FINITE DISJUNCTIVE PROGRAMMING METHODS FOR GENERAL MIXED INTEGER LINEAR PROGRAMS

by

Binyuan Chen

A Dissertation Submitted to the Faculty of the

SYSTEMS AND INDUSTRIAL ENGINEERING DEPARTMENT

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2011

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Binyuan Chen
entitled Finite Disjunctive Programming Methods for General Mixed Integer Linear Programs
and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Suvrajeet Sen — Date: 25 March 2011

Güzin Bayraksan — Date: 25 March 2011

Simge Küçükyavuz — Date: 25 March 2011

Ferenc Szidarovszky — Date: 25 March 2011

Daniel Zeng — Date: 25 March 2011

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.
I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Dissertation Director: Suvrajeet Sen — Date: 25 March 2011

Dissertation Director: Güzin Bayraksan — Date: 25 March 2011

STATEMENT BY AUTHOR

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

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Binyuan Chen

ACKNOWLEDGEMENTS

It has been a long journey since I started to write down the first word of this dissertation. Along this journey, my dissertation has been a priority, and most of the time, it is the number one priority. I feel elated when I finished my dissertation. However, I could not have succeeded and come to where I am today without the valuable support from several.

First, I would like to give sincere gratitude to my advisor, Dr. Suvrajeet Sen. His patience, caring and flexibility have helped push me through the last chapter of my dissertation. "He is a treasure land" , as one of the fellow that we both worked with commented. His profound knowledge in so many areas surprises me all the time. He is not only a great mentor but a close friend to his students. His zealot for uncovering unknown and energetic pursuit for truth set a good example for me and are always the impetus for me to finish.

I would also like to give a heartfelt thanks to Dr. Simge Küçükyavuz. We have worked together for quite a while on some very interesting aspects of mixed-integer programming, and I learned a lot from her. I am grateful for her contributions to our joint papers and advice on my dissertation.

Sincere thanks also goes to Dr. Güzin Bayraksan, who had introduced various aspects of linear programming, statistics, stochastic processes and stochastic programming and provided me with help and support for years. I also want to thank her for her helpful suggestions on shaping this dissertation.

I am also very grateful to Dr. Ferenc Sizdarovszky and Dr. Daniel Zeng for serving on my dissertation committee and for their help over the years. My thanks also goes to Dr. Pitu Mirchandani, Dr. Wei H. Lin and Dr. Yong Jun Son for their mentoring in

classes. My gratitude is also extended to Dr. Larry Head, chair of the Industrial and Systems Engineering department of the University of Arizona, and Dr. Julia L. Higle, chair of the Integrated Systems Engineering department of the Ohio State University, for their constant supports, while I was working as a teaching/research assistant and as a visiting scientist in the University of Arizona and in the Ohio State University. I'd also like to thank Linda, Myra, Ashely and Karen in the Industrial and Systems Engineering department of the University of Arizona, and Darline, Cedric and Mike in the Integrated Systems Engineering department of the Ohio State University for their help and services in all aspects.

Supports and help also come from fiends of mine, with whom I have shared minds and happiness. They are Jinquan Li, Hong Huo, Feng Huang, Zhiping Wei, Qin He, Yejuan Long, Yang Yuan, Li Li, Yunwei Qi, Lili Zhuang, Zhihong Zhou, Shugang Kang, Dinakar Gade, Praneeth AVS and many others.

The acknowledgments would not be completed without giving thanks to my parents, Lianggong Chen and Shuqing Chen. They've taught me about self-respect and set a role model of persistence and independence. They love me and are proud of me, I love them and am proud of them too.

Finally, I want to dedicate this dissertation to my dearest wife, Bing Qiu, whom I always mean the best of everything for.

DEDICATION

to *Bing Qiu*

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In this dissertation, a finitely convergent disjunctive programming procedure, the Convex Hull Tree (CHT) algorithm, is proposed to obtain the convex hull of a general mixed-integer linear program with bounded integer variables. The CHT algorithm constructs a linear program that has the same optimal solution as the associated mixed-integer linear program. The standard notion of sequential cutting planes is then combined with ideas underlying the CHT algorithm to help guide the choice of disjunctions to use within a new cutting plane method, the Cutting Plane Tree (CPT) algorithm. We show that the CPT algorithm converges to an integer optimal solution of the general mixed-integer linear program with bounded integer variables in finitely many steps. We also enhance the CPT algorithm with several techniques including a "round-of-cuts" approach and an iterative method for solving the cut generation linear program (CGLP). Two normalization constraints are discussed in detail for solving the CGLP. For moderately sized instances, our study shows that the CPT algorithm provides significant gap closures with a pure cutting plane method.

Key words: Mixed-integer linear program, disjunctive programming, convex hull, cutting plane, finite convergence.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Mixed-integer linear programming has come a long way. Advanced software (e.g., XPRESS, CPLEX, etc.) are routinely solving a mixed-integer linear program (MILP) with thousands of variables with very reasonable computational efforts. The area is now growing towards mixed-integer nonlinear programming, as well as stochastic MILPs. In both cases, impressive computational results have been reported by Bonami et al. (2008) and Yuan and Sen (2009), respectively.

One of the more important lessons learned from mixed-integer linear programming and related literature is that valid inequalities are indispensable in robust software. Thus, it is common for commercial software to include both general purpose valid inequalities, such as Gomory cuts and mixed-integer rounding cuts (see Cornuéjols (2008) for a recent survey), and special purpose valid inequalities, such as flow cover inequalities (Padberg et al., 1985) and flow path inequalities (Van Roy and Wolsey, 1985), within branch-and-cut methods. As a result, the study of valid inequalities remains an important part of the mixed-integer linear programming literature, even though the performance of *pure* cutting plane methods leaves a lot to be desired.

The polyhedral study of a binary MILP (MILP-B) has gained significant achievements during the past several decades. Various algorithmic approaches are available to generate cutting planes that define the convex hull of feasible points of a MILP-B. However, for a MILP with general integer variables (MILP-G), this issue has not been as well understood,

even when the integer variables are bounded. Although disjunctive cuts are natural to use for MILP-Bs, there is no cutting plane procedure using disjunctive cuts that has been proved to converge in finitely many steps.

In this dissertation, we study a new finitely convergent disjunctive cutting plane procedure, referred to as the Cutting Plane Tree (CPT) algorithm, to solve MILP-Gs. In this algorithm, we create finer and finer partitions of the feasible region that result in multiple disjunctions, as guided by the objective function.

1.2 Problem Statement

In this dissertation, assuming a nonempty feasible set and bounded general integer variables of MILP-Gs, we address the following questions.

- Is it possible to use the disjunctive programming methodology to describe the convex hull of MILP-G solutions in finitely many steps without introducing binary variables?
- Is there a constructive methodology to obtain an optimal solution to a MILP-G using a disjunctive programming characterization of its convex hull?
- If we are restricted to introduce only one cutting plane in any iteration, is there a finitely convergent disjunctive programming algorithm that solves a MILP-G?
- If the answers to the above questions are affirmative, does the algorithmic power of the found algorithm translate into superior computational performance?

The assumption "bounded general integer variables" above is sometimes replaced by an alternative statement "a bounded optimal objective value", when an objective function of the form "$\min c^T x$ or $\max c^T x$" is involved. As the convexification process is directed

by the objective function, as long as the optimal objective value is bounded, for those integer variables that start with infinite bound(s) and define the objective function in the optimal Simplex tableau as non-basic variables, the partitioning (or splitting) of these variables derives finite upper/lower bounds for them.

1.3 Organization of the Dissertation

This dissertation consists of seven chapters, all chapters from Chapter 2 are organized as below.

Chapter 2 reviews the literature of mixed integer linear programming, major algorithmic and computational advances and challenges of mixed-integer linear programming are discussed. Chapter 3 introduces the CHT algorithm as a convexification procedure for MILP-Gs and the CPT algorithm which combines sequential convexification ideas with the CHT for solving MILP-Gs. Some important examples in the literature are re-visited to demonstrate the effectiveness of the CPT algorithm. Computational study is performed on an implementation of the CPT algorithm which generates one disjunctive cut in each iteration. Chapters 4 - 6 introduce several enhancements and variants of the CPT algorithm that improve the computational performance of this algorithm. Chapter 4 introduces an iterative method for solving the CGLP: Multi-cut Row Generation (MRG). Chapter 5 discusses the effects of two normalization methods for the CGLP: Weighted Cut Coefficient (WCC) and Minimum 1-norm Cut (M1NC). Computational results show that MRG, WCC and M1NC greatly enhance the computational power of the CPT algorithm. Chapter 6 introduces several other algorithmic alternatives to the CPT algorithm and gives an overall computational comparisons among all the algorithms discussed in this dissertation. Finally, Chapter 7 gives an overall summary of the work in this dissertation and recommends future research.

