obtain a polyhedral approximation using a B&B tree as given.

Input: X_L, set of leaf nodes $\mathcal{Q} = \mathcal{Q}_{\text{left}} \cup \mathcal{Q}_{\text{fathom}}$, all fractional points that were generated during the B&B algorithm S.
Initialize iteration $k = 1$, LP relaxation $X_L^k \longleftarrow X_L$. The cardinality of $\mathcal{Q}$ be T.
for all fractional solution $x \in S$ **do**:
    Formulate CGLP as in (3.7) using x, X_L and $\mathcal{Q}$. Solve it to get a cut.
end for
Set X_L^1 as X_L plus all derived cuts.
while true **do**
    Solve $\min_{x \in X_L^k} c^\top x$ and get solution x^k.
    if $x^k \in \bigcup_{t \in \{1..T\}} Q_t$ **then break**.
    else, formulate CGLP as in (3.7) using x^k, X_L and $\mathcal{Q}$. Solve it to get a cut.
    end if
    Update X_L^k with the new cut and set it as X_L^{k+1}.
    $k \leftarrow k + 1$.
end while

Proof. Each time x^k changes,a new extreme point of the CGLP is generated. Because there are finitely many extreme points of the CGLP and a subset of these corresponds to facets of clconv$\{X_D\}$, we will have $x^K \in X_D$ in finitely many iterations, which is equivalent to the stopping rules: $x^K \in \bigcup_{t \in \{1..T\}} Q_t$. For the latter part of the theorem, also from the algorithm's stopping rules, there exists x^K such that $x^K \in X_D$. If $x^K = x^*$, then obviously (3.2) and (3.3) are met. On the other hand, let the output of the B&B procedure be denoted as x^*. Suppose that $x^K \neq x^*$. Since the B&B tree embodies the disjunction, it follows that $x^* \in X_D$. Hence $c^\top x^K \geq c^\top x^*$. But since every solution generated by the algorithm is based on approximations strengthened by valid inequalities, it follows that $c^\top x^K \leq c^\top x^*$. It then follows that $c^\top x^K = c^\top x^*$. □

3.3 Computational experiments with "wait-and-see" SMIP

As discussed in the introduction, there are several occasions in which uncertainty is managed using a "wait-and-see" approach. In this section, we report results of a computational experiment using the Algorithm 4.

The experimental plan was aimed at examining the accuracy of the approximation LP obtained by Algorithm 4 and the time consumed to obtain such an approximation. In this experiment we will also compare the performance of multi-term disjunctions with (arguably) the most popular class of disjunctive cuts, namely, the Lift-and-Project (L& P) cuts [BCC93]).

Algorithm 5 Algorithm for approximating IP convex polytope using lift-and-project cuts

initialize iteration $k = 1$, LP relaxation $X_L^k \longleftarrow X_L$. Leaves nodes set $\mathcal{Q} = \mathcal{Q}_{\text{left}} \cup \mathcal{Q}_{\text{fathom}}$ with the cardinality of $\mathcal{Q}$ be T. Fractional solution set from B&B method S
for each fractional solution $x \in S$ **do**
  for each fractional x_j where $j \in \{1...n_1\}$ **do**
    Derive lift and project cut using x_j, X_L.
  end for
end for
Set X_L^1 as X_L plus all derived cuts.
while (**do**true):
  Solve $\min_{x \in X_L^k} c^\top x$ and get solution x^k.
  if $x^k \in \bigcup_{t \in \{1..T\}} Q_t$ **then break**.
  else
    for each fractional x_j^k where $j \in \{1...n_1\}$ **do**:
      Derive lift and project cut using x_j^k, X_L^k.
    end for
  end if
  Update X_L^k with the new cuts and set it as X_L^{k+1}.
  Update $k \leftarrow k + 1$.
end while

Cuts are derived for each fractional variable from the solution. Although it has been shown in [Bal79] and [OM01] that the absence of facial disjunctive properties may lead to infinitely convergent process for general MILP if using algorithm (in Table 5). Practically lift-and-project cuts take less time to generate.

Our computational results are reported in two steps. In the first step, we compare the performance of each algorithm in generating the approximation X_D. The second step determines the quality of the approximations by examining the solution time to recover optimality using MIP solver after perturbing cost vectors as well as the quality of the LP optimum using the approximation. All experiments were conducted on an Intel CPU 2500k running at 3.3GHz. The CPLEX 12.3 LP/MIP solver was adopted to optimize all LPs/MIPs encountered in the algorithms. We chose the instances from MIPLIB 2003 ([BES$^+$96]) that could be solved by the default setting of CPLEX MIP solver within 15 minutes. The basic information of these instances are shown in Table 3.1. **Num of Cols** denotes the number of variables. **Num of Rows** denotes the number of constraints. **Num of Rows with Cuts** denotes the number of constraints on the root node from CPLEX MIP solver. **Non Zeros** denotes the non-zeros elements in the constraint matrix on the root node. In the experiment, X_{LP} is initialized from the root node LP by CPLEX MIP solver. Then it is solved by the B&B algorithm. The setting for Theorem 5 is impractical in practice because the B&B tree is normally too large to use for generating cuts using the entire B&B tree. Instead we truncate the B&B tree to a size that is manageable for the CGLP. The size we use in the experiment is dictated by the number of leaf nodes T. Letting NZ denote the number of non-zero elements of the matrix defining X_{LP}, we limit T so that $NZ * T \leq 3 \times 10^4$. Compared to the experiments shown in [CKS12], which compare the performance of using multi-term disjunctive cuts and two-term disjunctive cuts to solve MILP, this experiment

Instances	Num of Cols	Num of Rows	Num of Rows with Cuts	Non Zeros
aflow30a.mps	842	479	594	4680
bell3a.mps	133	123	140	400
bell5.mps	104	91	110	372
blend2.mps	353	274	300	1920
dcmulti.mps	548	290	437	3530
fixnet6.mps	878	478	698	3885
flugpl.mps	18	18	25	107
gen.mps	870	780	813	2834
glass4.mps	322	396	525	2168
gt2.mps	188	29	53	744
lseu.mps	89	28	44	696
mas74.mps	151	13	15	1998
mas76.mps	151	12	15	2075
misc03.mps	160	96	111	2564
misc07.mps	260	212	224	9036
mod008.mps	319	6	24	5014
modglob.mps	422	291	427	2063
noswot.mps	128	182	188	778
p0033.mps	33	16	36	214
p0201.mps	201	133	156	3279
p0282.mps	282	241	302	2810
p0548.mps	548	176	253	2323
pk1.mps	86	45	45	915
pp08a.mps	240	136	289	1124
pp08aCUTS.mps	240	246	392	1732
qiu.mps	840	1192	1195	4436
rgn.mps	180	24	56	1084
rout.mps	556	291	317	3782
set1ch.mps	712	492	693	2327
stein15.mps	15	36	37	128
stein27.mps	27	118	118	378
stein45.mps	45	331	331	1034
vpm1.mps	378	234	275	1171
vpm2.mps	378	234	307	1322

Table 3.1: Instances

focuses more on comparing the time needed to obtain convex approximation as well as the quality of this approximation given a B&B tree from the MILP solution using the multi-term and two-term disjunctive cuts.

			Cuts using Multi-term disjunctions			Lift-and-Project Cuts		
Instances	**T**	**NZ*T**	**Time(secs)**	**Cuts (nos)**	**Obj Inc**	**Time(secs)**	**Cuts (no.)**	**Obj Inc**
aflow30a.mps	7	32760	33.649	6	15.05%	13.837	396	0.00%
bell3a.mps	76	30400	106.54	75	1.64%	0.328	90	0.00%
bell5.mps	81	30132	58.11	41	33.97%	3.713	948	0.00%
blend2.mps	16	30720	27.222	16	0.56%	2.668	358	0.00%
dcmulti.mps	9	31770	44.382	8	15.20%	8.892	378	>100%
egout.mps	7	2912	0.094	6	95.60%	0.11	22	0.00%
fixnet6.mps	8	31080	79.701	7	14.10%	24.398	225	0.00%
flugpl.mps	149	15943	60.076	657	53.92%	9.048	2909	0.00%
gen.mps	5	14170	5.491	4	100.00%	0.344	10	100.00%
glass4.mps	14	30352	2.512	6	100.00%	16.021	1133	>100%
gt2.mps	41	30504	48.407	18	0.00%	4.009	376	INSTABLE
lseu.mps	44	30624	17.207	43	0.56%	4.15	535	INSTABLE
mas74.mps	16	31968	2.028	15	100.00%	2.309	196	33.95%
mas76.mps	15	31125	0.592	0	0.00%	4.15	171	>100%
misc03.mps	12	30768	0.577	11	100.00%	3.292	331	35.26%
misc07.mps	4	36144	0.125	3	100.00%	1.544	76	67.74%
mod008.mps	6	30084	0.671	5	37.23%	2.012	92	0.00%
modglob.mps	15	30945	39.562	1	10.39%	7.082	304	0.00%
noswot.mps	39	30342	10.639	38	NO INC	4.618	698	NO INC
p0033.mps	2	428	0	1	NO INC	0.016	2	NO INC
p0201.mps	10	32790	7.722	9	0.00%	9.5	615	0.00%
p0282.mps	11	30910	5.023	10	24.93%	6.131	487	0.00%
p0548.mps	13	30199	49.639	12	24.99%	6.645	434	0.00%
pk1.mps	33	30195	65.504	32	NO INC	4.181	480	NO INC
pp08a.mps	27	30348	171.943	26	25.36%	12.199	658	0.00%
pp08aCUTS.mps	18	31176	47.33	17	8.62%	7.503	426	0.00%
qiu.mps	7	31052	18.58	6	37.36%	9.469	224	0.00%
rgn.mps	28	30352	5.429	27	5.82%	2.933	293	0.00%
rout.mps	8	30256	9.251	7	100.00%	43.586	310	>100%
set1ch.mps	13	30251	286.104	12	64.78%	51.746	1547	0.00%
stein15.mps	133	17024	3.853	132	66.67%	7.785	945	0.00%
stein27.mps	80	30240	21.497	79	0.00%	15.164	1560	0.00%
stein45.mps	30	31020	16.177	29	0.00%	17.691	1155	0.00%
vpm1.mps	26	30446	37.113	25	NO INC	3.962	254	NO INC
vpm2.mps	23	30406	166.046	22	7.17%	6.443	397	0.00%
Sum			1448.796	1406		317.479	19035	

Table 3.2: Performance of Convex Polyedral Approximations using Multi-term and Lift-and-Project Cuts

3.3.1 Performance of Polyhedral Approximations using alternative types of disjunctions

The experiment results are shown in Table 3.2. In the table, **T** denotes the leaves' size. **Cuts count** shows the number of cuts added. **obj inc** means the relative objective gap closed for the convex approximation obtained. Its calculation is shown in (3.9),

$$(Obj_C - Obj_{LP})/(Obj_B - Obj_{LP}) \tag{3.9}$$

where Obj_C is the LP value obtained by the approximation. Obj_{LP} is the LP relaxed value without the B&B. Obj_B is the objective value obtained from the truncated B&B. If $Obj_B - Obj_{LP} = 0$, we denote the **Obj Inc** as $NOINC$. Because of instability of CGLP, instance "gt2.mps" and "lseu" have invalid cuts in the solving process (shown in Table 3.2). We can see from the result , there are four instances: "dcmulti", "glass4", "mas76" and "rout" that give higher objectives than the B&B optimal objective value. This is when the disjunctive cut shows better performance than B&B method under a fixed solution time. For other instances, multi-term disjunctions provide stronger approximations (i.e., higher LP objective values) than L&P in most cases. On average, approximation using L&P cuts takes less time to generate while producing many more cuts.

3.3.2 Comparison of IP solution times

The next step of the experiment considers the situation in which we warm start the MILP by using the polyhedral approximation obtained in the previous step. Instances that have feasible approximations for both algorithms in the previous experiment are included in this experiment. The cost coefficient c_i is randomly perturbed within $p\%$ of its original value for each instance as shown in (3.10).

$$c_i = c_i(1 + 0.01pU_i) \tag{3.10}$$

where $U_i \sim U(-1, 1)$ is a random variable conforming to uniform distribution between $(-1, 1)$. For each instance and $p \in \{5, 10, 20, 50, 100\}$, we examine 20 random generated cost vectors and measure the approximations performance by comparing

1 the difference between real objective value with the objective value obtained from solving approximation LP,

2 the solution time,

3 and the solution time after warm-starting using the approximate LP.

The methods included are

1 CPLEX MILP solver (IP),

2 using approximation by Algorithm 4 and solve it as LP (Multi-term),

3 using approximation by Algorithm 4 and solve it as IP (Multi-term warm start for IP),

4 approximation obtained by Algorithm 5 and solve it as LP (L&P LP),

5 and approximation obtained by Algorithm 5 and solve it as IP (L&P warm start for IP).

The detailed result for deviation from the IP optimal solution is shown in Table B.1 of Appendix-B. Suppose the approximated objective value is V_a, the true value is V, and the deviation is measured by (3.11)

$$(V - V_a)/|V| \tag{3.11}$$

For each (instance, p) pair, 20 replications are performed in the experiment. The deviation results of the 20 replications are summarized using **mean** (the average of deviation), **std**(the

standard deviation), **5% prc** (5% percentile), **95% prc** (95% percentile). If we compare the deviation of LP optimal for the approximation from the IP optimal, approximation with multi-term disjunctive cuts has less deviation in total. At the same time, the solution time in Table B.2 of Appendix B shows that approximations obtained using multi-term disjunctive cuts also takes less time in total.

When using the approximation as a starting point to continue optimizing to IP optimal, we compare the solution time for the three methods and rank them from 1 to 3 for each instance tested in the experiment. A summary of the solution time ranking is provided in Table 3.3 where the numbers in the table denote how many times the method is placed as Rank 1, 2, or 3 in the experiment. We can see from the result that L&P gets the most counts of Rank 1 and is comparable to CPLEX. Multi-term disjunctive cuts in this case, has the most counts of Rank 3 and least counts of Rank 1 and 2. In summary, if using

Rank	Multi-term disjunctive Cuts	L&P	CPLEX
1	22	55	52
2	31	49	55
3	77	26	23

Table 3.3: The ranking of solution time to IP Optimal

approximation as a starting point to approach IP optimal, approximation obtained using L&P cuts is better. If only using approximations's LP optimal to approximate IP optimal, approximations derived by multi-term disjunctions are better.

3.4 Stochastic Integer Programming with Recourse

In this section, we focus our experiment on solving the stochastic mixed-integer programming (SMIP) problems with recourse. The goal is to verify whether it is possible to speed up existing SMIP algorithms using polyhedral approximations.

The specific algorithm we investigate is perhaps the oldest decomposition-based algorithm for SMIP, namely, the integer L-shaped method of Laporte and Louveaux ([LL93]). We will refer to this algorithm as the L2 method. Since this method requires that we solve each subproblem MILP in each iteration, it may be possible to reduce solution times for each scenario by warm-starting each subproblem SIP using cuts that may have been generated in previous iterations.