CHAPTER 2

LITERATURE REVIEW

2.1 Advances of General Mixed-integer Linear Program

Over the past several decades, the polyhedral study of MILP-Bs has progressed along several avenues. Disjunctive and lift-and-project cuts (Balas, 1979; Balas et al., 1993), semidefinite relaxations (Lovász and Schrijver, 1991) and reformulation-linearization technique (RLT) (Sherali and Adams, 1990, 1994) have provided alternative approaches to generate cutting planes that define the convex hull of feasible points of a MILP-B. The case for the MILP-Gs has not been as well understood, even when the integer variables are bounded. While pure integer programs can be shown to have a finite representation using Gomory cuts (Gomory, 1963), the same is not true for MILP-Gs, unless the optimal objective function is integral. Recently, Adams and Sherali (2005) 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. Thus, formation of the convex hull of integer points in MILP-Gs using RLT has been resolved, though, to our knowledge, there are no computational experiments with this method. However, formation of the convex hull of integer points in MILP-Gs using disjunctive cuts remains unresolved.

While disjunctive cuts are natural to use for MILP-Gs, a cutting plane procedure using disjunctive cuts has not been proved to be finitely convergent. Indeed, the facial disjunctive property (Balas, 1979) was deemed critical for finite convergence and this property holds for MILP-Bs, but not for MILP-Gs. Unfortunately, as shown in Figure 2 of Sen and Sherali (1985), the absence of the facial disjunctive property could lead to

infinitely many iterations. Subsequently, Owen and Mehrotra (2001) provided the proof that in the absence of the facial disjunctive property, a one-variable-at-a-time method for convexifying disjunctive sets leads to an infinite convergent process that ultimately does provide the convex hull of feasible points. Of course, one could write a binary expansion of each general integer variable and the resulting formulation is a MILP-B which can be sequentially convexified. However, Owen and Mehrotra (2002) illustrate that this approach is not practical. They discussed the issue with two reformulations of a MILP-G to a MILP-B: a compact reformulation, which requires the least number of additional binary variables, and a full reformulation, which requires one additional binary variable for each value that a integer variable can take on. They show that to convexify a MILP-G in its compact reformulation, the sequential convexification procedure of Balas et al. (1993), and the reformulation-linearization technique of Sherali and Adams (1990, 1994) have to be carried out for all the binary variables to eliminate more than half of the possible ranges for extreme points that are fractional in an integer variable. For the full reformulation, they show that all binary variables corresponding to integer values larger than the optimal integer value need to be fixed during branching.

Over the past 15 years, computations using disjunctive cuts (Balas, 1979) have appeared in a variety of contexts. The first successful implementation of a special case of disjunctive cuts for MILP-Bs within a branch-and-cut framework, known as the lift-and-project cuts (Balas et al., 1993), is described in Balas et al. (1996). Lift-and-project cuts rely on two-term (simple) disjunctions of the form $x_k \leq 0$ or $x_k \geq 1$, for some binary variable x_k. Balas and Perregaard (2002) provide a survey on the recent progress with lift-and-project cuts for MILP-Bs. In addition, they suggest a normalization constraint for the CGLP of the lift-and-project cuts. Balas and Perregaard (2003) show that lift-and-project cuts from simple disjunctions can be obtained directly from the simplex tableau of the

linear programming (LP) relaxation, without solving a larger CGLP. Bonami and Minoux (2005) provide a computational study on the strength of rank-1 lift-and-project cuts (The cuts obtained by applying a given separation procedure for a family of cuts only to the initial formulation of the problem are called rank-1 cuts) and the elementary lift-and-project closure for MILP-Bs. They conclude that rank-1 lift-and-project cuts are useful in closing a high percentage of the integrality gap in some instances of MIPLIB 3.0 (Bixby et al., 1996) even when only one round of these cuts is added.

For MILP-Gs, Balas and Saxena (2008) report computational experiments on optimizing over the elementary split closure. Split cuts are obtained from disjunctions that are more general than the simple variable disjunctions. These types of (split) disjunctions are of the form $\alpha^\top x_1 \leq \beta$ or $\alpha^\top x_1 \geq \beta + 1$, where $(\alpha, \beta) \in \mathbb{Z}^{n_1}$ and x_1 is the vector of integer variables. Despite the generality of the split disjunctions, Cook et al. (1990) provide an example to show that the split rank of a MILP-G could be infinite; in other words, a cutting plane algorithm using split cuts could take infinitely many iterations. Furthermore, the separation of split cuts is shown to be $\mathcal{NP}$-hard (Caprara and Letchford, 2003). In their computational study, Balas and Saxena (2008) solve a parametric integer program to generate a violated split cut. While the rank-1 split cuts seem to close a high percentage (about 72%) gap on average, the computational effort required for their separation is excessive (several hours/days) for most of the test instances. Finally, Zanette et al. (2008) implement Gomory's fractional cutting plane algorithm (Gomory, 1963), which is finitely convergent for MILP-Gs when the optimal objective function value is integral. They test the method on pure ILPs from MIPLIB 2003 (Achterberg et al., 2006) and MIPLIB 3.0 (Bixby et al., 1996) instances on a Intel Core 2 Q6600, 2.40GHz computer with a memory limit of 2GB for each instance. They report that an average of about 50% gap can be closed on these instances within an hour time limit using two of their heuristic methods.

CHAPTER 3

CUTTING PLANE TREE ALGORITHM

This chapter is based on a published paper (Chen et al., 2011). In this chapter, we give a finite disjunctive programming procedure to obtain the convex hull of a MILP-G with bounded integer variables. We propose a finitely convergent Convex Hull Tree (CHT) algorithm which constructs a linear program that has the same optimal solution as the associated MILP-G. In addition, we combine the standard notion of sequential cutting planes with ideas underlying the CHT algorithm to help guide the choice of disjunctions to use within a new cutting plane method, the Cutting Plane Tree (CPT) algorithm. This algorithm is shown to converge to an integral optimal solution in finitely many iterations. Finally, we illustrate the proposed algorithm on three well-known examples in the literature that require an infinite number of elementary or split disjunctions in a rudimentary cutting plane algorithm. One-cut-per-iteration algorithm is a direct implementation of the CPT algorithm which generate only one disjunctive cut for a chosen fractional variable of the solution of the linear program solved in each iteration. Computational result of the one-cut-per-iteration algorithm is presented at the end of this chapter.

3.1 Introduction

In this chapter, assuming a nonempty feasible set and bounded general integer variables of MILP-Gs, we address the following questions that are part of the questions proposed in Section 1.2.

- Is it possible to use the disjunctive programming methodology to describe the con-

vex hull of MILP-G solutions in finitely many steps without introducing binary variables?

- Is there a constructive methodology to obtain an optimal solution to a MILP-G using a disjunctive programming characterization of its convex hull?
- If we are restricted to introduce only one cutting plane in any iteration, is there a finitely convergent disjunctive programming algorithm that solves a MILP-G?

We provide answers to the above questions in the affirmative. Moreover, we provide evidence that our approach can be implemented in a manner that the convexification process is guided by the objective function, so that fewer inequalities are generated.

3.2 Convex Hull of Bounded General Mixed-integer Linear Program

Consider a mixed-integer linear program with n variables, of which the first n_1 ($n_1 \leq n$) are integer, and the remaining are continuous. Such a problem may be stated as

$$\min\{c^T x | Ax \geq b, x \in \mathbb{Z}_+^{n_1} \times \mathbb{R}_+^{n-n_1}\}, \tag{3.1}$$

where X is the set of feasible solutions, and the set obtained by relaxing the integrality requirements of X is denoted by X_L. We assume that all integer variables are bounded with $x_j \in [\ell_j, u_j]$ for all $j = 1, \ldots, n_1$. Such problems will be referred to as MILP-Gs with bounded integer variables.

Suppose now that each interval $[\ell_j, u_j]$ is divided into t_j sub-intervals $[\ell_{1j} := \ell_{1j}, u_{1j}]$, $[\ell_{2j}, u_{2j}], \ldots, [\ell_{t_j j}, u_{t_j j} := u_j]$, where $\ell_{\kappa_j j} \in \mathbb{Z}_+$ and $u_{\kappa_j j} \in \mathbb{Z}_+$, with $\ell_{\kappa_j j} \leq u_{\kappa_j j}$, define the left- and right-hand sides of interval $\kappa_j \in \{1, \ldots, t_j\}$ for $j = 1, \ldots, n_1$, and $\ell_{\kappa_j+1 j} - u_{\kappa_j j} \leq 1$ for $\kappa_j \in \{1, \ldots, t_j - 1\}$ so that the sub-intervals span all integers in $[\ell_j, u_j]$. Given a

partition $\mathcal{P}$, the collection of all n_1-tuples $\kappa := (\kappa_1, \ldots, \kappa_{n_1})$, where $\kappa_j \in \{1, \ldots, t_j\}$ for $j = 1, \ldots, n_1$, is denoted by $K(\mathcal{P})$. Then a *unit partition*, $\mathcal{P}^*$, of all integer points is a partition for which $u_{\kappa_j j} - \ell_{\kappa_j j} \leq 1$, for all $\kappa_j = 1, \ldots, t_j$, and all $j = 1, \ldots, n_1$. For a given vector $\kappa \in K(\mathcal{P}^*)$, an index $j \in \{1, \ldots, n_1\}$, and a polyhedron $\bar{X}$, we define two sets $P^-(\kappa, j, \bar{X})$ and $P^+(\kappa, j, \bar{X})$ as follows:

$$P^-(\kappa, j, \bar{X}) := \{x \in \bar{X} | \ell_{\kappa_i i} \leq x_i \leq u_{\kappa_i i}, i = 1, \ldots, n_1; x_j \leq \ell_{\kappa_j j}\},$$

$$P^+(\kappa, j, \bar{X}) := \{x \in \bar{X} | \ell_{\kappa_i i} \leq x_i \leq u_{\kappa_i i}, i = 1, \ldots, n_1; x_j \geq u_{\kappa_j j}\}.$$

We also define $\mathcal{H}^{\kappa}_j(\bar{X}) := \text{clconv}(P^-(\kappa, j, \bar{X}) \cup P^+(\kappa, j, \bar{X}) \setminus \emptyset)$, where empty sets are removed from consideration, and clconv denotes the closure of the convex hull.

Theorem 3.2.1. Assume that the set X as defined in (3.1) is non-empty and has bounded integer variables. Then for any unit partition $\mathcal{P}^*$,

$$\text{clconv}(X) = \text{clconv}\{\cup_{\kappa \in K(\mathcal{P}^*)}[\mathcal{H}^{\kappa}_{n_1}(\mathcal{H}^{\kappa}_{n_1-1}(\cdots(\mathcal{H}^{\kappa}_1(X_L))\cdots)) \setminus \emptyset]\}.$$

Proof. The set $K(\mathcal{P}^*)$ decomposes the problem into boxes of at most unit size (in $x_1 \in \mathbb{R}^{n_1}$ subspace) each of which can be sequentially convexified. To see this, note that for a given κ, the facial disjunctive description of the set $\mathcal{H}^{\kappa}_{n_1}(\mathcal{H}^{\kappa}_{n_1-1}(\cdots(\mathcal{H}^{\kappa}_1(X_L))\cdots))$, where $\ell_{\kappa_j j} \leq x_j \leq \ell_{\kappa_j j} + 1$, for all $j = 1, \ldots, n_1$, can be obtained by sequential convexification of MILP-Bs by letting $\bar{x}_j = x_j - \ell_{\kappa_j j}$, $\bar{x}_j \in \{0, 1\}$ (Balas et al., 1993; Sherali and Adams, 1994; Lovász and Schrijver, 1991). As the extreme points of these polyhedra have binary values for $\bar{x}_j$, and $\ell_{\kappa_j j}$ are integer, $j = 1, \ldots, n_1$, the polyhedron given by the union of all such polyhedra have integer values for x_j in its extreme points, and hence the result follows. □

Note that the introduction of binary variables in the proof of Theorem 3.2.1 is done only for expositional clarity. If one does not introduce the binary variables, the proof follows from sequential convexification of facial disjunctive programs.

Example 3.2.1. OM01.

Owen and Mehrotra (2001) give an example illustrating that a rudimentary cutting plane algorithm using elementary disjunctions may be "infinitely" convergent. We show on this example that the we can find its convex hull in finite iterations using a unit partition of its feasible region.

In this example,

$$X = \left\{ x \in \mathbb{Z}^2 \;\middle|\; \begin{array}{l} 8x_1 + 12x_2 \leq 27 \\ 8x_1 + 3x_2 \leq 18 \\ 0 \leq x_1, x_2 \leq 3 \end{array} \right\}, c = (-1, -1).$$

The feasible region is illustrated in Figure 3.1. For this example, a unit partition $\mathcal{P}^*$ is given by $x_j \in \{[0,1],[1,2],[2,3]\}$ for $j = 1,2$. Thus $t_j = 3$ and $\kappa_j \in \{1,2,3\}$ for $j = 1,2$. Therefore,

$$K(\mathcal{P}^*) = \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(3,1),(3,2),(3,3)\}.$$

Consider each $\kappa \in K(\mathcal{P}^*)$. For $\kappa = (1,1)$, $\mathcal{H}_2^{(1,1)}(\mathcal{H}_1^{(1,1)}(X_L))$ gives the convex hull of the points (0,0),(0,1), (1,0) and $(1,1)$; for $\kappa = (1,2)$, $\mathcal{H}_2^{(1,2)}(\mathcal{H}_1^{(1,2)}(X_L))$ gives the convex hull of the points $(0,2),(0,1),(1,1)$; for $\kappa = (2,1)$, $\mathcal{H}_2^{(2,1)}(\mathcal{H}_1^{(2,1)}(X_L))$ gives the convex hull of the points $(1,0),(2,0),(1,1)$. For $\kappa = (1,3)$, $\mathcal{H}_2^{(1,3)}(\mathcal{H}_1^{(1,3)}(X_L))$ gives the point $(0,2)$; for $\kappa = (2,2)$, $\mathcal{H}_2^{(2,2)}(\mathcal{H}_1^{(2,2)}(X_L))$ gives the point $(1,1)$; and for $\kappa = (3,1)$, $\mathcal{H}_2^{(3,1)}(\mathcal{H}_1^{(3,1)}(X_L))$ gives the point $(2,0)$. For all other $\kappa \in K(\mathcal{P}^*)$, $\mathcal{H}_2^{\kappa}(\mathcal{H}_1^{\kappa}(X_L)) = \emptyset$. Taking the union of the non-empty polytopes, we get $\text{clconv}(X)$.

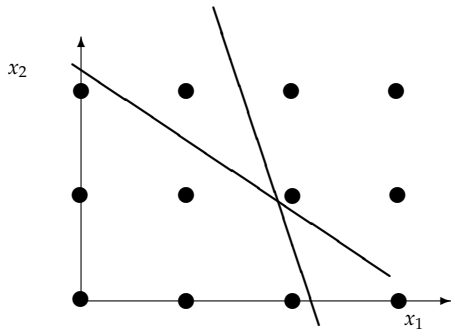

Figure 3.1: OM01: Feasible set.

In the next section, we address the question: Is there a constructive methodology to obtain an optimal solution to a MILP-G with bounded integer variables using a disjunctive programming characterization of its convex hull?

3.3 Convex Hull Tree Algorithm

The convex hull tree is a new concept in which we construct a polyhedron X_Z such that the solution to the problem

$$\min_{x \in X_Z} c^T x$$

yields an optimal solution to the original MILP-G. The construction of X_Z relies on a tree that guides the convexification process. We propose an algorithm to construct X_Z, which we refer to as the Convex Hull Tree (CHT) algorithm, because it is similar to a branch-and-bound tree, but instead of searching the tree, we form partial convex hulls based on the tree. The pseudo-code of the proposed algorithm is given in Algorithm 1.

At iteration k, an index i represents a distinct subset, Q_i, in which a solution could fall. As in branch-and-bound methods, each subset Q_i is a cross product of n_1 intervals corresponding to the bounded integer variables, and $n - n_1$ half lines corresponding to the non-negative continuous variables. The collection of all such subsets forms a set $\boldsymbol{Q}_k$.

If $x^k \in Q_i$ and $x^k_j \notin \mathbb{Z}_+$ for some $j = 1, \ldots, n_1$, then we replace Q_i with two smaller non-overlapping subsets P^-_k and P^+_k, the union of which includes all integers in Q_i (see line 4 of Algorithm 1). Whenever we encounter a subset Q_i for which $u_{ij} - \ell_{ij} \leq 1$ for $j = 1, \ldots, n_1$, we replace Q_i with the closure of the convex hull of feasible points of $Q_i \cap X_k \cap \mathbb{Z}^{n_1}$, where for simplicity of notations, $X_k \cap \mathbb{Z}^{n_1}$ refers to $X_k \cap (\mathbb{Z}^{n_1} \times \mathbb{R}^{n-n_1})$, throughout this chapter. At the end of iteration k, the collection of subsets is updated as $\mathcal{Q}_{k+1}$. We form a new set $X_{k+1} := \text{clconv}\{\cup_{Q_t \in \mathcal{Q}_{k+1}}(Q_t \cap X_k)\}$. We stop when we obtain an integral optimal solution to $x^k \in \arg\min_{x \in X_k} c^T x$ and let $X_Z = X_k$ at termination. Even though there is an underlying tree in the convex hull tree algorithm, we do not introduce a tree notation in this section to ease the exposition.