Suppose we have a two-stage SMIP. Its subproblems have the following formulation for each scenario (realization) ω of $\tilde{\omega}$:

$$\begin{aligned} f(x,\omega) = \min \quad & g^{\mathsf{T}} y \\ \textbf{s.t.} \quad & Wy \geq r(\omega) - T(\omega)x \\ & y \in Y \cap Q \end{aligned} \tag{3.12}$$

where

$$\begin{aligned} Y = \{y \mid & y_j \geq 0, \forall j \in J_1 \subseteq \{1...n_2\} \\ & y_j \text{ is integer}, \forall j \in J_2 \subseteq \{1...n_2\} \backslash J_1\} \subseteq \mathcal{R}^{n_2} \\ Q = \{y \mid & l \leq y \leq u\} \end{aligned}$$

and x is the first stage variable. Then for fixed x^k and scenario ω, suppose we have the fractional solution $y^*(\omega)$ from (3.12). The multi-term disjunctive cut (3.13) can be obtained based on (3.7).

$$\pi^\top (y - y^*) \geq \min_{t \in \{1...T\}} ((r(\omega) - T(\omega)x)^\top \lambda_t + l_t^\top \mu_t - u_t^\top \nu_t) = \pi_0(x, \omega) \tag{3.13}$$

For iteration $k' > k$ with different $x^{k'}$ or $\omega' \neq \omega$, we can derive a new valid multi-term disjunctive cut by just inserting $x^{k'}$ or ω' into $\pi(x, \omega)$ which is generated from previous iterations or other scenarios. We call this process "warm starting subproblems solution". However, it is important to note that without additional simplifications (as suggested by convexifications proposed in Sen and Higle (2005)), cut generation can become extremely complicated because of nested formulas using (3.13) beyond the first "warm-start". Hence our conclusions will be limited to only using one set of cuts, generated at the start of the process. In the experiment, we allow generating convex polytope approximation up to a certain iteration K_a and only one approximation generated for each scenario. After iteration K_a, these approximation are used to warm-start the subproblem solution. The algorithm is shown in Algorithm 6.

Algorithm 6 L2 algorithm with subproblems warm start

initialize iteration $k = 1$, approximation iteration limit K_a, objective value lowerbound $v = -\infty$, upperbound $V = \infty$ and stopping limit ϵ.
while (**do**true):
  Solve master problem and obtain x^k, v.
  If $V - v \leq \epsilon$, **break**.
  for each fractional scenarios $\omega \in \Omega$ **do**
    if $k > K_a$ **then**
      Solve the subproblem $f(x^k, \omega)$ using B&B method.
      Obtain approximation using Algorithm (Table-4) or (Table-5) for scenario ω.
    else
      Warm start subproblem solution.
      Solve the subproblem $f(x^k, \omega)$ using the approximation.
    end if
    Obtain the subproblems value function cut as described in L2 algorithm.
    Update objective upperbound V.
    Update $k \leftarrow k + 1$.
  end for
end while

We test the L2 algorithm using stochastic server location problem (SSLP) instances ([NS05]). The need for such models may arise in circumstances where the presence of demand in a network is not known with certainty at the time that server location decisions need to be made. As an example, the Department of Defense may be interested in choosing the locations of bases, recognizing that terrorist threats may occur in the future. Similar applications arise in situations in which discrete resources must be assigned to tasks before complete information regarding the location or type of tasks become known. The size of the model is captured by the name "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. We generated a set of SSLP instances following the method in ([NS05]). Table 3.4 gives the dimensions of the deterministic equivalent problems and their subproblems for comparative purposes. Here, the column **Bvars** reports the number of binary variables and the column **Cvars** reports the number of continuous variables.

The experiment has five settings as shown in Table 3.5. The first one is purely the L2 algorithm. It is compared with the decomposition algorithm described in Algorithm 6. Inside Algorithm 6, two proposed methods are compared to get the approximation of second stage mix-integer points' convex hull as shown in the **Approximation Algorithm** column. In addition, recall that $\pi_0(x, \omega)$ shown in (3.13) is valid for any pair of (x, ω). Thus the cuts derived for one scenarios approximation can be used to set up approximations for other scenarios. To test the effectiveness of utilizing cuts from other scenarios, another option column is inserted called **Warm-starting approximation sources**. The algorithm can use either all scenarios' cuts or merely its own cuts from previous iterations. The iterations that allow generating cuts in the approximation are restricted by $\bar{k}$ as shown in Algorithm 4 and Algorithm 5. All instances are tested under the experiment settings shown in Table 3.5

	DEP			Subproblem		
Instances	Constrs	Bvars	Cvars	Constrs	Bvars	Cvars
sslp_5_25_50	1501	6255	250	30	130	5
sslp_5_25_100	3001	12505	500	30	130	5
sslp_6_25_50	1551	7506	300	30	130	5
sslp_6_25_100	3101	15006	600	30	130	5
sslp_7_25_50	1601	8757	350	30	130	5
sslp_7_25_100	3201	17507	700	30	130	5
sslp_5_50_50	2751	12505	250	55	255	5
sslp_6_50_50	2801	15006	300	55	255	5
sslp_7_50_50	2851	17507	350	55	255	5
sslp_5_25_1000	30001	125005	5000	30	130	5
sslp_6_25_1000	31001	150006	6000	30	130	5
sslp_7_25_1000	32001	175007	7000	30	130	5
sslp_5_50_1000	55001	250005	5000	55	255	5
sslp_6_50_1000	56001	300006	6000	55	255	5
sslp_7_50_1000	57001	350007	7000	55	255	5

Table 3.4: Instances Dimension

Settings	Decomposition Algorithm	Approximation Algorithm	Warm-starting approximation sources
1	L2 algorithm	N/A	N/A
2	Algorithm-6	Algorithm-4	scenario's own
3	Algorithm-6	Algorithm-4	all scenarios'
4	Algorithm-6	Algorithm-5	scenario's own
5	Algorithm-6	Algorithm-5	all scenarios'

Table 3.5: Settings Description

with $\bar{k} \leq 1$. We also did the experiment with $\bar{k} \leq 3$ on instances with less than 50 scenarios for comparison purposes.

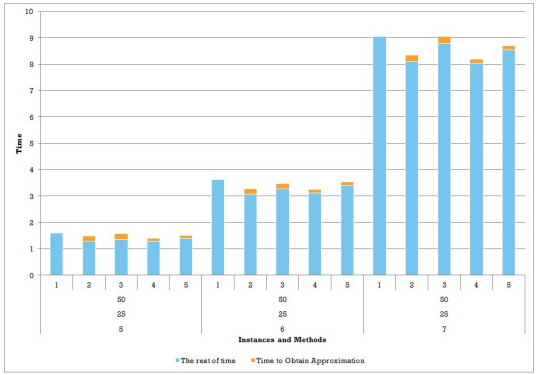
To ensure the reliability of the experiment, all computational results reported in this section were averaged based on 10 replications. The results are shown in Table 3.6. From

	$\bar{k} \leq 1$					$\bar{k} \leq 3$				
Settings	1	2	3	4	5	1	2	3	4	5
sslp_5_25_50	1.59	1.49	1.57	1.41	1.51	1.62	1.76	1.88	1.51	1.67
sslp_5_25_100	3.28	3.09	3.44	2.98	3.34	2.88	3.06	3.53	2.76	3.45
sslp_6_25_50	3.63	3.28	3.47	3.25	3.54	3.67	3.63	3.95	3.39	3.90
sslp_6_25_100	7.67	6.90	7.69	6.74	7.84	7.43	7.21	8.55	7.00	8.75
sslp_7_25_50	9.06	8.35	9.05	8.20	8.70	8.95	8.60	9.35	8.25	9.28
sslp_7_25_100	16.79	15.13	16.74	14.89	17.43	16.45	15.53	18.29	14.97	19.11
sslp_5_50_50	8.38	12.93	13.40	8.59	8.96	8.22	21.99	23.15	8.85	9.80
sslp_6_50_50	15.52	19.10	20.12	15.23	15.98	15.20	27.30	29.75	15.51	17.51
sslp_7_50_50	30.21	35.51	37.73	29.69	31.55	29.82	47.43	53.01	30.08	34.44
Average	10.68	11.75	12.58	10.11	10.98	10.47	15.17	16.83	10.26	11.99
sslp_5_25_1000	27.41	24.76	48.22	24.08	57.03					
sslp_6_25_1000	71.20	64.86	139.80	63.86	163.20					
sslp_7_25_1000	170.37	158.81	349.41	154.77	408.48					
sslp_5_50_1000	156.80	249.33	418.15	158.76	296.93					
sslp_6_50_1000	301.30	427.43	813.75	310.35	622.37					
sslp_7_50_1000	589.21	726.47	1549.94	577.15	1320.16					
Average	219.38	275.28	553.21	214.83	478.03					

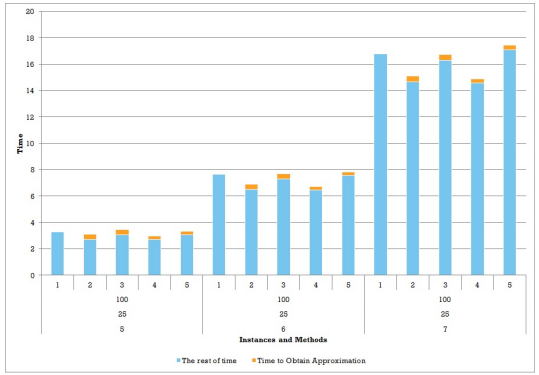
Table 3.6: SIP experiment solution time (secs) comparison

the result, we can see for both $\bar{k} \leq 1$ and $\bar{k} \leq 3$, approximation using L&P cuts achieved the best performance, and approximation using multi-term disjunctive cuts takes the longest time on average. The warm-start using approximation from all scenarios (settings 3 and 5) does not help in reducing the solution time (compared with settings 2 and 4) which means the overhead of all scenarios' cuts is much larger than its benefits. One possible reason is there is no overlap among the feasible regions of different scenarios. From the result of settings 1, 2 and 4, for instances with less variables in the subproblems (e.g., $n = 25$), the approximation method using multi-term disjunctive cuts (setting 2) outperforms the pure L2 method (setting 1) and is almost the same in solution time as approximations with L&P cuts (setting 4) for all instances. This suggests that warm-start subproblems using multi-term disjunctive cuts also helps in solving stochastic programming problems. For instances with more variables (e.g., $n = 50$), the B&B tree grows and CGLP takes longer time to

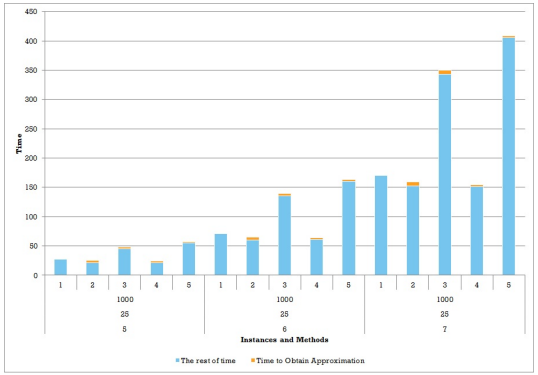
derive multi-term disjunctive cuts, which can be seen from the details of solution times in Figure-3.1. The time that it takes to obtain approximation is in orange color and the remaining solution time is in blue. If we only look at the blue part, the L2 decomposition method is warm-started more effectively by the approximation from multi-term disjunctive cuts than from L&P cuts. The detail time data is shown in Table-3.7 where the blue part is represented as **Time spent on decomposition**. Most instances (9 out of 15) shows multi-term disjunctive cut (setting 2) are better in warm-starting the L2 algorithm excluding the cut generating time.



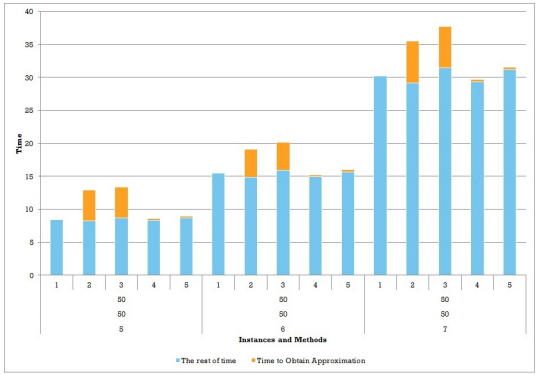
(a) sslp-m-25-50



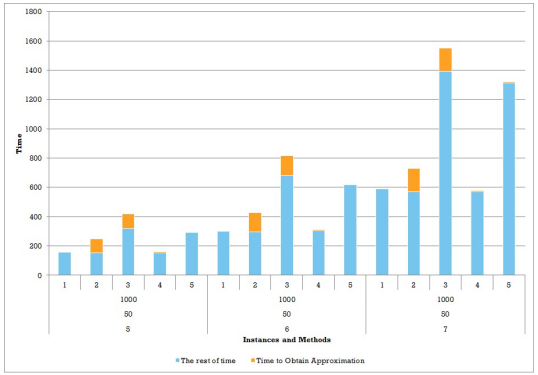
(b) sslp-m-25-100



(c) sslp-m-25-1000



(d) sslp-m-50-50



(e) sslp-m-50-1000

Figure 3.1: The details of total solution time

$k \leq 1$	Time spent on CGLP (orange)		Time spent on decomposition (blue)	
Settings	2	4	2	4
sslp_5_25_100	0.37	0.27	2.72	2.72
sslp_5_25_1000	3.21	2.49	21.55	21.59
sslp_5_25_50	0.20	0.12	1.29	1.29
sslp_5_50_1000	97.55	5.53	151.78	153.23
sslp_5_50_50	4.70	0.27	8.23	8.32
sslp_6_25_100	0.39	0.29	6.51	6.45
sslp_6_25_1000	4.48	2.88	60.38	60.99
sslp_6_25_50	0.21	0.14	3.07	3.12
sslp_6_50_1000	131.02	6.47	296.42	303.89
sslp_6_50_50	4.21	0.31	14.89	14.92
sslp_7_25_100	0.45	0.32	14.68	14.57
sslp_7_25_1000	6.69	3.27	152.12	151.50
sslp_7_25_50	0.26	0.16	8.10	8.04
sslp_7_50_1000	157.80	7.46	568.67	569.68
sslp_7_50_50	6.32	0.37	29.19	29.32
Average	26.24	2.15	83.85	84.60

Table 3.7: Two approximation methods detailed time comparison (secs)

3.5 Conclusion

We proposed a finite algorithm for obtaining the approximation of convex hull of mixed-integer solutions. It uses multi-term disjunctive cus which are derived from disjunctions formulated from the leaf nodes of a Branch and Bound tree. The method is investigated using computations with both wait-and-see as well as here-and-now formulations of stochastic programming problems. It is also compared with a competing algorithm that uses lift-and-project cuts to get the approximation. Although it has been shown for some cases to incur infinite loops, our experiment shows for both wait-and-see and here-and-now situations, the approximations obtained from the algorithm with lift-and-project cuts is better in terms of continuing optimization to optimality and warming-up the subproblem solution in the L2 decomposition algorithm. At the same time, the algorithm with multi-term disjunctive cuts

needs fewer cuts to generate the same or better quality of approximations but takes a longer time to derive the cut.

Chapter 4: Computing with Multi-term Disjunctions under Cutting Plane Tree Framework

4.1 Introduction

Mixed-integer linear programming (MILP) problems have been studied for decades. Numerous theories and algorithms have been developed. However, the knowledge-base is far from clear to perform post-optimality analysis for MILP problems [Gre98]. This is a major road-block for developing solution algorithms for two-stage stochastic mixed-integer programs (SMIP). This chapter is a follow-on to two recent papers [CKS11] and [CKS12]. In [CKS11] the authors developed a cutting plane algorithm (called CPT algorithm) for solving general mixed-integer linear programming problems (MILP-G) using disjunctive cuts and it is proved to a finite procedure even for instances from [OM01] and [CKS90] which are not finite if solved only by splitting disjunctions. In [CKS12] computational experiments are conducted for CPT algorithms with different normalization in the cut generation LP (CGLP).

In the current chapter, we continue on the line of algorithmic work for CPT algorithm. We compare CPT algorithm with different variable splitting rules. We also develop some specialized methods for solving multi-term CGLP arising in the CPT algorithm. Computational comparisons are given within CPT algorithm framework for these methods. There are

methods solving multi-term CGLP in the literature. In [BP02], the authors proposed four methods to solve multi-term CGLP and their computational results suggest Benders' decomposition algorithm is faster over the other three methods. Nevertheless, the experiments reported in [BP02] were not intended to test the effectiveness of multi-term disjunctions in solving MILP-G instances. In [PB01a], the authors proposed methods using simplex tableau methods to solve two-term CGLP. Although it is not for multi-term CGLP solving, it motivates us to find similar methods for multi-term CGLP.

This chapter is organized in the following order: In section 4.2, we will briefly introduce CPT algorithm. Based on the CPT framework, section 4.3 will state three variables splitting strategies with comparisons using computational experiment. In section 4.4, we will study methods for solving multi-term CGLPs. In section 4.5, we continue the experiments with multi-term CGLP and test them in solving MILP instances. Finally, we summarize results and suggest future research directions.

4.2 CPT cuts

Consider a mixed-integer linear programming problem with n variables

$$\min_{x \in X} c^{\mathsf{T}} x \tag{4.1}$$

where $X = \{Ax \geq b, x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n-n_1}\}$ and first n_1 variables are required to be integer.
A is a $m \times n$ matrix and the bounding constraints $\{l_j \leq x_j \leq u_j \text{ for } j = 1...n\}$ (if exist) are assumed to be included in X. Let the LP relaxation of X be

$$X_L = \{Ax \geq b, x \in \mathbb{R}^n\}. \tag{4.2}$$

The multi-term disjunctive cuts are generated by considering a disjunctive relaxation of X:

$$X_D = \bigcup_{t \in T} (X_L \bigcap Q_t), \tag{4.3}$$

where $Q_t, t \in \{1, ..., T\}$ are a series of bounding constraints that are disjoint from each other.

$$Q_t = \{x | L_t \leq x \leq U_t\}. \tag{4.4}$$

In the CPT algorithm, Q_t's are maintained by a tree structure $\mathcal{T}$ with each Q_t being the bounding constraints for a leaf node. The set X_D (in Equations-4.3) is a cone. To truncate this feasible set and obtain a bounded solution, various normalization constraints have been studied in the literature ([Bal79], [BB09], [BP02] and [BCC93]). In [CKS12], the author compared two normalization schemes under the CPT algorithm. Our experiment uses the minimum one norm cut (m1nc) version [Cad10] of normalization although others are clearly possible. The m1nc normalization formulation is:

$$\begin{aligned} \textbf{Minimize} \quad & \sum_j |\pi_j| \\ \textbf{s.t.} \quad & \pi - A^\top \lambda_t - \mu_t + \nu_t = 0, \quad \text{for } t \in \{1, ..., T\} \\ & b^\top \lambda_t + L_t^\top \mu_t - U_t^\top \nu_t \geq \pi^\top x^k + 1, \quad t \in \{1, ..., T\} \\ & \lambda_t \geq 0, \mu_t \geq 0, \nu_t \geq 0 \quad t \in \{1, ..., T\}, \end{aligned} \tag{4.5}$$

where x^k is the fractional solution to be cut off. The cut derived is:

$$\pi^\top (x - x^k) \geq 1.$$

Detailed description and convergence proof of algorithms using CPT cuts to solve a general MILP is in [CKS11]. We present a simplified version of the algorithm in Table 4.1.

4.3 Computations with different variable splitting strategies

As CPT algorithm (shown in Table 4.1) proceeds, CPT tree grows as needed. Same as how B&B tree grows in the B&B method, there are various approaches for a CPT tree to grow by splitting variables (Table 4.1- Step 4) and the way to split variables can influence

- Initialize.: Set iteration $k = 1$, LP relaxation $X_L^k \longleftarrow X_L$. Let tree $\mathcal{T}^k = \{root\}$ with leaf nodes number $T = 1$.
- While(true):
  1. Solve $\min_{x \in X_L^k} c^\top x$ and get solution x^k.
  2. If $x^k \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n-n_1}$, BREAK.
  3. Else, for each fractional x_j^k where $j \in \{1...n_1\}$:
     (a) Formulate CGLP using $\mathcal{T}^k, x_j^k, X_L^k$.
     (b) Solve CGLP and get a CPT cut.
  4. Split nodes in $\mathcal{T}^k$.
  5. Update X_L^k with new CPT cuts.
- End while

Table 4.1: CPT Algorithm

the performance of the algorithm. In this section, we are going to examine the impact of different variable splitting strategies on the CPT algorithm. The strategy used in [CKS12] is called conservative splitting. As shown in Table 4.2, conservative splitting rule splits a node in the CPT tree only when a variable turns from fractional number in previous LP relaex solution vector to integer in the current solution vector. Apparently, one could allow more aggressive variable splitting strategy. We propose two strategies which are more aggressive: maximum gap splitting (Table 4.3) and aggressive splitting (Table 4.4).

Maximum gap splitting makes splits a little bit more than conservative since it allows splitting in each iteration. For aggressive splitting, it is based on maximum gap splitting and splits variables also when the current solution is not belonged to any leaf nodes of cutting plane tree. In this situation, the variables chosen to split are based on vertices x^{kt} calculated

Conservative Splitting:

- Splitting conditions:
  - x^k belongs to a leaf node of CPT tree.
  - $\exists\ j \in \{1...n_1\}$ such that x_j^{k-1} is not integer while x_j^k is integer.
- Variable selection:
  - The first variable j that satisfies splitting condition

Table 4.2: Conservative Splitting

Maximum Gap Splitting:

- Splitting conditions:
  - x^k belongs to a leaf node of CPT tree.
- Variable selection:
  - Suppose $x^k \in Q_t$, $f_l^k(j) = \min\{c^\top x | x \in X_L^k \bigcap Q_t \bigcap \{x \leq \lfloor x_j \rfloor\}\}$ and $f_r^k(j) = \min\{c^\top x | x \in X_L^k \bigcap Q_t \bigcap \{x \geq \lceil x_j \rceil\}\}$, the splitting variable is selected with the maximum gap closure: $\arg\min_{j \in \{1..n_1\}}\{\max(f_l^k(j), f_r^k(j))\}$.

Table 4.3: Maximum Gap Splitting

from the dual solution of the multi-term CGLP. Detailed description of these vertices is shown in Appendix C.

We design a computational experiment to compare these variables splitting rules under CPT algorithm. The goal of the experiment is to examine the impact of tree exploration and the progress of CPT algorithm. The instances we use are from MIPLIB 3.0 and MIPLIB 2003 libraries. A full description of the instances is shown in Appendix D. The experiment

Aggressive Splitting:

- If x^k belongs to a leaf node of CPT tree, the rules are same as Maximum Gap Splitting.
- Else for each vertices supporting the CPT cut, choose a variable to split.
  - Variable selection:
    - For vertex $x^{kt} \in Q_t$, the first variable that is fractional is selected.

Table 4.4: Aggressive Splitting

is conducted on a machine with Quad Xeon E5430 /2.66GHz/32GB RAM under Linux. All linear programming problems are solved by default CPLEX (v. 12.2) LP solver. The stopping rule of the algorithm is the same as in [CKS12] (i.e. $\min\{x_j^k - \lfloor x_j^k \rfloor, \lceil x_j^k \rceil - x_j^k\} \leq \epsilon$). We set $\epsilon = 10^{-6}$. All instances are given one-hour time limit to solve. To make a more thorough comparison, we also include a "no splitting" rule which does not use CPT tree to record any splits on variables. It basically adds "round of" elementary cuts in each iteration. Following this rule, the CGLP only contains one-term or two-term disjunctions depending on each term's feasibility.

The results are shown in Table 4.5 for MILP-B and Table 4.6 for MILP-G. The instances with bold fonts are with greater than 1000 number of variables. Time(secs), Disj, #Iter, #Cuts, GapCl denote the solving time in seconds (>3600 means solving time exceeds 3600 secs), the number of disjunctions (T) at termination, the number of iterations of CPT , the number of cuts added in total and the percentage of integrality gap closed.

As we can see from the result, in Table 4.5, for instances with relatively small number of variables, more aggressive variable splitting obtains better integrality gap closure (i.e., misc03

and stein15). For relatively large instances, the "no splitting" rule performs better. Similar phenomena is captured in Table 4.5. On average, the disjunctions size (T) on termination increases from left to right of the table which conforms to our design intention of exploring the tree more aggressively. However, the average gap closure does not increase. At the same time, the average number of iterations and the average number of cuts added also decreases as the algorithm explore CPT tree more aggressively. This implies that CPT algorithm is slowed down. Because CPT tree is explored more aggressively, the size of disjunctions is also growing faster. So the multi-term CGLP used in the CPT algorithm is becoming increasingly time-consuming to solve. If there are methods that can reduce the impact of disjunction size to the solution of multi-term CGLP, CPT algorithm may perform better.

Instances	No Splitting					Conservative Splitting					Maximum Gap Splitting					Aggressive Strong Branching (First both) Limited				
Problem	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl
aflow30a	>3600	1	27	2018	77.99%	>3600	2	27	1924	75.35%	>3600	4	24	1669	72.24%	>3600	22	14	823	58.87%
danoint	>3600	1	28	1276	3.65%	>3600	3	26	1163	2.61%	>3600	4	20	803	1.74%	>3600	13	12	443	1.74%
dcmulti	590.71099	1	208	1786	100.00%	1326.6	3	30	1458	100.00%	2222	5	26	1236	100.00%	>3600	39	15	702	94.14%
egout	0.88	1	6	85	100.00%	2.61	2	7	109	100.00%	3.394	6	8	111	100.00%	3.1	8	8	111	100.00%
enigma	>3600	1	622	10136	NO GAP	>3600	48	153	1584	NO GAP	>3600	40	121	1520	NO GAP	>3600	135	77	909	NO GAP
fixnet6	>3600	1	53	2493	97.98%	>3600	5	23	1032	90.17%	>3600	9	14	652	87.17%	>3600	11	11	505	83.60%
glass4	>3600	1	53	6710	0.00%	>3600	14	30	144	58.00%	>3600	11	17	2106	0.00%	>3600	4	5	261	0.00%
lseu	>3600	1	154	5520	73.70%	>3600	3	154	5300	63.14%	>3600	7	156	5437	63.56%	>3600	316	34	749	51.65%
markshare1	>3600	1	1053	6498	0.00%	>3600	80	92	550	0.00%	>3600	83	88	518	0.00%	>3600	74	71	424	0.00%
markshare2	>3600	1	861	6111	0.00%	>3600	68	77	539	0.00%	>3600	66	71	487	0.00%	>3600	57	57	390	0.00%
mas74	>3600	1	402	11570	9.95%	>3600	2	376	10443	9.93%	>3600	6	286	7584	9.76%	>3600	157	101	1602	13.07%
mas76	>3600	1	333	8855	10.52%	>3600	3	263	6198	10.26%	>3600	8	183	2716	9.02%	>3600	171	106	1516	15.10%
misc03	>3600	1	131	11095	79.54%	>3600	3	136	11484	60.76%	>3600	16	120	9992	59.52%	2503	77	78	4302	100.00%
misc07	>3600	1	75	10171	17.26%	1027.88	3	43	4741	12.04%	>3600	3	59	7450	14.46%	>3600	61	55	5198	24.91%
mod008	>3600	1	67	2070	31.81%	>3600	2	73	2191	32.34%	>3600	4	69	2101	31.92%	>3600	70	15	205	20.01%
modglob	>3600	1	13	609	99.76%	3139	5	22	829	100.00%	>3600	7	12	521	98.76%	>3600	11	10	432	97.20%
opt1217	>3600	1	137	6040	0.90%	>3600	14	33	1014	0.53%	>3600	10	72	3745	1.28%	>3600	28	18	539	0.53%
p0033	>3600	1	351	8250	99.62%	>3600	4	355	8608	94.44%	>3600	10	268	6093	100.00%	>3600	171	99	2096	99.98%
p0201	>3600	1	34	3277	84.73%	>3600	2	33	3398	87.37%	>3600	3	32	3215	86.66%	>3600	63	15	1067	62.02%
p0282	>3600	1	67	3841	98.40%	>3600	2	76	4164	98.44%	>3600	5	62	3075	98.24%	>3600	176	20	622	96.68%
p0548	180.98	1	28	986	100.00%	921	3	52	2180	100.00%	>3600	9	18	1170	99.98%	>3600	18	15	925	99.82%
pk1	>3600	1	495	8506	0.00%	>3600	31	48	718	0.00%	>3600	28	61	939	0.00%	>3600	81	31	473	0.00%
pp08a	>3600	1	62	2788	99.60%	>3600	2	62	2800	99.54%	>3600	3	68	3071	99.69%	>3600	78	16	665	98.50%
pp08aCUTS	>3600	1	60	2743	98.84%	>3600	2	61	2790	98.80%	>3600	5	57	2615	98.82%	>3600	27	17	662	96.04%
qiu	>3600	1	21	910	80.31%	>3600	1	21	932	80.42%	>3600	2	22	960	80.38%	>3600	16	10	421	74.01%
rgn	>3600	1	93	3634	63.15%	>3600	3	93	3781	64.81%	>3600	3	90	3685	64.12%	>3600	183	29	894	58.80%
set1ch	316.36	1	31	1511	100.00%	1074	5	31	1743	100.00%	1635	8	33	1863	100.00%	>3600	17	14	1224	97.54%
stein15	>3600	1	781	11685	43.99%	>3600	3	767	11472	43.14%	>3600	3	832	12453	43.02%	49.1129	135	39	530	100.00%
stein27	>3600	1	284	7606	28.51%	>3600	3	299	8012	26.67%	>3600	5	304	8134	28.65%	>3600	546	51	1296	48.09%
stein45	>3600	1	120	5262	7.38%	>3600	2	125	5477	7.19%	>3600	7	120	5239	4.61%	>3600	229	27	1100	14.56%
vpm1	65.338	1	31	871	100.00%	387	2	70	2092	100.00%	1079.08	12	52	1564	100.00%	>3600	67	18	485	76.87%
vpm2	>3600	1	68	3375	90.59%	>3600	2	69	3302	90.60%	>3600	5	61	3112	90.78%	>3600	66	17	659	78.10%
Average		1	210.91	4946.50	58.01%		10.22	116.47	3505.38	58.28%		12.41	107.06	3307.38	56.27%		97.72	34.53	1007.19	56.83%
10teams	>3600	1	12	1749	0.00%	>3600	1	11	1575	0.00%	>3600	4	6	701	0.00%	>3600	5	5	614	0.00%
air03	123.619	1	2	36	100.00%	152.848	1	2	36	100.00%	162.967	2	2	36	100.00%	243.99	2	2	36	100.00%
air04	>3600	1	2	19	1.07%	>3600	1	2	15	1.07%	>3600	2	2	15	1.07%	>3600	2	2	15	1.07%
air05	>3600	1	2	139	13.32%	>3600	1	2	124	13.14%	>3600	2	2	100	11.82%	>3600	2	2	100	11.82%
cap6000	>3600	1	48	848	63.88%	>3600	3	25	313	59.68%	2373.81	2	35	528	61.56%	>3600	25	12	101	48.72%
dano3mip	>3600	1	2	3	0.00%	>3600	1	2	3	0.00%	>3600	2	2	2	0.00%	>3600	1	1	0	0.00%
fast0507	>3600	1	2	4	0.00%	>3600	1	2	4	0.00%	>3600	2	2	3	0.00%	>3600	1	1	0	0.00%
fiber	>3600	1	35	3121	91.29%	>3600	2	35	3188	90.65%	>3600	5	19	1271	68.90%	>3600	19	11	532	48.58%
harp2	>3600	1	12	0	0.00%	>3600	1	12	0	0.00%	>3600	2	11	0	0.00%	>3600	2	12	10	0.00%
khb05250	121.45	1	36	369	100.00%	210.497	3	50	476	100.00%	>3600	8	27	303	99.91%	>3600	2	12	10	99.39%
l152lav	>3600	1	20	1773	27.82%	>3600	1	20	1788	28.59%	>3600	3	18	1463	26.66%	>3600	10	9	595	17.89%
misc06	>3600	1	108	2166	65.53%	>3600	8	15	223	37.02%	>3600	12	14	196	37.00%	>3600	14	14	198	37.03%
mitre	>3600	1	5	2568	17.79%	>3600	1	4	1637	10.26%	>3600	3	3	1308	9.64%	>3600	3	3	1122	8.09%
mkc	>3600	1	13	1323	33.61%	>3600	3	7	521	19.60%	>3600	6	6	414	44.38%	>3600	6	6	398	44.38%
mod010	>3600	1	22	1793	42.73%	>3600	2	23	1371	31.22%	>3600	3	23	1826	41.36%	>3600	26	12	630	22.60%
mod011	>3600	1	18	642	6.95%	>3600	2	17	567	6.77%	>3600	2	23	855	7.31%	>3600	16	10	270	5.26%
nw04	>3600	1	9	100	0.27%	>3600	3	7	66	0.24%	>3600	5	8	75	0.24%	>3600	7	7	64	0.20%
p2756	>3600	1	60	2306	99.97%	>3600	3	59	2229	100.00%	>3600	10	17	956	97.88%	>3600	17	14	774	88.53%
rentacar	>3600	1	22	166	82.84%	>3600	2	15	128	76.42%	>3600	2	2	5	0.20%	>3600	6	7	57	61.07%
seymour	>3600	1	4	1704	50.35%	>3600	2	3	1119	30.57%	>3600	2	4	1632	50.92%	>3600	4	4	1553	49.11%
swath	>3600	1	7	420	0.40%	>3600	2	5	220	0.21%	>3600	3	8	430	0.35%	>3600	3	3	110	0.12%
Average		1	21.00	1011.86	37.99%		2.10	15.14	743.00	33.59%		3.90	11.14	577.10	31.39%		8.24	7.10	342.33	30.66%
Overall average		1	135.66	3387.49	49.92%		7.00	76.32	2410.85	48.31%		9.04	69.06	2225.57	46.22%		62.26	23.66	743.75	46.26%