Algorithm 1 Convex Hull Tree algorithm

1: Initialization: $k = 1$, $\mathcal{Q}_k := \{Q_1 = \times_{j=1}^{n_1}[\ell_j, u_j] \times \mathbb{R}^{n-n_1}_+\}$. Let $X_k = X_L \cap Q_1$, and $x^k \in \arg\min_{x \in X_k} c^T x$. In case of multiple optima, let x^k be a vertex of X_k.
2: **while** $X_k \neq \emptyset$ and $x^k_j \notin \mathbb{Z}_+, j = 1, \ldots, n_1$ **do**
3: Choose a variable $x^k_j \notin \mathbb{Z}$. Find the (unique) subset $\bar{Q}$ in the list $\mathcal{Q}_k$ such that $x^k \in \bar{Q}$. Remove the set $\bar{Q}$ from $\mathcal{Q}_k$ and denote the revised list by $\bar{\mathcal{Q}}_k$.
4: Form two subsets $P^-_k = \{x \in \bar{Q} | x_j \leq \lfloor x^k_j \rfloor\}$ and $P^+_k = \{x \in \bar{Q} | x_j \geq \lceil x^k_j \rceil\}$. Let $P^-_k = \emptyset$ (or $P^+_k = \emptyset$) if $P^-_k \cap X_k = \emptyset$ (or $P^+_k \cap X_k = \emptyset$). Define the new list by $\mathcal{Q}_{k+1} = \bar{\mathcal{Q}}_k \cup P^-_k \cup P^+_k \setminus \emptyset$ and denote all subsets in the updated list $\mathcal{Q}_{k+1}$ by $\{Q_t\}_{t=1}^T$.
5: **for** $t = 1, \ldots, T$ **do**
6: **if** $u_{ti} - \ell_{ti} \leq 1$ for $i = 1, \ldots, n_1$ **then**
7: Let $Q_t \leftarrow \text{clconv}(Q_t \cap X_k \cap \mathbb{Z}^{n_1})$.
8: **end if**
9: **end for**
10: Form the set $X_{k+1} := \text{clconv}\{\cup_{t=1}^T (Q_t \cap X_k)\}$.
11: Let $k \leftarrow k + 1$. Find $x^k \in \arg\min_{x \in X_k} c^T x$ such that x^k is a vertex of X_k.
12: **end while**
13: Return $X_Z = X_k$.

The idea behind the CHT algorithm is to break the MILP-G feasible region into smaller, disjoint pieces and carry out convexification on these smaller pieces. The set Q_t in line 7 in Algorithm 1 is a "unit box" in $x_1 \in \mathbb{R}^{n_1}$ subspace, so we can use sequential con-

vexification of facial disjunctive programs to convexify the set $Q_t \cap X_k \cap \mathbb{Z}^{n_1}$. In this case, using the disjunctive cutting plane method (Balas, 1979) is a natural choice. It is clearly not the only choice, especially if one want to perform convexification on a piece that uses a "non-unit box" Q_t in $x_1 \in \mathbb{R}^{n_1}$ subspace (for example, when the distance between the upper and lower bounds of some integer variable x_j, $j \in \{1, \ldots, n_1\}$ is larger than 1). General purpose cutting planes such as Intersection cuts (Balas, 1971), Gomory cuts (Gomory, 1963) and reformulation-linearization technique (Sherali and Adams, 1990, 1994) etc. can also be used in this case. For MILP-Bs, lift-and-project cuts Balas et al. (1993) is a possible choice too. Moreover, some special purpose cutting planes such as flow cover inequalities (Padberg et al., 1985) and flow path inequalities (Van Roy and Wolsey, 1985) may also be used if special problem structure is derived using a "non-unit box" Q_t.

In line 3 of Algorithm 1, we currently choose the first fractional variable (i.e., the one with the smallest index) in the order $1, 2, ..., n_1$. In fact, the order does not matter since in line 7 we use sequential convexification for facial disjunctive programs.

Proposition 3.3.1. Assume that the set X as defined in (3.1) is non-empty and has bounded integer variables. Then the convex hull tree algorithm constructs X_Z in finitely many iterations.

Proof. Since the MILP-G has bounded integers, the splitting invoked in line 4 will, in finite time, yield a unit partition in the worst case, which will lead to the execution of line 7, in finite time. From Theorem 3.2.1 it follows that the convex hull tree algorithm constructs X_Z in finitely many iterations. □

Example 3.2.1 OM01. (cont.) We illustrate each iteration of the convex hull tree algorithm.

Iteration 1 Initially, $Q_1 := Q_1 = [0,3]\times[0,3]$, $X_1 = X_L$ and $x^1 = (15/8, 1)$. Because $x_1^1 \notin$

$\mathbb{Z}$ and $x^1 \in Q_1$, we replace the set Q_1 with the two sets $P_1^- = \{x \in Q_1 | x_1 \leq \lfloor \frac{15}{8} \rfloor\}$ and $P_1^+ = \{x \in Q_1 | x_1 \geq \lceil \frac{15}{8} \rceil\}$. The updated list is $\mathcal{Q}_2 := \{Q_1 = [0,1] \times [0,3], Q_2 = [2,3] \times [0,3]\}$. Hence, we have $X_2 = \text{clconv}[(X_1 \cap Q_1) \cup (X_1 \cap Q_2)]$. The facet of X_2 generated in line 10 passes through the points (1,19/12) and (2,2/3), which deletes x^1. This is the same cut as that of Owen and Mehrotra (2001). The next point obtained at the end of this iteration is $x^2 = (2, 2/3)$.

Iteration 2 Because $x_2^2 \notin \mathbb{Z}$ and $x^2 \in Q_2$, we replace the set Q_2 with the two sets $P_2^- = \{x \in Q_2 | x_2 \leq \lfloor \frac{2}{3} \rfloor\}$ and $P_2^+ = \{x \in Q_2 | x_2 \geq \lceil \frac{2}{3} \rceil\}$. The updated subsets are $Q_1 = [0,1] \times [0,3], Q_2 = [2,3] \times \{0\}, Q_3 = [2,3] \times [1,3]$. Note that $X_2 \cap Q_3$ is infeasible, so Q_3 is removed from consideration. In line 7 of Iteration 2, because $2 \leq x_1 \leq 3, x_2 = 0$, we convexify the set $\{x \in X_2 \cap Q_2 \cap \mathbb{Z}^2\}$, which results in a single feasible point $(2, 0)$, and we update the set $Q_2 \leftarrow \{2\} \times \{0\}$. Therefore, $\mathcal{Q}_3 := \{Q_1 = [0,1] \times [0,3], Q_2 = \{2\} \times \{0\}\}$. Hence, we have $X_3 = \text{clconv}[(X_2 \cap Q_1) \cup (X_2 \cap Q_2)]$. The facet of X_3 generated in line 10 goes through (1,19/12) and (2,0), which deletes x^2. As a result, we get a deeper cut than that obtained from the procedure of Owen and Mehrotra (2001). The next point obtained at the end of this iteration is $x^3 = (1, 19/12)$.

Iteration 3 Because $x_2^3 \notin \mathbb{Z}$ and $x^3 \in Q_1$, we replace the set Q_1 with the two sets $P_3^- = \{x \in Q_1 | x_2 \leq \lfloor \frac{19}{12} \rfloor\}$ and $P_3^+ = \{x \in Q_1 | x_2 \geq \lceil \frac{19}{12} \rceil\}$. The updated list is $\mathcal{Q}_4 := \{Q_1 = [0,1] \times [0,1], Q_2 = \{2\} \times \{0\}, Q_3 = [0,1] \times [2,3]\}$. Similar to Iteration 2, we convexify the set $\{x \in X_3 \cap Q_3 \cap \mathbb{Z}^2\}$, which results in a single feasible point $(0, 2)$, and we update the set $Q_3 \leftarrow \{0\} \times \{2\}$. We also convexify the set $\{x \in X_3 \cap Q_1 \cap \mathbb{Z}^2\}$, which gives Q_1 itself. Hence, we have $X_4 = \text{clconv}[(X_3 \cap Q_1) \cup (X_3 \cap Q_2) \cup (X_3 \cap Q_3)]$. The facet of X_4 generated in line 10 goes through (0,2) and (2,0) and thus gives the convex hull of X, which deletes x^3.

We illustrate these iterations in Figure 3.2.