Table 4.5: Variable Splitting Rule Comparison - Mixed Binary Instances

Instances	No Splitting					Conservative Splitting					Maximum Gap Splitting					Aggressive Strong Branching (First both) Limited				
Problem	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl
Bell3a	>3600	1	794	6286	73.23%	>3600	12	125	1184	74.62%	>3600	12	172	1913	74.54%	>3600	36	37	242	72.83%
bell5	>3600	1	320	7666	97.56%	>3600	3	130	3098	97.43%	>3600	10	92	2066	97.41%	>3600	61	23	399	96.04%
blend2	>3600	1	60	2934	45.90%	>3600	5	83	3325	45.90%	>3600	2	64	3441	48.12%	>3600	64	30	895	32.17%
flugpl	>3600	1	1396	13422	29.73%	>3600	4	1014	10013	26.71%	>3600	6	1173	11614	27.54%	2468	142	157	1499	98.92%
gen	>3600	1	64	2355	98.59%	>3600	2	57	2439	99.10%	>3600	6	22	1049	87.59%	>3600	40	19	906	89.08%
gt2	>3600	1	412	5415	100.00%	>3600	13	79	1452	93.74%	>3600	7	70	2020	95.71%	>3600	100	22	431	93.04%
noswot	>3600	1	224	10340	NO GAP	>3600	7	57	3191	NO GAP	>3600	17	30	1293	NO GAP	>3600	35	20	962	NO GAP
rout	>3600	1	42	3534	9.71%	>3600	2	42	3422	8.69%	>3600	2	42	3526	9.69%	>3600	53	28	1143	18.81%
timtab1	>3600	1	28	3634	47.53%	>3600	2	30	3833	47.08%	>3600	2	30	3842	48.43%	>3600	49	14	1673	43.43%
timtab2	>3600	1	16	3609	35.54%	>3600	2	17	3665	35.84%	>3600	3	13	2882	30.25%	>3600	13	10	2053	27.49%
Average		1	348	5428.33	59.75%		5	175.22	3603.44	58.79%		5.56	186.44	3594.78	57.70%		62	37.78	1026.78	63.54%
arki001	>3600	1	20	888	13.76%	>3600	1	20	899	13.76%	>3600	3	18	896	17.31%	252	4	4	225	11.29%
gesa2	2781.82	1	46	2473	100.00%	>3600	3	26	1607	98.40%	>3600	7	22	1367	98.74%	>3600	17	12	666	83.06%
gesa2_o	>3600	1	31	2945	99.60%	>3600	4	14	1176	81.48%	>3600	6	20	1792	98.84%	>3600	73	16	1499	89.65%
gesa3	865.211	1	29	1113	100.00%	>3600	3	18	1386	99.87%	>3600	6	22	1102	100.00%	>3600	14	11	939	99.30%
gesa3_o	957.6859999	1	20	1205	100.00%	>3600	3	11	1293	99.26%	>3600	7	18	1192	99.98%	>3600	12	11	1144	99.60%
qnet1	>3600	1	19	1651	45.98%	>3600	2	15	1190	41.56%	>3600	2	18	1628	45.37%	>3600	52	15	1282	42.86%
qnet1_o	>3600	1	24	1924	59.04%	>3600	3	14	846	53.87%	>3600	2	24	1884	59.11%	>3600	83	18	1255	56.92%
n4-3	>3600	1	11	550	81.29%	>3600	2	9	413	74.67%	>3600	4	9	391	73.19%	>3600	5	5	190	56.34%
mik.250-1-100.1	>3600	1	40	4492	91.28%	>3600	4	43	4758	92.34%	>3600	5	38	4037	90.03%	>3600	83	21	2071	87.44%
dfn-gwin-UUM	0.194999933	1	1	0	0.00%	0.192000151	1	1	0	0.00%	0.25	1	1	0	0.00%	0.19	1	1	0	0.00%
Average		1	24.1	1724.1	69.10%		2.6	17.1	1356.8	65.52%		4.3	19	1428.9	68.26%		34.4	11.4	927.1	62.65%
Overall average		1	179.85	3821.8	64.67%		3.9	90.25	2459.5	62.33%		5.5	94.9	2396.75	63.26%		46.85	23.7	973.7	63.07%

Table 4.6: Variable Splitting Rule Comparison- Mixed Integer Instances

4.4 Methods for computing with multi-term CGLPs

Last section's result shows specialized methods for solving multi-term CGLP are necessary. So we investigate some methods for solving multi-term CGLP and divide them into two categories. The methods in the first category utilizes special structure of the constraint matrix to reduce the time complexity. Since CGLP with multi-term disjunctions has the block angular structure (will be shown in Section 4.4.1), special methods can be applied to utilize the structure. The method in the other category is to approximate the original constraint set used in the CGLP using less number of constraints. Both approaches actually work for reducing the CGLP's solution time based on our experiment result to be shown in Section 4.5. To illustrate these methods, the remaining of this section is divided into three parts with each part describing one method.

4.4.1 Basis factorization

The dual of (Equations-4.5) is:

$$\textbf{Maximize} \quad \sum_t z_t \tag{4.6a}$$

$$\textbf{s.t.} \quad Ay_t - \bar{b}z_t \geq 0 \tag{4.6b}$$

$$y_t - \bar{L}_t z_t \geq 0 \tag{4.6c}$$

$$y_t - \bar{U}_t z_t \leq 0 \tag{4.6d}$$

$$-1 \leq \sum_t y_{t,j} \leq 1 \tag{4.6e}$$

$$z_t \geq 0. \tag{4.6f}$$

It has the block-angular structure as shown in (Figure-4.1) where the constraints matrix with surplus variables and the objective are in the first row of the matrix and I_m (or I_n) is a $m \times m$ (or $n \times n$) identity matrix given A is a $m \times n$ matrix. and because of this structure, applying

specialized basis factorization schemes could be beneficial for large scale CGLP instances (based on large number of disjunctions). Basis factorization method was first proposed by [Dan55] and extended by [Win74] for solving block-angular structured linear programming problems. [Str74] and [BL97] applied it in solving stochastic linear programming problem.

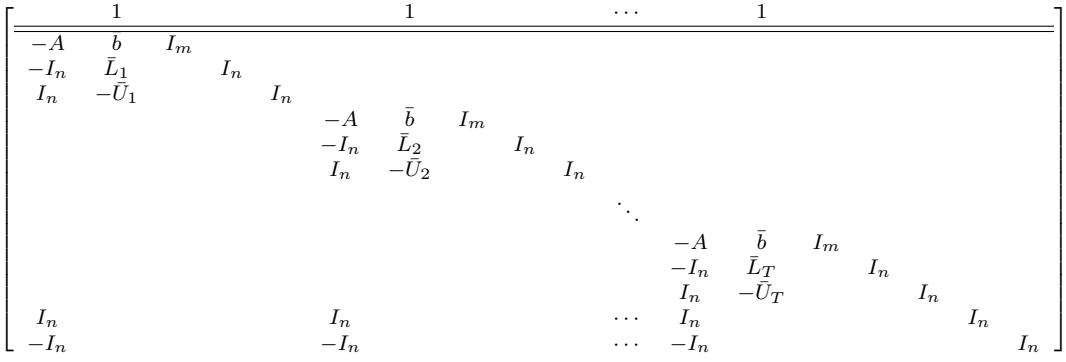


Figure 4.1: Block-angular structure

To apply the basis factorization method, for a linear program like (4.6), we denote the basis of (4.6) as $\widehat{B}$ and its t-th sub block as $[B_t, E_t]$, where B_t is a feasible basis for the t-th subsystem (4.6b). E_t is the rest part of the rows that is in the basis $\widehat{B}$. Therefore, $\widehat{B}$ can be rearranged as

$$\begin{bmatrix} B & E \\ C & D \end{bmatrix},$$

where C and D are the coupling constraints parts (4.6c) and are separated corresponding to columns B and E.

The whole basis has the structure:

$$\widehat{B} = \begin{bmatrix} B_1 & & & & E_1 & & & \\ & \cdot & & & & \cdot & & \\ & & \cdot & & & & \cdot & \\ & & & \cdot & & & & \cdot \\ & & & B_t & & & & E_t \\ C_1 & \cdot & \cdot & C_t & D_1 & \cdot & \cdot & D_t \end{bmatrix} \tag{4.7}$$

Basis factorization method works directly on the basis matrix. It is incorporated into the simplex method. If we denote the dual value associated with basis $\widehat{B}$ as

$$\begin{bmatrix} \beta \\ \pi \end{bmatrix} = \begin{bmatrix} \beta_1 \\ \cdot \\ \cdot \\ \cdot \\ \beta_T \\ \pi \end{bmatrix}. \tag{4.8}$$

These dual values can be calculated by solving:

$$\widehat{B}^\top \begin{bmatrix} \beta \\ \pi \end{bmatrix} = \begin{bmatrix} B^\top & C^\top \\ E^\top & D^\top \end{bmatrix} \begin{bmatrix} \beta \\ \pi \end{bmatrix} = \begin{bmatrix} \rho^1 \\ \rho^2 \end{bmatrix}, \tag{4.9}$$

where $\begin{bmatrix} \rho^1 \\ \rho^2 \end{bmatrix}$ are the rearrangement of objective coefficients (4.6a) according to columns B and E.

By solving (4.9), we get

$$\pi = (D^\top - E^\top (B^\top)^{-1} C^\top)^{-1} (\rho^2 - E^\top (B^\top)^{-1} \rho^1), \tag{4.10}$$

and for $t = 1, ..., T$, we have

$$\beta_t = (B_t^\top)^{-1} (\rho_t^1 - C_t^\top \pi). \tag{4.11}$$

One step further

$$\pi = (D^\top - \sum_{t=1}^{T} E_t^\top (B_t^\top)^{-1} C_t^\top)^{-1} (\rho^2 - \sum_{t=1}^{T} E_t^\top (B_t^\top)^{-1} \rho_t^1). \tag{4.12}$$

Notice that to obtain β and π, based on (4.12,4.11) we only need to know the inverse of matrix $B_t^\top$ with size $(n \times n)$ and $(D^\top - (B^\top)^{-1} E^\top C^\top)$. Compared to calculating the inverse of entire matrix with $(Tm + n)^2$ elements, one only needs to work with matrices with $(Tm^2 + n^2)$ elements in total.

The next step of simplex algorithm is to find the leaving column. Suppose the incoming column is $\begin{pmatrix} u \\ v \end{pmatrix}$ and we need to solve $\widehat{B}d = \begin{pmatrix} u \\ v \end{pmatrix}$ to get the leaving column. We can factorize the basis the same way and get the directions for each box $t = 1, ..., T$

$$d_{\lambda_t} = B_t^{-1}(u_t - C_t d_\pi) \tag{4.13}$$

$$d_\pi = (D - \sum_{t=1}^{T} B_t^{-1} E_t C_t)^{-1} (v - \sum_{t=1}^{T} B_t^{-1} E_t u_t) \tag{4.14}$$

The rest of the algorithm stays the same as in simplex method.

We implemented this method for solving multi-term CGLPs. We use instance "rout" from MIPLIB 2003 and we sampled CGLPs upto 16 disjunctions encountered during solving the instance using CPT algorithm. Because existing commercial solvers do not provide interfaces that allow users to specialize the solution of linear systems solved within the simplex method, we develop is a "home-grown" implementation of the simplex method. We use Bartels-Golub as the LU-update algorithm and implement it in Matlab. Our preliminary computational result is shown in Figure-4.2 where the lines show the solution time curve of CGLP as the disjunctions size getting larger (blue for normal simplex and green for factorized simplex). From the result, we can see that factorized simplex method is faster than traditional revised simplex method when the partition size is greater than four. And the solution time gap is getting larger as the disjunctions size grows. Thus basis factorization method is better than the normal simplex method in this experiment

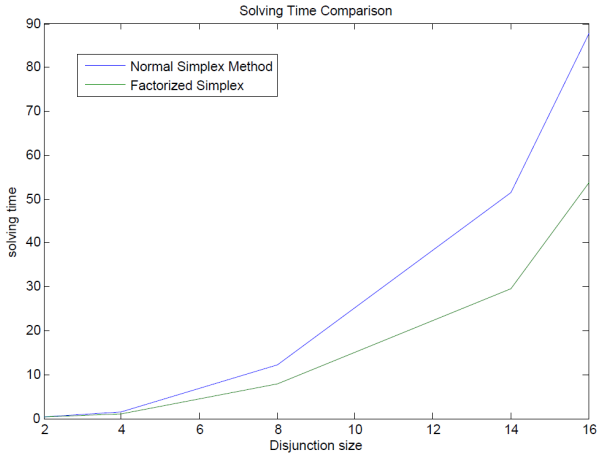


Figure 4.2: A comparison between factorized simplex and normal simplex method

4.4.2 Decomposition methods

An alternative method of decomposing the matrix is to use Benders' decomposition [Ben62]. [PB01b]'s result shows using Benders' decomposition with all extreme points that are encountered during optimizing the subproblems being added to the master problem achieves the best performance. But their results were not designed towards measuring solution time for MILP instances. We propose a slightly modified version of this iterative method with two newly generated terms included in master problem:

$$\textbf{Minimize} \quad \sum_j |\pi_j|$$

$$\textbf{s.t.} \quad \pi_j - A_j^\top \lambda_t - \mu_t + \nu_t = 0 \quad \text{for } j = 1...n, \text{ and } t \in \{tr, tl\}$$

$$b^\top \lambda_t + L_t^\top \mu_t - U_t^\top \nu_t \geq \pi^\top x^k + 1 \quad \text{for } t \in \{tr, tl\}$$

$$\lambda_t \geq 0, \mu_t \geq 0, \nu_t \geq 0 \quad t \in \{tr, tl\}.$$

$$\pi^\top x_s \geq \pi^\top x^* + 1 \quad \text{for } s \in S$$

$$\pi^\top x_r \geq 0 \quad \text{for } r \in R$$

where S and R are the set of extreme points and rays from $\bigcup_{t\in\{1,\dots,T\}\setminus\{tr,tl\}}(X_L \bigcap Q_t)$. tr, tl are two disjunctive sets that are generated most recently. Other extreme points and rays are generated from other disjunctive set by checking the subproblem $X_L \bigcap Q_t$:

$$\begin{aligned} \textbf{Minimize} \quad & \pi^\top x \\ \textbf{s.t.} \quad & Ax \geq b \\ & L_t \leq x \leq U_t. \end{aligned} \tag{4.15}$$

By solving (4.15), if the subproblem is bounded and there are extreme points x such that $\pi^\top x \leq \pi^\top x^k + 1$, these x are added to S. If the subproblem is unbounded and there are rays r such that $\pi^\top r \leq 0$, these r are added to R. The algorithm stops when all the subproblems has no legitimate points or rays to add. This iterative method is denoted as: modified Benders' decomposition. We devise a computational experiment to compare different versions of Benders' methods:

- pure Benders' decomposition (Method 2),
- modified Benders' decomposition (Method 3),
- and modified Benders' decomposition with all encountered extreme points added (Method 4).

Together with CGLP solved by CPLEX (Method 1), basis factorization (Method 5) and our self-written revised simplex method (Method 6), six methods are compared. The experiment is conducted by sampling 5 CGLP instances for each disjunction size ranging from 1 to 13, which are encountered during solving instance "noswot" under CPT framework. All Benders' methods are integrated with trusted region method. The result is shown in Figure 4.3. Each dot in the figure denotes the averaged solution time of 5 CGLP instances which are sampled from encountered CGLPs during solving "noswot". From the result, we can see, Method

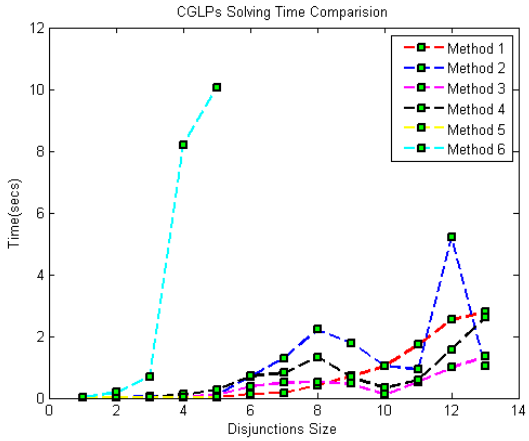


Figure 4.3: Comparison of solving time (secs) for multi-term CGLPs with 6 methods

3 is more stable and its CGLP solution time increases the slowest among four methods as the number of disjunctions size getting larger. Although in Method 4 extra extreme points are added to the master problem, these points are not worth to incorporate from the result. Method 6 performs about the same as Method 3 for the first 5 disjunctions, from the Table 4.7 we can see it is not stable enough and a lot of instances incur numerical difficulties due to LU factorization package.

4.4.3 Feasible region approximation

$X_L \bigcap Q_t$ can be approximated using less number of constraints and we can use the approximation to derive CPT cuts. In our implementation , we use the original constraints set with an extra constraint to represent X_L for box Q_t.(Note that in our previous implementation, we use X_L^k which also contains CPT cuts generated up to iteration k). The additional constraint is obtained by aggregating constraints in X_L using its dual multiplier. The dual solution is attained when checking the feasibility of the box. Mathematically, when box t is

Disjunction Size	Number of Instances	Revised Simplex	Basis Factorization
1	3	3	3
2	5	5	1
3	5	3	1
4	4	4	4
5	5	4	4
6	5	0	0
7	5	0	0
8	5	0	0
9	5	0	0
10	5	0	0
11	5	0	0
12	5	0	0
13	5	0	0
14	5	0	0

Table 4.7: Instances successfully solved by revised simplex method and basis factorization method

created, we check its feasibility by solving:

$$\begin{aligned} \textbf{Minimize} \quad & c^\top x_t \\ \textbf{s.t.} \quad & Ax_t \geq b \\ & L_t \leq x_t \leq U_t, \end{aligned}$$

Or in the dual formulation:

$$\begin{aligned} \textbf{Maximize} \quad & b^\top \lambda_t + L_t^\top \mu_t - U_t^\top \nu_t \\ \textbf{s.t.} \quad & A_j^\top \lambda_t + \mu_t - \nu_t = c_j \quad \text{for } j = 1...n \\ & \lambda_t \geq 0, \mu_t \geq 0, \nu_t \geq 0. \end{aligned}$$

Using λ_t, we can get the aggregated constraint for box t:

$$\bar{a}_t^\top x \geq \bar{b}$$

where

$$\bar{a}_{tj} = A_j^\top \lambda_t \quad \text{for } j = 1...n$$

$$\bar{b}_t = b^\top \lambda_t$$

Using the original constraints set with an extra constraint aggregated constraints as the representation of X_L, CGLP formulation is:

$$\begin{aligned}
\textbf{Minimize} \quad & \sum_j |\pi_j| \\
\textbf{s.t.} \quad & \pi_j - \bar{a}_t^\top \lambda_t - \mu_t + \nu_t = 0 \quad j = 1...n, t = 1, ..., T \\
& \bar{b}_t^\top \lambda_t + L_t^\top \mu_t - U_t^\top \nu_t \geq \pi^\top x^k + 1 \quad t = 1, ..., T \\
& \pi_j - A_j^\top \lambda_t - \mu_t + \nu_t = 0 \quad \text{for } j = 1...n, , t = 1, ..., T \\
& b^\top \lambda_t + L_t^\top \mu_t - U_t^\top \nu_t \geq \pi^\top x^k + 1 \quad \text{for } , t = 1, ..., T \\
& \lambda_t \geq 0, \mu_t \geq 0, \nu_t \geq 0 \quad t = 1, ..., T.
\end{aligned} \tag{4.16}$$

Because each box needs feasibility checking when it is created, the aggregated constraint calculation will not incur additional optimization problem solving.

4.5 Computations

The infrastructure of the experiment stays the same as in Section 4.3. The specialized methods for solving multi-term CGLPs we are comparing are

- directly solved by CPLEX,
- modified Benders' decomposition,
- and aggregated approximation.

We did not include basis factorization in the comparison due to its numerical difficulties. In the implementation, we use aggressive splitting rule across the comparison to. For Benders' decomposition method, we use the default CPLEX LP solver while aggreated approximation uses primal simplex solver. The computational result is shown in Table 4.8 for MILP-B instances and Table 4.9 for MILP-G instances. From the result, we observe the aggregated

approximation method achieves better averaged integrality gap closure than the other two methods. It takes more iterations to run and generates more cuts on average. There are instances that Benders' decomposition performs better(i.e., misc03, mas74, mas76, cap6000) but aggregated approximation is primarily more stable.

Instances	CPLEX					Aggregated Approximation					Modified Benders' Decomposition				
Problem	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl
aflow30a	>3600	22	14	823	58.87%	>3600	23	17	1011	67.84%	>3600	3	3	48	17.32%
danoint	>3600	13	12	443	1.74%	>3600	17	12	474	1.74%	>3600	15	13	514	1.74%
dcmulti	>3600	39	15	702	94.14%	>3600	51	16	765	95.96%	>3600	41	13	626	92.37%
egout	3.1	8	8	111	100.00%	0.27	6	6	83	100.00%	4.2	9	9	145	100.00%
enigma	>3600	135	77	909	NO GAP	>3600	88	75	875	NO GAP	>3600	88	75	875	NO GAP
fixnet6	>3600	11	11	505	83.60%	>3600	103	21	798	91.09%	>3600	4	4	116	18.02%
glass4	>3600	4	5	261	0.00%	>3600	21	21	2589	0.00%	>3600	9	275	103	51.10%
lseu	>3600	316	34	749	51.65%	>3600	391	35	762	69.39%	>3600	326	36	719	55.33%
markshare1	>3600	74	71	424	0.00%	>3600	34	34	198	0.00%	>3600	220	197	1201	0.00%
markshare2	>3600	57	57	390	0.00%	>3600	29	29	199	0.00%	>3600	147	145	1018	0.00%
mas74	>3600	157	101	1602	13.07%	>3600	148	57	1111	12.02%	>3600	296	174	2925	14.55%
mas76	>3600	171	106	1516	15.10%	>3600	145	45	822	12.21%	>3600	300	206	3111	18.09%
misc03	2503	77	78	4302	100.00%	>3600	198	98	6711	93.70%	1256.93	88	85	4666	100.00%
misc07	>3600	61	55	5198	24.91%	>3600	132	51	4710	31.91%	>3600	74	83	7972	26.57%
mod008	>3600	70	15	205	20.01%	>3600	68	15	219	22.56%	>3600	75	17	248	22.21%
modglob	>3600	11	10	432	97.20%	>3600	12	12	540	99.75%	>3600	6	14	228	91.81%
opt1217	>3600	28	18	539	0.53%	>3600	71	33	1003	0.53%	>3600	84	31	980	0.53%
p0033	>3600	171	99	2096	99.98%	480	461	133	2919	100.00%	>3600	204	51	878	99.99%
p0201	>3600	63	15	1067	62.02%	>3600	135	21	1729	70.57%	>3600	42	17	1346	69.51%
p0282	>3600	176	20	622	96.68%	>3600	195	26	974	97.67%	>3600	113	18	565	96.58%
p0548	>3600	18	15	925	99.82%	>3600	15	12	622	99.61%	>3600	60	15	744	98.82%
pk1	>3600	81	31	473	0.00%	>3600	326	53	1111	0.72%	>3600	223	61	970	0.00%
pp08a	>3600	78	16	665	98.50%	>3600	245	27	1105	99.17%	>3600	63	16	634	98.08%
pp08aCUTS	>3600	27	17	662	96.04%	>3600	200	25	1054	98.02%	>3600	6	6	228	73.66%
qiu	>3600	16	10	421	74.01%	>3600	31	11	457	74.04%	>3600	2	3	37	9.91%
rgn	>3600	183	29	894	58.80%	>3600	369	43	1369	66.64%	>3600	156	30	883	54.31%
set1ch	>3600	17	14	1224	97.54%	1655	44	34	1592	100.00%	>3600	5	5	412	82.00%
stein15	49.1129	135	39	530	100.00%	7.82	168	40	556	100.00%	23.07	128	37	507	100.00%
stein27	>3600	546	51	1296	48.09%	>3600	1547	99	2606	52.54%	>3600	499	48	1208	44.38%
stein45	>3600	229	27	1100	14.56%	>3600	625	44	1872	18.13%	>3600	216	29	1183	11.77%
vpm1	>3600	67	18	485	76.87%	>3600	302	45	1334	98.54%	>3600	134	21	587	89.77%
vpm2	>3600	66	17	659	78.10%	>3600	199	41	1740	89.42%	>3600	38	21	866	82.89%
Average		97.72	34.53	1007.19	56.83%		199.97	38.47	1372.19	60.12%		114.81	54.94	1141.97	52.30%
10teams	>3600	5	5	614	0.00%	>3600	6	6	707	0.00%	>3600	3	3	235	0.00%
air03	243.99	2	2	36	100.00%	65.851	2	2	35	100.00%	265.23	2	2	36	100.00%
air04	>3600	2	2	15	1.07%	>3600	2	2	15	1.07%	>3600	2	2	15	1.07%
air05	>3600	2	2	100	11.82%	>3600	2	2	100	11.82%	>3600	2	2	100	11.82%
cap6000	>3600	25	12	101	48.72%	>3600	21	10	74	45.48%	>3600	23	14	98	61.96%
dano3mip	>3600	1	1	0	0.00%	>3600	2	2	2	0.00%	>3600	2	2	2	0.00%
fast0507	>3600	1	1	0	0.00%	>3600	2	2	3	0.00%	>3600	2	2	3	0.00%
fiber	>3600	19	11	532	48.58%	>3600	50	17	1103	64.93%	>3600	32	12	600	50.03%
harp2	>3600	2	12	10	0.00%	1699.547	2	2	30	0.84%	>3600	2	396	1	0.00%
khb05250	>3600	2	12	10	99.39%	>3600	38	16	164	99.88%	>3600	5	5	49	86.31%
l152lav	>3600	10	9	595	17.89%	>3600	39	13	1042	22.92%	>3600	22	10	767	18.76%
misc06	>3600	14	14	198	37.03%	>3600	33	18	202	74.77%	>3600	23	14	176	33.45%
mitre	>3600	3	3	1122	8.09%	>3600	4	6	3383	24.65%	>3600	3	3	800	7.16%
mkc	>3600	6	6	398	44.38%	>3600	19	17	1953	53.13%	>3600	27	19	1576	54.20%
mod010	>3600	26	12	630	22.60%	>3600	41	13	791	29.65%	>3600	25	14	910	28.96%
mod011	>3600	16	10	270	5.26%	>3600	9	8	230	4.76%	>3600	16	10	307	5.22%
nw04	>3600	7	7	64	0.20%	>3600	49	15	283	0.43%	>3600	5	4	44	0.13%
p2756	>3600	17	14	774	88.53%	>3600	22	15	596	97.90%	>3600	4	4	127	3.01%
rentacar	>3600	6	7	57	61.07%	>3600	2	2	5	0.20%	>3600	4	3	28	41.19%
seymour	>3600	4	4	1553	49.11%	>3600	2	2	386	22.22%	>3600	1	1	246	0.00%
swath	>3600	3	3	110	0.12%	2653.13	14	9	786	0.44%	>3600	7	8	408	0.37%
Average		8.24	7.10	342.33	30.66%		17.19	8.52	566.19	31.19%		10.10	25.24	310.86	23.98%
Overall average		62.26	23.66	743.75	46.26%		127.55	26.60	1052.83	48.44%		73.32	43.17	812.66	40.86%