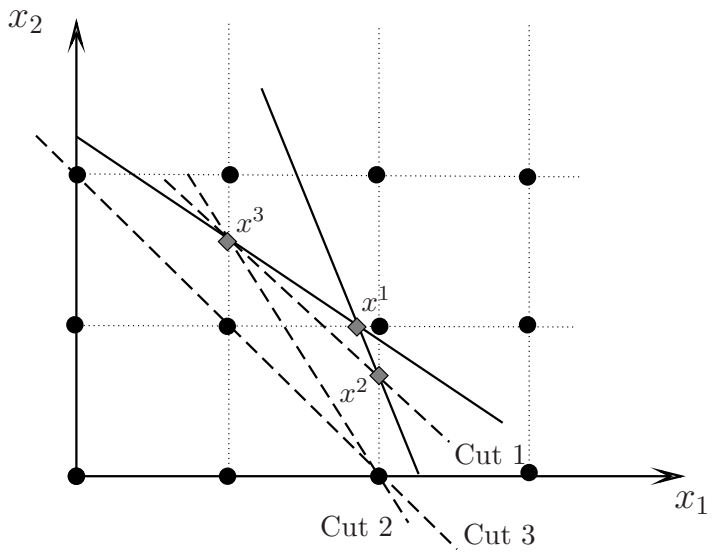

Figure 3.2: OM01: Convex hull tree cuts

3.4 Cutting Plane Tree Algorithm: One-cut-per-iteration

In this section, we address the question: If we are restricted to introduce only one cutting plane in any iteration, is there a finitely convergent disjunctive programming algorithm that solves a MILP-G? In other words, we consider a simplification of Algorithm 1, where only one disjunctive cut is generated in each iteration, which we refer to as the "one-cut-per-iteration" algorithm.

In the cutting plane tree $\mathcal{T}$, there is a single root node o. For each node $\sigma \in \mathcal{T}$, an integer m_σ keeps track of the cutting planes that will be used to generate a disjunctive cut when this node is revisited, an integer $v_\sigma \in \{1, 2, \ldots, n_1\}$ stores the index of the integer variable that is split, an integer q_σ stores the (lower) level of the splitting. Let l_σ, r_σ and p_σ denote links to the left child, right child and parent nodes of node σ, respectively. Let $\mathcal{S}(\sigma)$ be all nodes on the subtree rooted at node σ (not including node σ). Let $\mathcal{N}(\sigma)$ be

the collection of the nodes on the path from the root node to node σ (not including the root node), let $\mathcal{N}^-(\sigma)$ be the collection of nodes in $\mathcal{N}(\sigma)$ that were formed as the left child node of its parent, and let $\mathcal{N}^+(\sigma)$ be the collection of nodes in $\mathcal{N}(\sigma)$ that were formed as the right child node of its parent. Given $\sigma \in \mathcal{T}$ define

$$C_\sigma = \{x | x_j \in [\ell_j, u_j], x_{v_{p_s}} \leq q_{p_s}, \forall s \in \mathcal{N}^-(\sigma), x_{v_{p_s}} \geq q_{p_s} + 1, \forall s \in \mathcal{N}^+(\sigma)\}. \quad (3.2)$$

We let m_σ store an iteration index, which gives the set X_{m_σ} to be used in the CGLP (Balas, 1979; Sherali and Shetty, 1980). The set X_{m_σ} corresponds to X_L together with the first $m_\sigma - 1$ cuts added to it. If $X_{m_\sigma} \cap C_{l_\sigma} = \emptyset$ ($X_{m_\sigma} \cap C_{r_\sigma} = \emptyset$), we say that the left (right) child node of σ is "fathomed", i.e., $l_\sigma = null$ ($r_\sigma = null$).

Now that we have introduced the tree notations necessary for describing the one-cut-per-iteration Algorithm 2, we relate the notation in this section to the notation introduced in Section 3.2 and Section 3.3. In Algorithm 2, $\mathcal{L}_{k+1}$ denotes the collection of all leaf nodes of the cutting plane tree at the end of iteration k, it will be used to generate a disjunctive cut in iteration k. Note that, each feasible region given by C_σ for $\sigma \in \mathcal{L}_{k+1}$ defines a subset Q_i in iteration k. The collection of these non-overlapping subsets Q_i, for $i = 1, \ldots, |\mathcal{L}_{k+1}|$ at iteration k, denoted by $\mathcal{Q}_{k+1}$, gives a valid partition, $\mathcal{P}$ of $\times_{j=1}^{n_1}[0, u_j] \times \mathbb{R}_+^{n-n_1}$.

The pseudo-code of the one-cut-per-iteration algorithm is given in Algorithm 2. At iteration k, if the current extreme point solution to $\min_{x \in X_k} c^T x$, given by x^k is integral, then we have found the optimal solution to the MILP-G. Otherwise, we search the cutting plane tree, to find the last node σ on the path from the root node such that $x^k \in C_\sigma$ (this is called "the algorithm **visits** node σ at iteration k" or "x^k visits node σ" or "x^k **falls in** node σ"). There are two cases: Case (1) σ is a leaf node ($\sigma \in \mathcal{L}_k$), Case (2) σ is not a leaf node ($\sigma \notin \mathcal{L}_k$, $x^k \in C_\sigma$, $x^k \notin C_{l_\sigma}$ and $x^k \notin C_{r_\sigma}$). In Case (1), we choose a fractional

variable x_j, $j = 1, \ldots, n_1$ with the smallest index, and let the split variable be $v_\sigma = j$. We create two new nodes: left (l_σ) and right (r_σ) children of σ at the split level $q_\sigma = \lfloor x_j \rfloor$. We let $C_{l_\sigma} = \{x \in C_\sigma | x_j \leq \lfloor x_j^k \rfloor\}$ and $C_{r_\sigma} = \{x \in C_\sigma | x_j \geq \lceil x_j^k \rceil\}$. In this case, we also let $m_\sigma = k$, as this is the first time the tree search for a fractional solution stops at node σ. In Case (2), the cutting plane tree and m_σ are unchanged. However, in this case, we update $m_t = k$ for all successors of σ, $t \in \mathcal{S}(\sigma)$. We generate a valid inequality for the set $\text{clconv}\{\cup_{t \in \mathcal{L}_{k+1}} (X_{m_\sigma} \cap C_t)\}$ that cuts off x^k (from an extreme point of the CGLP). The new inequality is included along with those defining X_k, and the resulting set is denoted X_{k+1}. This process continues until one of the stopping criteria is satisfied.

Algorithm 2 Cutting Plane Tree (one-cut-per-iteration) algorithm

Initialization: $k = 1$, create a node o, where $p_o = null$, $m_o = 1$. Let $\mathcal{T} = \{o\}$, $\mathcal{L}_k := \{o\}$, $C_o = \{x | x_j \in [\ell_j, u_j], j = 1, \ldots, n_1\}$, $X_k = X_L$ and $x^k \in \arg\min_{x \in X_k} c^T x$, where x^k is a vertex of X_k.
while $X_k \neq \emptyset$ and $x^k_j \notin \mathbb{Z}_+$, $j = 1, \ldots, n_1$ **do**
  Search for node $\sigma \in \mathcal{T}$ such that: either $\sigma \in \mathcal{L}_k$ and $x^k \in C_\sigma$; or, node $\sigma \notin \mathcal{L}_k$ and $x^k \in C_\sigma$, $x^k \notin C_{l_\sigma}$ and $x^k \notin C_{r_\sigma}$.
  if $\sigma \in \mathcal{L}_k$ **then**
    Choose the smallest index j such that variable $x^k_j \notin \mathbb{Z}$. Let $m_\sigma = k$. Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k \setminus \{\sigma\}$.
    Create a node l^- with $l_\sigma = l^-$, $p_{l^-} = \sigma$ and a node l^+ with $r_\sigma = l^+$, $p_{l^+} = \sigma$.
    Let $C_{l^-} = \{x \in C_\sigma | x_j \leq \lfloor x^k_j \rfloor\}$ and $C_{l^+} = \{x \in C_\sigma | x_j \geq \lceil x^k_j \rceil\}$.
    if $C_{l^-} \cap X_k \neq \emptyset$ **then**
      Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_{k+1} \cup \{l^-\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^-\}$.
    else
      Let $l^- = null$ (fathom).
    end if
    if $C_{l^+} \cap X_k \neq \emptyset$ **then**
      Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_{k+1} \cup \{l^+\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^+\}$.
    else
      Let $l^+ = null$ (fathom).
    end if
    Let $v_\sigma = j$, $q_\sigma = \lfloor x^k_j \rfloor$.
  else if $\sigma \notin \mathcal{L}_k$ and $x^k \in C_\sigma$, $x^k \notin C_{l_\sigma}$ and $x^k \notin C_{r_\sigma}$ **then**
    Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k$, let $m_t \leftarrow k, \forall t \in \mathcal{S}(\sigma)$.
  end if
  Generate a valid inequality for the set $\text{clconv}\{\cup_{t \in \mathcal{L}_{k+1}} (X_{m_\sigma} \cap C_t)\}$ that cuts off x^k (using an extreme point of the CGLP). Call the set with the additional valid inequality X_{k+1}.
  Let $k \leftarrow k + 1$ and $x^k \in \arg\min_{x \in X_k} c^T x$, where x^k is a vertex of X_k.
end while

Proposition 3.4.1. Assume that the set X as defined in (3.1) is non-empty and has bounded integer variables. Then Algorithm 2 converges to the optimal solution in finitely many iterations.