Table 4.8: CPT with various CGLP solving comparison - Mixed Binary Instances

Instances	CPLEX					Aggregated Approximation					Modified Benders' Decomposition				
Problem	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl	Time(secs)	#Disj	#Iter	#Cuts	GapCl
Bell3a	>3600	36	37	242	72.83%	>3600	21	8987	9046	71.89%	>3600	43	74	219	74.30%
bell5	>3600	61	23	399	96.04%	>3600	287	53	899	97.40%	>3600	67	28	528	96.57%
blend2	>3600	64	30	895	32.17%	>3600	70	17	345	26.56%	>3600	70	27	667	39.99%
flugpl	2468	142	157	1499	98.92%	2376	168	153	1449	100.00%	614.94	146	142	1365	100.00%
gen	>3600	40	19	906	89.08%	>3600	66	19	817	89.18%	>3600	12	17	817	87.01%
gt2	>3600	100	22	431	93.04%	>3600	340	47	850	99.00%	>3600	27	15	316	80.96%
noswot	>3600	35	20	962	NO GAP	>3600	138	34	1698	NO GAP	>3600	82	26	1330	NO GAP
rout	>3600	53	28	1143	18.81%	>3600	21	15	610	14.16%	>3600	87	39	1554	22.94%
timtab1	>3600	49	14	1673	43.43%	>3600	90	21	2590	45.41%	>3600	65	15	1815	43.93%
timtab2	>3600	13	10	2053	27.49%	>3600	21	12	2427	33.62%	>3600	8	8	1499	25.75%
Average		62	37.78	1026.78	63.54%		122.2	1036	2114.78	64.14%		58.33	40.56	975.56	63.49%
arki001	252	4	4	225	11.29%	>3600	3	3	87	3.48%	267.613	2	2	50	0.00%
gesa2	>3600	17	12	666	83.06%	>3600	55	16	1013	93.46%	>3600	4	4	114	55.45%
gesa2_o	>3600	73	16	1499	89.65%	>3600	19	14	1190	99.49%	>3600	4	4	149	55.47%
gesa3	>3600	14	11	939	99.30%	>3600	27	27	1083	100.00%	>3600	7	7	525	95.65%
gesa3_o	>3600	12	11	1144	99.60%	>3600	18	14	1154	99.92%	>3600	10	9	896	97.80%
qnet1	>3600	52	15	1282	42.86%	>3600	33	13	964	42.28%	>3600	26	11	804	40.55%
qnet1_o	>3600	83	18	1255	56.92%	>3600	71	14	848	55.98%	>3600	45	13	699	53.88%
n4-3	>3600	5	5	190	56.34%	>3600	9	7	314	71.13%	>3600	2	2	46	27.15%
mik.250-1-100.1	>3600	83	21	2071	87.44%	>3600	102	22	2209	87.10%	>3600	5	5	293	72.02%
dfn-gwin-UUM	0.19	1	1	0	0.00%	0.047999859	1	1	0	0.00%	>3600	3	1077	48420	0.00%
Average		34.4	11.4	927.1	62.65%		33.8	13.1	886.2	65.28%		10.8	113.4	5199.6	49.80%
Overall average		46.9	23.7	973.7	63.07%		78	474.45	1479.65	64.74%		35.75	76.25	3105.3	56.29%

Table 4.9: CPT with various CGLP solving comparison- Mixed Integer Instances

4.6 Conclusion

In this chapter, we present two new variable splitting rules under the CPT framework. We compare their performance with splitting rules shown previously in [CKS12] and elementary disjunctions using 53 MILP-B instances and 20 MILP-G instances from MIPLIB. Although computational studies show elementary disjunctions performs the best on overall average gap closure, there are instances with significant gap closured improved as the tree being explored aggressively. We then introduce a series of algorithms for solving multi-term CGLPs. Computational experiments with these algorithms integrated with CPT algorithms show aggregated approximation performs the best. It is better than using two-term disjunctive cuts to solve instances. Although overall average gap closure does not improve much, a bundle of instances' gap closure increases dramatically.

One issue that is worth to investigate in the future is the algorithm does not perform well when the instance has large number of variables and constraints. The methods proposed in this chapter for solving multi-term CGLPs are good for large disjunction size but relative small problems. For relatively large sized instances, different variables splitting rules or approximation methods may be useful.

Chapter 5: Contributions and Future Work

Over the last two decades, there has been significant progress with algorithms for solving SMIP problems. Nevertheless, most of the algorithms address sub-classes of these problems which either focus on binary mixed-integer recourse decisions, or pure integer recourse decisions or assumed fixed tenders. In this dissertation, we presented a decomposition algorithm to solve two-stage stochastic mixed-integer program. The algorithm allows general mixed-integer variables in both stages. The algorithm uses multi-term disjunctive cuts to obtain value function approximation of the second-stage mixed-integer programs. This dissertation has made algorithmic and computational contributions to the field of general stochastic mixed-integer programming as follows

- **M-D^2** is the first time-staged decomposition method that is proved to be finitely convergent. Convergence results for most other methods are restricted to certain special cases of the model we consider. Moreover, our method combines both B&B as well as cutting planes, and in this sense, can be looked upon as a decomposition-based branch-and-cut algorithm. Finally, the method allows partial solutions of subproblems, which can be solved to optimality towards the end of the method. Given that the subproblems are mixed-integer programs, this feature avoids having to solve each set of subproblems to optimality in every iteration.

- We proposed two finite algorithms, each of which can construct a polyhedron which can be used to approximate the value function of a mixed-integer program.As with the decomposition method, our approximations are also constructed using multi-term disjunctive cuts.We have also provided computational results which demonstrate both strengths as well as some weaknesses of these approximations.

- In conjunction with our work on decomposition for SMIP, we have also investigated several aspects of the CPT algorithm. In particular, we have compared the performance of alternative ways to explore the cutting plane tree. We have also designed alternative ways to solve problems arising in the generation of multi-term disjunctive cuts, and computational results with these methods have also been presented.

As for future research, there are several avenues that might pursue.

- Parallel Computing. The structure of SMIP problems is ideally suited for decomposition, but the full power of such methods can only be realized if we can harness the power of parallel computing to solving multiple MIP subproblems in parallel. With the proliferation of parallel computing technology, this feature should be added to the algorithms proposed in this dissertation.

- Multi-stage SMIP. The **M-**D^2 method is rather general, and can be used to decompose multi-stage SMIPs into a sequence of two-stage SMIPs in the same manner that multi-stage SLPs can be decomposed into a sequence of two-stage SLPs. This is possible because of the very general structure that we allow for the two-stage model.

- The generality of the proposed scheme can also be exploited by adopting the use of specialized cuts within our decomposition framework. For instance, it should be relatively easy to integrate Gomory cuts [GKS12] into our algorithm, thus providing

a platform for the first SMIP solver which can incorporate much of MIP technology, including B&B, disjunctive cuts, and Gomory cuts.

Appendix A: Example 1.0

Consider the following example that is shown in [SHN03] which is a variation of an example from ([SSvdV98]).

$$\min \quad -1.5x_1 - 4x_2 + \sum_{\omega \in \Omega} p(\omega) f(x, \omega) \tag{A.1a}$$

$$\text{s.t.} \quad x_1, x_2 \text{ binary} \tag{A.1b}$$

where

$$f(x, \omega) = \min \quad -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \tag{A.2a}$$

$$\text{s.t.} \quad \begin{bmatrix} 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \\ 6y_1 + 1y_2 + 3y_3 + 2y_4 - R \end{bmatrix} \leq r(\omega) - T(\omega)x \tag{A.2b}$$

$$y_i \text{ binary } i = 1, ..., 4, R \geq 0 \tag{A.2c}$$

$\Omega = \{\omega_1, \omega_2\}$, $p(\omega_1) = p(\omega_2) = 0.5$, $r(\omega_1) = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$, $T(\omega_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $r(\omega_2) = \begin{bmatrix} 10 \\ 3 \end{bmatrix}$, $T(\omega_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. We first apply the **M-D²** algorithm with **CPT-D** to solve this example.

At iteration $k = 1$, the algorithm starts from solving the LP relaxed master problem with η bounded.

$$\min \quad -1.5x_1 - 4x_2 + \eta \tag{A.3a}$$

$$\text{s.t.} \quad x_1, x_2 \text{ binary} \tag{A.3b}$$

$$\eta \geq -M \tag{A.3c}$$

We get solution $(x_1, x_2, \eta) = (1, 1, -M)$ with objective $v = -M - 5.5$. Here only the root node is in the $B\&B$ tree. $x = (1, 1)$ with $Q_1^o = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ is inserted into subproblems. Then **CPT-D** is called for each $\omega \in \Omega$.

For ω_1, $f_L(x, \omega_1)$ is solved and we get $y(\omega_1) = (0, 1, 0, 0.5, 0)$. y_4 is fractional and partitions are formed for integer variables: $\{y_4 \leq 0\} \cap Q_2$ or $\{y_4 \geq 1\} \cap Q_2$. The cut derived from CGLP for $x \in Q_1^o$ is

$$-2y_2 - 2y_4 + 2R \geq -4 + 2x_2. \tag{A.4}$$

After adding the cut, $f_L(x, \omega_1)$ is re-optimized and the solution is $y(\omega_1) = (0, 0, 0, 1, 0)$. It satisfies integer constraints. So no more cuts are generated in this iteration.

For ω_2, $f_L(x, \omega_2)$ is solved and we get $y(\omega_2) = (0, 1, 0, 0, 0)$. Again, this solution satisfies integer constraints, and hence, no cuts are needed. All scenarios have integer solution, so V is updated. $V = -29$.

The value function cut for $x \in Q_1^o$ is

$$-16.5x_2 + \eta \geq -40. \tag{A.5}$$

At iteration $k = 2$, the master problem continues to be solved by B&B method. We get solution $(x_1, x_2, \eta) = (1, 0, -40)$ with objective $v = -41.5$. Still only the root node is in the B&B tree. $x = (1, 0)$ with $Q_1^o = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ is inserted into subproblems. **CPT-D** is called for each $\omega \in \Omega$.

For ω_1, $f_L(x,\omega)$ is initialized as follows:

$$f_L(x,\omega_1) = \min \quad -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \tag{A.6a}$$

$$\text{s.t.} \quad 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \leq 5 - x_1 \tag{A.6b}$$

$$6y_1 + 1y_2 + 3y_3 + 2y_4 - R \leq 2 - x_2 \tag{A.6c}$$

$$-2y_2 - 2y_4 + 2R \geq -4 + 2x_2 \tag{A.6d}$$

$$0 \leq y_i \leq 1 \; i = 1, ..., 4, R \geq 0 \tag{A.6e}$$

where constraint (A.6d) is from cut (A.4) generated in iteration 1. (A.6) is solved and we get $y(\omega_1) = (0, 1, 0, 1, 0)$. It satisfies integer constraints. So no cuts are needed.

For ω_2, $f_L(x,\omega_2)$ is solved and we get $y(\omega_2) = (0.1154, 1, 0, 0.1538, 0)$. y_1 is the variable we choose to split. The partitions we form are: $\{y_1 \leq 0\} \cap Q_2$ or $\{y_1 \geq 1\} \cap Q_2$. The cut derived from CGLP for $x \in Q_1^o$ is

$$-4.875y_2 - 6.5y_4 + 1.625R \geq -6.5 + 1.625x_1. \tag{A.7}$$

After adding the cut, $f_L(x,\omega_2)$ is re-optimized and the solution is

$$y(\omega_2) = (0.056, 1, 0.222, 0, 0). \tag{A.8}$$

Since $y(\omega_2)$ is located on the root node, no more splits are needed. The same partition is used: $\{y_1 \leq 0\} \cap Q_2$ or $\{y_1 \geq 1\} \cap Q_2$. The cut derived from CGLP for $x \in Q_1^o$ is

$$-2.25y_2 - 4.5y_3 + 2.25R \geq -2.25. \tag{A.9}$$

After adding the cut, $f_L(x,\omega_2)$ is re-optimized and the new solution is $y(\omega_2) = (0.06, 0.68, 0.16, 0.24, 0)$. Again, $y(\omega_2)$ is located on the root node, and no more splits are needed. The same partition: $\{y_1 \leq 0\} \cap Q_2$ or $\{y_1 \geq 1\} \cap Q_2$ is used to formulate CGLP. With only $y(\omega_2)$ changed, another

cut derived from CGLP for $x \in Q_1^o$ is

$$-2.5y_3 - 2.5y_4 + 2.5R \geq -2.5 + 2.5x_1. \tag{A.10}$$

After adding the cut, $f_L(x, \omega_2)$ is re-optimized and the solution is $y(\omega_2) = (0.1667, 1, 0, 0, 0)$. $y(\omega_2)$ is located still on the root node. No more splits are needed. The same partition: $\{y_1 \leq 0\} \cap Q_2$ or $\{y_1 \geq 1\} \cap Q_2$ is used to formulate CGLP. The cut derived from CGLP for $x \in Q_1^o$ is

$$-6y_1 + 1.5R \geq 0. \tag{A.11}$$

After adding the cut, $f_L(x, \omega_2)$ is re-optimized and the solution is $y(\omega_2) = (0, 1, 0, 0, 0)$. It satisfies integer constraints. So no more cuts are needed. All scenarios have integer solution, so V is updated. $V = -34.5$

The value function cut for $x \in Q_1^o$ is

$$-7.55x_1 - 3.8333x_2 + \eta \geq -40.55. \tag{A.12}$$

At iteration $k = 3$, the master problem continues to be solved by the B&B method. We get solution $(x_1, x_2, \eta) = (0, 0, -40)$ with objective $v = -40$. The solution is on node 1 with $Q_1^1 = \{0 \leq x_1 \leq 0, 0 \leq x_2 \leq 1\}$. Hence $x = (0, 0)$ and Q_1^1 are treated as input parameters for **CPT-D** for each $\omega \in \Omega$.

For ω_1, 0 cuts are needed. The solution is $y(\omega_1) = (0, 1, 0, 1, 0)$.

For ω_2, 1 cut is needed (shown below). The solution is $y(\omega_1) = (0, 0, 0, 1, 0)$.

$$-3.6923y_2 - 2.4615y_3 - 3.6923y_4 + 1.2308R \geq -3.6923. \tag{A.13}$$

V is updated. $V = -37.5$ and the value function cut for Q_1^1 is

$$-8.3333x_2 + \eta \geq -37.5 \tag{A.14}$$

At iteration $k = 4$, with updated value function cut for node 1, the master problem continues to be solve by B&B method. We get solution $(x_1, x_2, \eta) = (0, 0, -37.5)$ with objective $v = -37.5$. $V - v \leq \epsilon$. The algorithm stops

A short summary of using **M-D²** algorithm with **CPT-D** to solve this problem is shown in Table A.1.

Iter	v	V	x	Node No.	$f(x, \omega_1)$	Cuts No.	$f(x, \omega_2)$	Cuts No.	Value function cut for Node
1	-M-5.5	inf	(1,1)	1	-28	1	-19	0	$\eta \geq -40 + 16.5x_2$
2	-41.5	-29	(1,0)	1	-47	0	-19	4	$\eta \geq -40.55 + 7.55x_1 + 3.8333x_2$
3	-40	-34.5	(0,0)	2	-47	0	-28	1	$\eta \geq -37.5 + 8.3333x_2$
4	-37.5	-37.5	(0,0)	2					

Table A.1: **M-D²** with **CPT-D** for Example 1

Each row shows the information for one iteration. Column "Node No." denotes the number of active nodes in the B&B tree. "Cuts No" means the number of multi-term disjunctive cuts generated for that scenario.

We also apply the same **M-D²** algorithm but with **BB-D** to solve this example. The algorithm starts from solving the LP relaxed master problem with η bounded.

At iteration $k = 1$, We get solution $(x_1, x_2, \eta) = (1, 1, -M)$ with objective $v = -M - 5.5$ from the B&B method. $x = (1, 1)$ with $Q_1^o = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ is inserted into **BB-D** for each $\omega \in \Omega$.

For ω_1, $f(x, \omega_1)$ is solved by the B&B method and we get 2 nodes in $\mathcal{T}_2$. With 1 cut derived from CGLP for $x \in Q_1^o$:

$$-2y_2 - 2y_4 + 2R \geq -4 + 2x_2, \tag{A.15}$$

the solution falls into $\mathcal{T}_2$ and no more cuts are needed. $y(\omega_1) = (0, 0, 0, 1, 0)$.