Proof. Note that the cutting plane tree cannot expand infinitely. In the worst case, the leaf nodes of the tree form a unit partition $\mathcal{P}^*$. Therefore, the number of leaf nodes is no

more than $\Pi_{j=1}^{n_1}(u_j - \ell_j + 1)$, where ℓ_j, u_j are the integer lower and upper bound of variable x_j, $j = 1, \ldots, n_1$. Let σ be the node found in line 3 of Algorithm 2 at iteration k. There are two cases: (1) $\sigma \in \mathcal{L}_k$ and $x^k \in C_\sigma$, or (2) $\sigma \notin \mathcal{L}_k$, $x^k \in C_\sigma$, $x^k \notin C_{l_\sigma}$ and $x^k \notin C_{r_\sigma}$. In Case (1), the cutting plane tree is expanded by splitting from node σ, and we update the collection of leaf nodes as $\mathcal{L}_{k+1}$. As the number of possible leaf nodes is finite, this case can happen only finitely many times. In Case (2), the tree remains constant and we have $\mathcal{L}_{k+1} = \mathcal{L}_k$. In either case, 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 all facets of $\text{clconv}\{\cup_{t\in\mathcal{L}_{k+1}}(X_{m_\sigma} \cap C_t)\}$ we will have either $x_{v_\sigma} \leq q_\sigma$ or $x_{v_\sigma} \geq q_\sigma + 1$ in finitely many iterations. In either case, the tree search in line 3 stops at l_σ, r_σ or one of its successors. As a result, the algorithm can visit node σ only finitely many times. Consequently, there exists a finite number, N, such that either the algorithm stops before iteration N, or at iteration N, a unit partition $\mathcal{P}^*$ is found.

If we reach a point where the leaf nodes correspond to a unit partition $\mathcal{P}^*$, then each path from the root node of $\mathcal{T}$ to a leaf node corresponds to a vector $\kappa \in K(\mathcal{P}^*)$ as defined in Section 3.2. The order in which the variables are split on this path defines a sequence. Note that the variable x_j can be split more than once, in which case we consider the last node at which it was split, denoted by σ_j. Without loss of generality, assume that this sequence of variables is given by $1, \ldots, n_1$ so that $\sigma_j \in \mathcal{S}(\sigma_i)$ for $i < j$. As a result, when a node σ_i is visited and a cut is added at this node, we update m_{σ_j} to include this cut for all $\sigma_j \in \mathcal{S}(\sigma_i)$. Therefore, for a given κ, we sequentially convexify $X_{m_{\sigma_1}}$ with respect to x_1 first. We convexify the resulting set with respect to x_2 next, and continue until we reach variable x_{n_1}. The union of the nonempty sets defined by each leaf node (or each $\kappa \in K(\mathcal{P}^*)$) gives us $\text{clconv}(X)$ from Theorem 3.2.1. Therefore, Algorithm 2 converges to the optimal solution in finitely many iterations. □

The update/use of the "**memory**" m_σ for node σ has deep connections with the sequential convexification of facial disjunctive programs and the non-convergence issue found in a disjunctive cutting plane procedure using two-term (elementary) disjunctive cuts to cut away the solution to the most recent LP relaxation (Sen and Sherali (1985)). At iteration k, three places in Algorithm 2 update or use m_σ, they are:

(1) If solution x^k falls in a leaf node σ (step 5), then m_σ is updated to the current iteration k;

(2) If solution x^k falls in a non-leaf node σ (step 20), then m_σ remains untouched, but for all nodes $t \in \mathcal{S}(\sigma)$, m_t is updated to k;

(3) To generate a disjunctive cut (step 22), X_{m_σ} is used, which is the set X_L augmented by the first $m_\sigma - 1$ disjunctive cuts.

The intent of m_σ is to keep track of a set of disjunctive cuts, of which, a subset defines a face F_σ, which is a common face for all $C_t \cap X_{m_\sigma}$, $t \in \mathcal{S}(\sigma) \cup \mathcal{L}(\sigma)$. To see this clearer, suppose when the algorithm stops, the set of leaf nodes $\mathcal{L}(\sigma)$ of the subtree rooted at σ constitutes a unit partition of set C_σ. For a given $t \in \mathcal{L}(\sigma)$, each node in $\{o\} \cup \mathcal{N}(t) \setminus \{t\}$ specifies a variable branched on it. Without loss of generality, Let iteration k_σ be the last iteration that the algorithm visits any ancestor node of σ, and let the order of the variable being branched along the path from node o to node t be $1, 2, ..., n_1$, where variable x_{v_σ} is of course one of them. Then the m_σ value after iteration k_σ is keeping track of the set of cuts that are performing the $\mathcal{H}^t_j(...(\mathcal{H}^t_1(X)))$ operation, which is part of the sequential convexification.

In (1) above, this is when x^k falls into a leaf node σ, so iteration k is the first iteration in which the algorithm visits node σ. In this iteration, m_σ is set to k, which explicitly excludes the k-th cut that will be generated at the end of iteration k. This is because at this point, the algorithm knows that this k-th cut is not part of the disjunctive cuts that defines

the face F_σ.

In (2) above, when a non-leaf node is visited at iteration k, which means a fractional solution (to be cut away) appears on the face F_σ, so a disjunctive cut will be generated in this iteration, this cut cuts away the fractional solution on face F_σ, hence it will be passed down the subtree, because face F_σ is a common face of all $C_t \cap X_{m_\sigma}$, $t \in \mathcal{S}(\sigma)$.

m_σ also plays another important role as a mechanism to prevent the algorithm from infinitely convergent to non-optimal solutions. This design is rooted in the research of Sen and Sherali (1985), where they found that for a disjunctive cutting plane procedure using elementary disjunctive cuts, if the procedure always generates a cut using the latest LP relaxation (enhanced by all disjunctive cuts previously generated), the algorithm may take infinite steps and converge to non-optimal points, even the disjunctive cuts generated in each step is strong (facet-defining). They propose a theoretical result showing that there has to be a certain "memory" mechanism to guarantee finitely convergence. In two iteration $k < k'$, where k is the smallest iteration such that x^k and $x^{k'}$ is defined by the same face, then when the algorithm generates a cut in iteration k', it should use the first $k-1$ cuts, instead of all previous $k'-1$ cuts. In the CPT algorithm, we are facing the same situation for each leaf node of a cutting plane tree. Suppose in iteration k, the CPT algorithm visits leaf node σ and $m_\sigma = k$, at a later iteration k', the algorithm visits node σ again, and if between iterations k and k', the algorithm does not visit any ancestor of σ so that m_σ is not changed from iteration $k+1$ to k', then to generate a disjunctive cut in step 22 at iteration k', only the first $m_\sigma - 1 = k-1$ cuts are used to strengthen X_L to X_{m_σ}. For more details about how "memoryless" leads to non-convergence please refer to Sen and Sherali (1985).

The reader might find it interesting to compare the concepts introduced in this chapter with a recent paper by Jörg (2007) who also addresses the finiteness of disjunctive pro-

gramming for solving MILP-Gs. His approach proposes the use of cuts obtained from multiple split disjunctions in the projected space of integer variables. The author proved that for bounded MILP, there exists a conjunction of $n_1 + 1$ split disjunctions that characterizes the mixed-integer hull of the MILP. The author devises a two phase algorithm as follows: Phase 1) solve a relaxation, and ascertain whether the optimal face includes any integer points. If an integer optimum is identified then the method stops. Otherwise, the fractional optimum point is deleted using a Gomory cut. This phase continues until the final fractional alternative optimum is identified as x^*. This point is deleted in the projected space, which constitutes the second phase: Phase 2) find all extreme directions of a polyhedral cone to reduce the inequalities to the space of the integer variables. Now using a conjunction of multiple split disjunctions one can delete the projection of x^* in the projected space of integer variables. The process then repeats from phase 1. The author proves that this process is finite. However, the identification of all alternative optima (in phase 1), as well as the identification of all extreme directions (in phase 2) suggest that the algorithmic map producing the subsequence of polyhedra (observed after successive phase 2 iterations) involves the solution of multiple NP-hard problems. To our knowledge, there are no computational results on this algorithm.