For ω_2, $f(x, \omega_2)$ is solved by B&B method and we get 1 nodes in $\mathcal{T}_2$. No cuts are needed. $y(\omega_2) = (0, 1, 0, 0, 0)$. All scenarios have integer solution, so V is updated. $V = -29$

The value function cut for $x \in Q_1^o$ is

$$-16.5x_2 + \eta \geq -40 \tag{A.16}$$

At iteration $k = 2$, the master problem continues to be solved by B&B method. We get solution $(x_1, x_2, \eta) = (1, 0, -40)$ with objective $v = -41.5$. $x = (1, 0)$ with $Q_1^o = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ is inserted into subproblems. **CPT-D** is called for each $\omega \in \Omega$.

For ω_1, $f(x, \omega_1)$ is initialized as follows:

$$\begin{aligned}
f(x, \omega_1) = \min \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R && \text{(A.17a)} \\
\text{s.t.} \quad & 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \leq 5 - x_1 && \text{(A.17b)} \\
& 6y_1 + 1y_2 + 3y_3 + 2y_4 - R \leq 2 - x_2 && \text{(A.17c)} \\
& -2y_2 - 2y_4 + 2R \geq -4 + 2x_2 && \text{(A.17d)} \\
& y_i \text{ binary } i = 1, ..., 4, R \geq 0 && \text{(A.17e)}
\end{aligned}$$

where constraint (A.17d) is from cut (A.15) generated in iteration 1. (A.17) is solved by B&B method and we get $\mathcal{T}_2$ with 1 node. So no cuts are needed. $y(\omega_1) = (0, 1, 0, 1, 0)$.

For ω_2, $f(x, \omega_2)$ is solved by B&B method and we get $\mathcal{T}_2$ with 4 nodes. 4 cuts are derived from CGLP for $x \in Q_1^o$:

$$\begin{aligned}
-8.6667y_1 + 2.1667R &\geq 0 \\
-4y_3 + 4R &\geq 0 \\
-3.75y_2 - 5y_4 + 1.25R &\geq -5 \\
-x_1 - y_4 + R &\geq -1
\end{aligned} \tag{A.18}$$

With 4 cuts added, the solution $y(\omega_1) = (0, 1, 0, 1, 0)$.

All scenarios have integer solution, so V is updated. $V = -34.5$

The value function cut for $x \in Q_1^o$ is

$$-4.5x_1 - 3.8333x_2 + \eta \geq -37.5. \tag{A.19}$$

At iteration $k = 3$, the master problem continues to be solve by B&B method. We get solution $(x_1, x_2, \eta) = (0, 0, -37.5)$ with objective $v = -37.5$. The solution is on node 2 with $Q_1^1 = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 0\}$. $x = (0, 0)$ and Q_1^2 are treated as input parameters for **BB-D** for each $\omega \in \Omega$.

For ω_1, 0 cuts are needed. The solution is $y(\omega_1) = (0, 1, 0, 1, 0)$.

For ω_2, 0 cuts are needed. The solution is $y(\omega_1) = (0, 0, 0, 1, 0)$. V is updated. $V = -37.5$ and the value function cut for Q_1^2 is

$$-2.8333x_1 - 5.1667x_2 + \eta \geq -37.5 \tag{A.20}$$

At iteration $k = 4$, with updated value function cut for node 2, the master problem continues to be solve by B&B method. We get solution $(x_1, x_2, \eta) = (0, 0, -37.5)$ with objective $v = -37.5$. $V - v \leq \epsilon$. The algorithm stops

A short summary of using **M-D²** algorithm with **BB-D** to solve this problem is shown in Table A.2. As you may notice, there is only small difference between **BB-D** and **CPT-D**

Iter	v	V	x	Node No.	$f(x, \omega_1)$	Cuts No.	$f(x, \omega_2)$	Cuts No.	Value function cut for Node
1	-M-5.5	inf	(1,1)	1	-28	1	-19	0	$\eta \geq -40 + 16.5x_2$
2	-41.5	-29	(1,0)	1	-47	0	-19	4	$\eta \geq -37.5 + 4.5x_1 + 3.8333x_2$
3	-37.5	-34.5	(0,0)	2	-47	0	-28	0	$\eta \geq -37.5 + 2.8333x_1 + 5.1667x_2$
4	-37.5	-37.5	(0,0)	2					

Table A.2: **M-D²** with **BB-D** for Example 1

for **Example 1**. At iteration 2, because **BB-D** uses partitions with 4 terms to generate cuts

from the beginning, the quality of the cut is better than **CPT-D** which results in no more cuts needed for subproblems in the following iterations. But CGLP with 4 terms take more time to solve. So it is not easy to say which method is better for this example.

Appendix B: Wait-and-see experiment result

Table B.1: Wait-and-see experiment objective deviation statistics

		CPT LP				CPT IP				Lift and Project LP				Lift and Project IP			
Instances	p	mean	std	5% prc	95% prc	mean	std	5% prc	95% prc	mean	std	5% prc	95% prc	mean	std	5% prc	95% prc
aflow30a	5	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00
aflow30a	10	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00
aflow30a	20	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00
aflow30a	50	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00
aflow30a	100	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00
bell3a	5	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
bell3a	10	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
bell3a	20	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
bell3a	50	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
bell3a	100	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
blend2	5	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00
blend2	10	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00
blend2	20	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00
blend2	50	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00
blend2	100	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00
dcmulti	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dcmulti	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dcmulti	20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dcmulti	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dcmulti	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
egout	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
egout	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
egout	20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
egout	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
egout	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
fixnet6	5	0.13	0.00	0.13	0.13	0.00	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.00	0.00	0.00	0.00

fixnet6	10	0.13	0.00	0.13	0.13	0.00	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.00	0.00	0.00	0.00
fixnet6	20	0.13	0.00	0.13	0.13	0.00	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.00	0.00	0.00	0.00
fixnet6	50	0.13	0.00	0.13	0.13	0.00	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.00	0.00	0.00	0.00
fixnet6	100	0.13	0.00	0.13	0.13	0.00	0.00	0.00	0.00	0.14	0.00	0.14	0.14	0.00	0.00	0.00	0.00
flugpl	5	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
flugpl	10	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
flugpl	20	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
flugpl	50	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
flugpl	100	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
gen	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gen	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gen	20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gen	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
gen	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
misc03	5	0.34	0.00	0.34	0.34	0.00	0.00	0.00	0.00	0.35	0.00	0.35	0.35	0.00	0.00	0.00	0.00
misc03	10	0.34	0.00	0.34	0.34	0.00	0.00	0.00	0.00	0.35	0.00	0.35	0.35	0.00	0.00	0.00	0.00
misc03x	20	0.34	0.00	0.34	0.34	0.00	0.00	0.00	0.00	0.35	0.00	0.35	0.35	0.00	0.00	0.00	0.00
misc03	50	0.34	0.00	0.34	0.34	0.00	0.00	0.00	0.00	0.35	0.00	0.35	0.35	0.00	0.00	0.00	0.00
misc03	100	0.34	0.00	0.34	0.34	0.00	0.00	0.00	0.00	0.35	0.00	0.35	0.35	0.00	0.00	0.00	0.00
mod008	5	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
mod008	10	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
mod008	20	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
mod008	50	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
mod008	100	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00	0.04	0.00	0.04	0.04	0.00	0.00	0.00	0.00
modglob	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
modglob	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
modglob	20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
modglob	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
modglob	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p0033	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p0033	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p0033	20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

p0033	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p0033	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p0201	5	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
p0201	10	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
p0201	20	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
p0201	50	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
p0201	100	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
p0282	5	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
p0282	10	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
p0282	20	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
p0282	50	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
p0282	100	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00
p0548	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p0548	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p0548	20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p0548	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
p0548	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pp08a	5	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00
pp08a	10	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00
pp08a	20	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00
pp08a	50	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00
pp08a	100	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00	0.03	0.00	0.03	0.03	0.00	0.00	0.00	0.00
pp08aCUTS	5	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
pp08aCUTS	10	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
pp08aCUTS	20	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
pp08aCUTS	50	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
pp08aCUTS	100	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.02	0.02	0.00	0.00	0.00	0.00
rgn	5	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00
rgn	10	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00
rgn	20	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00
rgn	50	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00
rgn	100	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00	0.16	0.00	0.16	0.16	0.00	0.00	0.00	0.00

set1ch	5	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
set1ch	10	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
set1ch	20	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
set1ch	50	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
set1ch	100	0.05	0.00	0.05	0.05	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
stein15	5	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.22	0.00	0.22	0.22	0.00	0.00	0.00	0.00
stein15	10	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.22	0.00	0.22	0.22	0.00	0.00	0.00	0.00
stein15	20	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.22	0.00	0.22	0.22	0.00	0.00	0.00	0.00
stein15	50	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.22	0.00	0.22	0.22	0.00	0.00	0.00	0.00
stein15	100	0.07	0.00	0.07	0.07	0.00	0.00	0.00	0.00	0.22	0.00	0.22	0.22	0.00	0.00	0.00	0.00
stein27	5	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00
stein27	10	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00
stein27	20	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00
stein27	50	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00
stein27	100	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00	0.28	0.00	0.28	0.28	0.00	0.00	0.00	0.00
stein45	5	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00
stein45	10	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00
stein45	20	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00
stein45	50	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00
stein45	100	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00	0.27	0.00	0.27	0.27	0.00	0.00	0.00	0.00
vpm1	5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
vpm1	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
vpm1	20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
vpm1	50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
vpm1	100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
vpm2	5	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
vpm2	10	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
vpm2	20	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
vpm2	50	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
vpm2	100	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00	0.06	0.00	0.06	0.06	0.00	0.00	0.00	0.00
Total		8.61				0.00				9.26				0.01			

Table B.2: Wait-and-see experiment solving time

Instances	p	CPT LP	CPT IP	L&P LP	L&P IP	IP
aflow30a.mps	5	0.12	29.56	0.12	26.97	26.07
aflow30a.mps	10	0.19	29.70	0.09	26.52	26.40
aflow30a.mps	20	0.12	29.36	0.11	26.88	25.99
aflow30a.mps	50	0.08	29.13	0.13	25.93	25.57
aflow30a.mps	100	0.16	29.11	0.03	26.79	25.60
bell3a.mps	5	0.05	14.96	0.06	13.54	15.74
bell3a.mps	10	0.05	14.82	0.06	13.54	15.62
bell3a.mps	20	0.05	14.79	0.05	12.86	15.73
bell3a.mps	50	0.06	14.98	0.05	13.51	15.65
bell3a.mps	100	0.05	14.88	0.03	12.70	15.66
blend2.mps	5	0.06	5.88	0.06	5.35	5.34
blend2.mps	10	0.05	5.93	0.08	5.55	5.57
blend2.mps	20	0.06	5.63	0.06	5.35	5.40
blend2.mps	50	0.06	6.30	0.06	5.59	5.58
blend2.mps	100	0.06	6.12	0.05	5.71	5.79
dcmulti.mps	5	0.08	3.09	0.11	3.39	3.14
dcmulti.mps	10	0.11	3.09	0.09	3.32	3.08
dcmulti.mps	20	0.09	3.02	0.11	3.37	3.06
dcmulti.mps	50	0.11	3.06	0.13	3.39	3.02
dcmulti.mps	100	0.11	3.07	0.11	3.46	2.95
egout.mps	5	0.06	0.30	0.06	0.25	0.30
egout.mps	10	0.08	0.28	0.05	0.28	0.27
egout.mps	20	0.08	0.31	0.08	0.27	0.25
egout.mps	50	0.08	0.30	0.08	0.30	0.27
egout.mps	100	0.09	0.28	0.06	0.27	0.30
fixnet6.mps	5	0.13	4.45	0.13	5.91	5.35
fixnet6.mps	10	0.17	4.49	0.14	6.19	5.51
fixnet6.mps	20	0.11	4.46	0.13	5.85	5.65
fixnet6.mps	50	0.17	4.57	0.13	6.20	5.55
fixnet6.mps	100	0.13	4.57	0.14	5.95	5.43
flugpl.mps	5	0.09	0.73	0.16	0.55	0.48
flugpl.mps	10	0.13	0.73	0.14	0.55	0.58
flugpl.mps	20	0.08	0.76	0.17	0.48	0.47
flugpl.mps	50	0.11	0.83	0.12	0.53	0.44
flugpl.mps	100	0.06	0.78	0.12	0.55	0.47
gen.mps	5	0.16	0.42	0.11	0.47	0.50
gen.mps	10	0.14	0.45	0.14	0.44	0.47
gen.mps	20	0.14	0.58	0.11	0.47	0.42
gen.mps	50	0.17	0.42	0.11	0.40	0.44

gen.mps	100	0.11	0.44	0.14	0.45	0.44
misc03.mps	5	0.13	3.09	0.12	2.50	3.78
misc03.mps	10	0.10	3.01	0.12	2.56	3.96
misc03.mps	20	0.09	2.95	0.11	2.58	3.42
misc03.mps	50	0.13	3.01	0.13	2.57	4.81
misc03.mps	100	0.13	3.28	0.12	2.54	4.18
mod008.mps	5	0.08	3.08	0.11	3.07	2.89
mod008.mps	10	0.11	3.34	0.11	2.71	2.59
mod008.mps	20	0.13	2.73	0.14	2.40	2.26
mod008.mps	50	0.09	2.76	0.15	2.26	2.22
mod008.mps	100	0.12	2.95	0.13	2.84	2.67
modglob.mps	5	0.12	1.52	0.14	1.44	1.55
modglob.mps	10	0.17	1.48	0.12	1.48	1.52
modglob.mps	20	0.16	1.40	0.16	1.48	1.55
modglob.mps	50	0.16	1.43	0.15	1.51	1.57
modglob.mps	100	0.16	1.49	0.16	1.54	1.59
p0033.mps	5	0.11	0.31	0.13	0.31	0.27
p0033.mps	10	0.11	0.06	0.13	0.12	0.31
p0033.mps	20	0.13	0.28	0.14	0.02	0.00
p0033.mps	50	0.13	0.31	0.14	0.28	0.31
p0033.mps	100	0.14	0.25	0.13	0.22	0.19
p0201.mps	5	0.16	2.28	0.16	2.31	2.14
p0201.mps	10	0.14	2.23	0.13	2.28	2.03
p0201.mps	20	0.14	2.37	0.14	2.31	2.17
p0201.mps	50	0.16	2.28	0.17	2.23	2.25
p0201.mps	100	0.15	2.28	0.16	2.04	2.06
p0282.mps	5	0.17	1.67	0.16	1.44	1.56
p0282.mps	10	0.16	1.64	0.19	1.42	1.56
p0282.mps	20	0.14	1.69	0.17	1.43	1.58
p0282.mps	50	0.14	1.67	0.17	1.42	1.58
p0282.mps	100	0.17	1.61	0.16	1.44	1.59
p0548.mps	5	0.14	1.05	0.16	0.94	0.92
p0548.mps	10	0.17	1.09	0.17	0.94	0.91
p0548.mps	20	0.17	1.09	0.17	0.95	0.97
p0548.mps	50	0.17	1.08	0.16	0.95	0.95
p0548.mps	100	0.17	1.08	0.17	0.94	0.95
pp08a.mps	5	0.16	3.25	0.19	3.07	2.81
pp08a.mps	10	0.17	3.25	0.19	3.32	2.72
pp08a.mps	20	0.17	3.17	0.17	2.95	3.12
pp08a.mps	50	0.17	3.20	0.19	3.04	2.73
pp08a.mps	100	0.17	3.12	0.19	3.50	2.70
pp08aCUTS.mps	5	0.17	5.33	0.17	5.55	5.87
pp08aCUTS.mps	10	0.17	5.18	0.19	6.18	4.38

pp08aCUTS.mps	20	0.16	4.70	0.17	4.66	4.24
pp08aCUTS.mps	50	0.17	4.96	0.19	5.12	5.18
pp08aCUTS.mps	100	0.16	5.24	0.19	4.20	4.88
rgn.mps	5	0.16	3.76	0.16	2.40	2.33
rgn.mps	10	0.19	3.59	0.16	2.36	2.43
rgn.mps	20	0.19	3.73	0.17	2.40	2.42
rgn.mps	50	0.19	3.81	0.17	2.29	2.34
rgn.mps	100	0.19	3.57	0.14	2.28	2.28
set1ch.mps	5	0.20	9.05	0.31	8.60	8.53
set1ch.mps	10	0.19	9.08	0.31	8.58	8.64
set1ch.mps	20	0.27	8.88	0.17	8.61	8.78
set1ch.mps	50	0.19	9.27	0.31	8.35	8.30
set1ch.mps	100	0.19	9.22	0.27	8.75	8.52
stein15.mps	5	0.17	0.38	0.22	0.65	0.48
stein15.mps	10	0.17	0.35	0.22	0.70	0.48
stein15.mps	20	0.19	0.39	0.20	0.67	0.50
stein15.mps	50	0.20	0.39	0.19	0.69	0.50
stein15.mps	100	0.14	0.34	0.19	0.68	0.51
stein27.mps	5	0.17	5.62	0.25	3.18	1.97
stein27.mps	10	0.16	5.63	0.22	3.18	1.86
stein27.mps	20	0.20	5.55	0.23	3.21	1.87
stein27.mps	50	0.16	5.55	0.23	3.21	1.95
stein27.mps	100	0.19	5.63	0.22	3.21	2.00
stein45.mps	5	0.17	45.21	0.19	35.68	36.25
stein45.mps	10	0.20	44.60	0.20	36.30	37.13
stein45.mps	20	0.16	44.76	0.19	36.05	36.07
stein45.mps	50	0.22	44.42	0.27	34.95	37.16
stein45.mps	100	0.19	44.80	0.24	36.38	36.58
vpm1.mps	5	0.16	0.41	0.20	0.41	0.33
vpm1.mps	10	0.19	0.36	0.14	0.37	0.37
vpm1.mps	20	0.16	0.37	0.17	0.41	0.36
vpm1.mps	50	0.20	0.37	0.20	0.41	0.37
vpm1.mps	100	0.19	0.37	0.19	0.40	0.38
vpm2.mps	5	0.22	4.05	0.23	3.25	3.24
vpm2.mps	10	0.17	3.81	0.19	3.09	3.15
vpm2.mps	20	0.15	3.82	0.19	3.29	3.31
vpm2.mps	50	0.20	3.99	0.22	3.26	3.31
vpm2.mps	100	0.20	3.93	0.22	3.15	3.12
Total		17.78	751.61	19.10	651.41	663.22

Appendix C: The vertices on the disjunctive cut

The dual solution of CGLP contains information to calculate vertices x^{kt} that supports the CPT cut. x^{kt} is simplified as x_t in this section.

Proposition 6. *Suppose the solution of m1nc formulation 4.5 is* $(\pi, \lambda_t, \mu_t, \nu_t)$ *for* $t \in \{1...T\}$ *and its dual solution* (y_t, z_t)*. For boxes with* $z_t \neq 0$*, there exist vertices* $x_t = y_t/z_t$ *that satisfies* $\pi^\top x_t = \pi^\top x^k + 1$*.*

Proof. For box t, since y_t is the dual multiplier for $\pi = A^\top \lambda_t + \mu_t - \nu_t$, and z_t is dual multiplier for $b^\top \lambda_t + L_t^\top \mu_t - U_t^\top \nu_t \geq \pi^\top x^k + 1$ based on complementary slackness condition at optimal, we have

$$\pi^\top y_t = \lambda_t^\top A y_t + \mu_t^\top y_t + \nu_t^\top y_t \tag{C.1}$$

and

$$z_t(b^\top \lambda_t + L_t^\top \mu_t - U_t^\top \nu_t) = z_t(\pi^\top x^k + 1) \tag{C.2}$$

Eq C.2 can also be written as

$$\lambda_t^\top (b z_t) + \mu_t^\top (L_t z_t) - \nu_t \top (U_t z_t) = z_t(\pi^\top x^k + 1) \tag{C.3}$$

From the dual of m1nc formulation 4.6, since λ_t is the dual multiplier for $A y_t \geq \bar{b} z_t$, μ_t corresponds to $y_t \geq \bar{L}_t z_t$ and μ_t corresponds to $y_t \leq \bar{U}_t z_t$, also from complementary slackness,

$$\lambda_t^\top A y_t = \lambda_t^\top b z_t \tag{C.4}$$

$$\mu_t^\top y_t = \mu_t^\top L_t z_t \tag{C.5}$$

$$\nu_t^\top y_t = \nu_t^\top U_t z_t \tag{C.6}$$

Thus

$$\pi^\top y_t = \lambda_t^\top A y_t + \mu_t^\top y_t + \nu_t^\top y_t = \lambda_t^\top (b z_t) + \mu_t^\top (L_t z_t) - \nu_t \top (U_t z_t) = z_t(\pi^\top x^k + 1) \tag{C.7}$$

For $z_t \neq 0$, we have $\pi^\top x_t = \pi^\top x^k + 1$. □

These vertices are from leaves nodes, they provides us the tool to continue grow CPT tree when x^k is not on the leaves node.

Appendix D: Instances from MIPLIB 3.0 and MIPLIB 2003

Table D.1: Mixed-Binary Instances

Instance	ROWS	COLS	NZ	INT	BIN	CON
aflow30a	479	842	2091	0	421	421
danoint	664	521	3232	0	56	465
dcmulti	290	548	1315	0	75	473
egout	98	141	282	0	55	86
enigma	21	100	289	0	100	0
fixnet6	478	878	1756	0	378	500
glass4	396	322	1815	0	302	20
lseu	28	89	309	0	89	0
markshare1	6	62	312	0	50	12
markshare2	7	74	434	0	60	14
mas74	13	151	1706	0	150	1
mas76	12	151	1640	0	150	1
misc03	96	160	2053	0	159	1
misc07	212	260	8619	0	259	1
mod008	6	319	1243	0	319	0
modglob	291	422	968	0	98	324
opt1217	64	769	1542	0	768	1
p0033	16	33	98	0	33	0
p0201	133	201	1923	0	201	0
p0282	241	282	1966	0	282	0
p0548	176	548	1711	0	548	0
pk1	45	86	915	0	55	31
pp08a	136	240	480	0	64	176
pp08aCUTS	246	240	839	0	64	176
qiu	1192	840	3432	0	48	792
rgn	24	180	460	0	100	80
set1ch	492	712	1412	0	240	472
stein15	36	15	120	0	15	0

stein27	118	27	378	0	27	0
stein45	331	45	1034	0	45	0
vpm1	234	378	749	0	168	210
vpm2	234	378	917	0	168	210
10teams	230	2025	12150	0	1800	225
air03	124	10757	91028	0	10757	0
air04	823	8904	72965	0	8904	0
air05	426	7195	52121	0	7195	0
cap6000	2176	6000	48243	0	6000	0
dano3mip	3202	13873	79655	0	552	13321
fast0507	507	63009	4E+05	0	63009	0
fiber	363	1298	2944	0	1254	44
harp2	112	2993	5840	0	2993	0
khb05250	101	1350	2700	0	24	1326
l152lav	97	1989	9922	0	1989	0
misc06	820	1808	5859	0	112	1696
mitre	2054	10724	39704	0	10724	0
mkc	3411	5325	17038	0	5323	2
mod010	146	2655	11203	0	2655	0
mod011	4480	10975	22271	0	96	10879
nw04	36	87482	6E+05	0	87482	0
p2756	755	2756	8937	0	2756	0
rentacar	6803	9559	41844	0	55	9504
seymour	4944	1372	33549	0	1372	0
swath	884	6805	34965	0	6724	81

Instance	ROWS	COLS	NZ	INT	BIN	CON
bell3a	123	133	347	32	39	62
bell5	91	104	266	28	30	46
blend2	274	353	1409	33	231	89
flugpl	18	18	46	11	0	7
gen	780	870	2592	6	144	720
gt2	29	188	376	164	24	0
noswot	182	128	735	25	75	28
rout	291	556	2431	15	300	241
timtab1	171	397	829	107	64	226
timtab2	294	675	1482	181	113	381
arki001	1048	1388	20439	123	415	850
gesa2	1392	1224	5064	168	240	816
gesa2_o	1248	1224	3672	336	384	504
gesa3	1368	1152	4944	168	216	768
gesa3_o	1224	1152	3624	336	336	480
qnet1	503	1541	4622	129	1288	124
qnet1_o	456	1541	4214	129	1288	124
n4-3	1236	3596	14036	174	0	3422
mik.250-1-100.1	151	251	5351	150	100	1
dfn-gwin-UUM	158	938	2632	90	0	848

Table D.2: Mixed-Integer Instances

Bibliography

[AS05] Warren Adams and Hanif Sherali. A hierarchy of relaxations leading to the convex hull representation for general discrete optimization problems. *Annals of Operations Research*, 140:21–47, 2005. 10.1007/s10479-005-3966-4.

[ATS04] Shabbir Ahmed, Mohit Tawarmalani, and Nikolaos V. Sahinidis. A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100(2):355–377, 2004.

[Bal79] Egon Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51, 1979.

[BB09] E. Balas and P. Bonami. Generating lift-and-project cuts from the LP simplex tableau: open source implementation and testing of new variants. *mathematical programming computation*, 1:165–199, 2009.

[BCC93] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.

[Ben62] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, September 1962.

[BES+96] R. Bixby, Ceria E., McZeal S., C. M., and M. W. P. Savelsbergh. An updated mixed integer linear programming library: Miplib 3.0. 1996.

[BL97] J.R. Birge and F Louveaux. *Introduction to Stochastic Programming*. Springer Verlag, 1997.

[BP02] Egon Balas and Michael Perregaard. Lift-and-project for mixed 0-1 programming: recent progress. *Discrete Applied Mathematics*, 123(1-3):129 – 154, 2002.

[Cad10] Florent Cadoux. Computing deep facet-defining disjunctive cuts for mixed-integer programming. *Mathematical Programming*, 122(2):197 – 223, 2010.

[CKS90] W. Cook, R. Kannan, and A. Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47:155–174, 1990. 10.1007/BF01580858.

[CKS11] Binyuan Chen, Simge Küçükyavuz, and Suvrajeet Sen. Finite disjunctive programming characterizations for general mixed-integer linear programs. *Operations Research*, 59(1):202–210, 2011.

[CKS12] Binyuan Chen, Simge Küçükyavuz, and Suvrajeet Sen. A computational study of the cutting plane tree algorithm for general mixed-integer linear programs. *Operations Resesarch Letters*, 40(1):15–19, 2012.

[CS98] Claus C. Carøe and Rüdiger Schultz. A two-stage stochastic program for unit commitment under uncertainty in a hydro-thermal power system. In *Preprint SC 98-11, Konrad-Zuse-Zentrum fur Informationstechnik*, pages 98–13, 1998.

[CT98] Claus Carøe and Jørgen Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83:451–464, 1998. 10.1007/BF02680570.

[Dan55] G. Dantzig. Upper bounds, secondary constraints and block triangularity in linear programming. *Econometrica*, 23:174–183, 1955.

[EGMP07] L. Escudero, A. Garín, M. Merino, and G. Pérez. A two-stage stochastic integer programming approach as a mixture of branch-and-fix coordination and benders decomposition schemes. *Annals of Operations Research*, 152:395–420, 2007. 10.1007/s10479-006-0138-0.

[GKS12] Dinakar Gade, Simge Küçükyavuz, and Suvrajeet Sen. Decomposition algorithms with gomory cuts for two-stage stochastic integer programs. page under review, 2012.

[Gom63] R. E. Gomory. An algorithm for integer solutions to linear programs. *In Graves,P. and P. Wolfe (eds.) Recent Advances in Mathematical Programming*, pages 269–302, 1963.

[Gre98] Harvey Greenberg. An annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization, 1998.

[KSH06] Nan Kong, Andrew J. Schaefer, and Brady Hunsaker. Two-stage integer programs with stochastic right-hand sides. *Mathematical Programming*, 108(24):275–296, 2006.

[LD60] A. H. Land and A. G Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[LL93] Gilbert Laporte and François V. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133 – 142, 1993.

[LS91] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM JOURNAL ON OPTIMIZATION*, 1:166–190, 1991.

[LS04] Guglielmo Lulli and Suvrajeet Sen. *Management Science*, 50(6), 2004.

[LvdV93] François V. Louveaux and Maarten H. van der Vlerk. Stochastic programming with simple integer recourse. *Mathematical Programming*, 61(3):301–325, 1993.

[NS05] Lewis Ntaimo and Suvrajeet Sen. The million-variable m̈archf̈or stochastic combinatorial optimization. *the Journal of Global Optimization*, 2005.

[NS07] Lewis Ntaimo and Suvrajeet Sen. A branch and cut algorithm for two stage stochastic mixed binary programs with continuous first stage variables. *Int. J. Comput. Sci. Eng.*, 3(3):232–241, 2007.

[OM01] Jonathan H. Owen and Sanjay Mehrotra. A disjunctive cutting plane procedure for general mixed-integer linear programs. *Mathematical Programming*, 89(3):437–448, 2001.

[PB01a] Michael Perregaard and Egon Balas. Generating cuts from multiple-term disjunctions. 2081:348–360, 2001.

[PB01b] Michael Perregaard and Egon Balas. Generating cuts from multiple-term disjunctions. In *Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pages 348–360, London, UK, 2001. Springer-Verlag.

[SA90] H D Sherali and W P Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990.

[Sen03] Suvrajeet Sen. *Algorithms for stochastic mixed-integer programming models*, volume 12. Elsevier, 2003. Handbooks in Operations Research and Management Science.

[SF02] Hanif D. Sherali and Barbara M.P. Fraticelli. A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22:319–342, 2002. 10.1023/A:1013827731218.

[SH05] Suvrajeet Sen and Julia L. Higle. The c3 theorem and a d2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104(1):1–20, 2005.

[She94] Hanif D Sherali. A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, 52(1):83–106, 1994.

[SHN03] S. Sen, J. Higle, and L. Ntaimo. A summary and illustration of disjunctive decomposition with set convexification. *Woodruff, D. (ed.) Network Interdiction and Stochastic Integer Programming*, pages 105–125, 2003.

[SS06] Suvrajeet Sen and Hanif D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223, 2006.

[SSvdV98] R. Schultz, L. Stougie, and van der Vlerk. Solving stochastic programs with integer recourse by enumeration: A framework using gröbner basis. *Mathematical Programming*, 83(1):229–252, 1998.

[Str74] Beata Strazicky. On an algorithm for solution of the two-stage stochastic programming problem. *Methods of operations research*, 19:142–156, 1974.

[SZ06] H. D. Sherali and Xiaomei Zhu. On solving discrete two-stage stochastic programs having mixed-integer first- and second-stage variables. *Mathematical Programming*, 108(2-3):597–616, 2006.

[TPS12] Andrew C. Trapp, Oleg A. Prokopyev, and Andrew J. Schaefer. On a level-set characterization of the integer programming value function and its application to stochastic programming. *Operations Research*, 2012.

[Win74] C. Winkler. Basis factorization for block-angular linear programs: unified theory of partition and decomposition using the simplex method. Technical report, Systems Optimization Laboratory, Stanford, 1974.

[Wol80] Richard D. Wollmer. Two-stage linear programming under uncertainty with $0-1$ integer first stage variables. *Mathematical Programming*, 19(3):279–288, 1980.

[YS09] Yang Yuan and Suvrajeet Sen. Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS J. on Computing*, 21(3):480–487, July 2009.