Another comment on Algorithm 2 is about the connection between the cutting plane tree and life-and-project cuts. As we mentioned, the cutting plane tree structure is similar to the branch-and-bound tree structure in the sense they both are partitions of the feasible solution space. Such similarity enables one to use lift-and-project cuts in the CPT algorithm, if binary variables are involved (not necessarily MILP-Bs). Suppose for the given MILP-G, variables $1, ..., p$, $p + 1, ..., n_1$, $n_1 + 1, ..., n$ are binary, general integer and continuous nonnegative variables, respectively. When the algorithm visits a node σ of the tree $\mathcal{T}$, and the variables branched on all ancestor nodes of σ are binary variables, these

"fixed" variable can be separated to two sets F_B^- and F_B^+, for variables fixed at 0 and variables fixed at 1 in solution x^k. Let $R = F_B^- \cup F_B^+ \cup \{i \in \{n_1 + 1, ..., n\} | x_i^k > 0\}$ for solution x^k on node σ. Let x^R denotes the variables in R. One can then generated a disjunctive cut $\pi^R x^R \geq \pi_0$ by solving a CGLP using only the subtree rooted at σ (restricted CGLP). This cut is a cut in x^R space and is only valid for node σ and its descendant nodes. However, this cut can be "lifted" to a cut $\pi x \geq \pi_0$ using approach similar to Balas et al. (1993), by calculating coefficient $\pi^{\{1,...,n\}\setminus R}$ using solutions from the CGLP. This latter cut is valid for the whole cutting plane tree, and cuts away x^k. However, if some general integer variables are branched on ancestor nodes of σ, the above argument does not hold.

3.5 Examples from the Literature

In this section, we illustrate finite convergence of the one-cut-per-iteration algorithm for three examples from the literature (Cook et al., 1990; Owen and Mehrotra, 2001; Sen and Sherali, 1985). Each of these examples illustrates specific properties of the one-cut-per-iteration algorithm and to the best of our knowledge, most of the examples have defied finite convergence for various approaches based on linear disjunctions.

Example 3.2.1 OM01. (cont.) This example illustrates that finite convergence of the cutting plane tree algorithm can be obtained even when the facial disjunctive property is absent. We observe that using the one-cut-per-iteration algorithm, OM01 is solved in 7 iterations. Table 3.1 illustrates each iteration of Algorithm 2. Table 3.1 also provides the node σ that is visited and the value of m_σ for each iteration. Note that Case (1) of the proof of Proposition 3.4.1 applies at each iteration. Figure 3.3 depicts the first and the last polyhedra. We refer the reader to Figure A.1 in Appendix A for an illustration of the cutting plane tree at termination.

Table 3.1: OM01 example

k	x^k	σ	m_σ	Q_{k+1}	cut
1	$(15/8, 1)$	1	1	$Q_1 = [0,1] \times [0,3]$ $Q_2 = [2,3] \times [0,3]$	$11/12x_1 + x_2 \leq 5/2$
2	$(2, 2/3)$	3	2	$Q_1 = [0,1] \times [0,3]$ $Q_2 = [2,3] \times [0,0]$	$x_1 + 15/19x_2 \leq 9/4$
3	$(1, 19/12)$	2	3	$Q_1 = [2,3] \times [0,0]$ $Q_2 = [0,1] \times [0,1]$ $Q_3 = [0,1] \times [2,3]$	$x_1 + 15/16x_2 \leq 9/4$
4	$(3/8, 2)$	6	4	$Q_1 = [2,3] \times [0,0]$ $Q_2 = [0,1] \times [0,1]$ $Q_3 = [0,0] \times [2,3]$	$x_1 + x_2 \leq 9/4$
5	$(9/4, 0)$	4	5	$Q_1 = [0,1] \times [0,1]$ $Q_2 = [0,0] \times [2,3]$ $Q_3 = [2,2] \times [0,0]$	$9x_1 + 8x_2 \leq 18$
6	$(0, 9/4)$	7	6	$Q_1 = [0,1] \times [0,1]$ $Q_2 = [2,2] \times [0,0]$ $Q_3 = [0,0] \times [2,2]$	$x_1 + x_2 \leq 2$
7	$(2, 0)$				

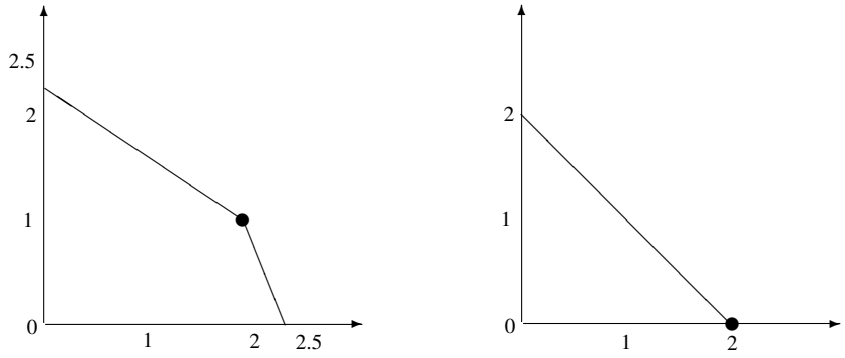

Figure 3.3: OM01: the first and last X_k.

For this problem, Owen and Mehrotra (2001) use a cutting plane procedure where all cutting planes are generated from variable disjunctions that are violated at an optimal solution to the current LP relaxation. They show that such an algorithm can infinitely approach the points (2.25, 0) and (0, 2.25), but can never cut away these points, hence the algorithm can never find optimal solutions. From Table 3.1, we see that in the CPT

algorithm, these two points becomes the extreme points of the LP relaxation problem in iteration 2 and 4, and are cut away in iteration 5 and 6.

A similar example without previously known finite convergence of disjunctive cuts, appears in Figure 2 of Sen and Sherali (1985). Since the behavior of the one-cut-per-iteration algorithm for that instance is similar to the above illustration, we omit it from the present discussion.

Example 3.5.1. CKS90. Unlike the pure integer linear program in Example 3.2.1, Cook et al. (1990) provide a MILP example which illustrates that a cutting plane procedure based on split cuts could take infinitely many iterations. In contrast to split or elementary disjunctions, Algorithm 2 works on multi-term disjunctions, which when guided by the tree, leads to a finitely convergent algorithm that provides an integral solution. For this example, Andersen et al. (2007) use inequalities derived from two rows of the simplex tableau and Li and Richard (2008) use inequalities derived from a nonlinear (multiplicative) disjunction to get the optimal solution in finite steps. In this example:

$$X = \left\{ x \in \mathbb{Z}^2 \times \mathbb{R}_+ \;\middle|\; \begin{array}{l} x_1 - x_3 \geq 0 \\ x_2 - x_3 \geq 0 \\ x_1 + x_2 + 2x_3 \leq 2 \\ 0 \leq x_1, x_2 \leq 2 \end{array} \right\}, c = (0, 0, -1).$$

Table 3.2 provides a summary of each iteration of Algorithm 2. Figure 3.4 depicts the first and the last polyhedra. Figure A.2 in Appendix A provides a more detailed illustration of the cutting plane tree at each iteration.

Example 3.5.2. SS85. This example appears as Figure 1 in Sen and Sherali (1985) where they show that sequential cutting plane algorithms can fail to converge when appropriate

Table 3.2: CKS90 example

k	x^k	σ	m_σ	Q_{k+1}	cut
1	$(1/2, 1/2, 1/2)$	1	1	$Q_1 = [0,0] \times [0,2] \times \mathbb{R}_+$ $Q_2 = [1,2] \times [0,2] \times \mathbb{R}_+$	$x_1 - 3x_3 \geq 0$
2	$(1, 1/3, 1/3)$	3	2	$Q_1 = [0,0] \times [0,2] \times \mathbb{R}_+$ $Q_2 = [1,2] \times [0,0] \times \mathbb{R}_+$ $Q_3 = [1,2] \times [1,2] \times \mathbb{R}_+$	$x_3 \leq 0$
3	$(0,0,0)$				

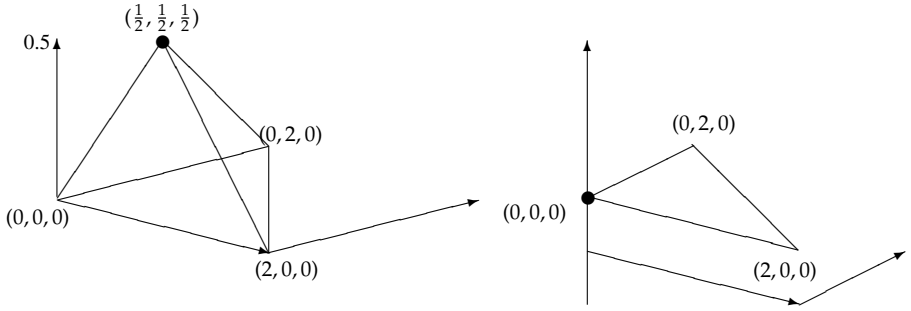

Figure 3.4: CKS90: the first and last X_k.

memory of cuts is not maintained. The tree in algorithm refalg:onecut provides a convenient mechanism to record such memory (via m_σ). In this example:

$$X = \left\{ x \in \mathbb{Z}^3 \;\middle|\; \begin{array}{l} x_1 + 2x_2 - 2x_3 \geq 0 \\ 2x_1 + 2x_2 - 3x_3 \geq 0 \\ 2x_1 + x_2 - 2x_3 \geq 0 \\ 2x_1 + 2x_2 \leq 3 \\ 0 \leq x_1, x_2, x_3 \leq 1 \end{array} \right\}, c = (0, 0, -1).$$

Table 3.3 illustrates each iteration of Algorithm 2. Note that in all previous examples, at each iteration k, Case (1) in the proof of Proposition 3.4.1 applies, and so the set X_{m_σ} used in CGLP is equivalent to X_k. However, for SS85, in iteration 3, case (2) of the proof of Proposition 3.4.1 applies, and X_{m_σ} used in CGLP is different than X_k. Figure 3.5 depicts the first and the last polyhedra. We refer the reader to Figure A.3 in Appendix A

for an illustration of the cutting plane tree at termination.

Table 3.3: SS85 example

k	x^k	σ	m_σ	Q_{k+1}	cut
1	$(1, 1/2, 1)$	1	1	$Q_1 = [0,1] \times [0,0] \times [0,1]$ $Q_2 = [0,1] \times [1,1] \times [0,1]$	$x_1 + 1/2x_2 \leq 1$
2	$(1/2, 1, 1)$	3	2	$Q_1 = [0,1] \times [0,0] \times [0,1]$ $Q_2 = [0,0] \times [1,1] \times [0,1]$	$x_1 + x_2 - 2x_3 \geq 0$
3	$(1/2, 1, 3/4)$	3	2	$Q_1 = [0,1] \times [0,0] \times [0,1]$ $Q_2 = [0,0] \times [1,1] \times [0,1]$	$x_1 + x_2 \leq 1$
4	$(1, 0, 1/2)$	2	4	$Q_1 = [0,0] \times [1,1] \times [0,1]$ $Q_2 = [0,1] \times [0,0] \times [0,0]$	$x_2 - 2x_3 \geq 0$
5	$(0, 1, 1/2)$	4	5	$Q_1 = [0,1] \times [0,0] \times [0,0]$ $Q_2 = [0,0] \times [1,1] \times [0,0]$	$x_3 \leq 0$
6	$(0, 0, 0)$				

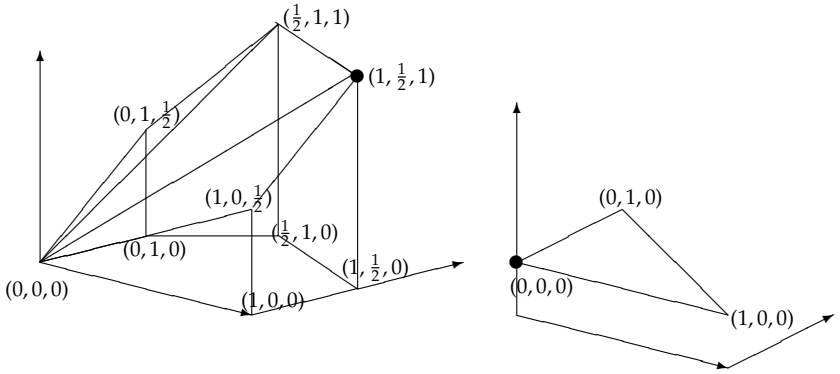

Figure 3.5: SS85: the first and last X_k.

In all previous examples, our algorithm obtains the convex hull of feasible solutions in the last polyhedron, while in general this is not always true. Note that in all of the examples, the number of nonempty subsets Q_i in any iteration is modest. This observation provides preliminary evidence that our approach can be implemented in a manner that the convexification process is guided by the objective function, and consequently, fewer inequalities may be generated. However, for large instances, it may become necessary to design more efficient ways to solve CGLP using special purpose techniques in the

spirit of (Perregaard and Balas, 2001; Balas and Perregaard, 2003). In the next section, we will formulate the CGLP used in the one-cut-per-iteration Algorithm 2. Later in the dissertation, we will introduce more effective techniques to solve the CGLP.

3.6 Cut Generation Linear Program

In Algorithm 2, we did not mention in line 22 how the CGLP is solved. This section clarifies the CGLP formulation. At iteration k for non-integral solution x^k, one fractional variable $j \in \{1, \ldots, n_1\}$ is chosen. Formulate set C_t as in (3.2), we can represent $X_m \cap C_t$ with $\{x | A_m x \geq b_m, L_t \leq x \leq U_t\}$ for $t \in \mathcal{L}_{k+1}$, where A_m, b_m are the cut-enhanced matrix and right-hand-side vector of the MILP-G, with $m = m_\sigma$. Let the multipliers used in the CGLP be $\{\lambda_t, \mu_t, \nu_t\}_{t \in \mathcal{L}_{k+1}}$, where λ_t corresponds to the vector of multipliers associated with $A_m x \geq b_m$, and μ_t, ν_t denote the vectors of multipliers for the lower and upper bound constraints, $L_t \leq x \leq U_t$, respectively. The CGLP has the form:

$$\bar{z} = \max \quad \pi_0 - \pi^T x^k \tag{3.3}$$

$$\text{s.t.} \quad \pi = \lambda_t^T A_m + \mu_t - \nu_t \qquad \forall t \in \mathcal{L}_{k+1} \tag{3.4}$$

$$\pi_0 \leq \lambda_t^T b_m + \mu_t^T L_t - \nu_t^T U_t \quad \forall t \in \mathcal{L}_{k+1} \tag{3.5}$$

$$(\lambda, \mu, \nu) \geq 0. \tag{3.6}$$

The optimal solution to the CGLP yields an inequality $\pi x \geq \pi_0$ valid for X that cuts off x^k.

The feasible set defined by (3.4)-(3.6) is a cone, and its extreme rays correspond to valid inequalities including all facets of the disjunctive program at hand. To truncate the feasible set and obtain a bounded solution, a normalization constraint is often included, although the inclusion of it may generate extreme points of CGLP that do not correspond

to facets of the convex hull (closure) of the disjunctive set. In Algorithm 2, the following normalization is used

$$-1 \leq \pi_j \leq 1, \quad \forall j = 1, \ldots, n. \tag{3.7}$$

3.7 Computational Results

In this section, the computational study of the one-cut-per-iteration algorithm is presented. In each iteration, the CGLP chooses to use the first fractional variable (from solution x^k of the linear program solved in each iteration) or the one that is given by searching the cutting plane tree $\mathcal{T}$. Only one disjunctive cut is generated per iteration.

Table 3.4 gives a summary of the 40 MILP instances. These instances are from MIPLIB 2.0, 3.0 and 2003 (Bixby et al., 1992, 1996; Achterberg et al., 2006) that have less than 1000 variables and less than 1000 constraints. Instances *noswot* is excluded from the set because the LP relaxation objective values fluctuates, i.e., sometimes it drops below previous values. Instance *enigma* is excluded because it has no integrality gap.

Table 3.4: Summary of the general mixed integer linear program instances

problem	row	col	nonzero	density	# $\mathbb{Z}$	# $\mathbb{B}$	initial	optimal/best
aflow30a	479	842	2091	0.52 %	0	421	983.17	1158.00
danoint	664	521	3232	0.93 %	0	56	62.64	65.67
dcmulti	290	548	1315	0.83 %	0	75	183975.54	188182.00
egout	98	141	282	2.04 %	0	55	149.59	568.10
fixnet6	478	878	1756	0.42 %	0	378	1200.88	3983.00
glass4	396	322	1815	1.42 %	0	302	800002400.00	1200012600.00
lseu	28	89	309	12.40 %	0	89	834.68	1120.00
markshare1	6	62	312	83.87 %	0	50	0.00	1.00
markshare2	7	74	434	83.78 %	0	60	0.00	1.00
mas74	13	151	1706	86.91 %	0	150	10482.80	11801.19
mas76	12	151	1640	90.51 %	0	150	38893.90	40005.05
misc03	96	160	2053	13.37 %	0	159	1910.00	3360.00
misc07	212	260	8619	15.64 %	0	259	1415.00	2810.00
mod008	6	319	1243	64.94 %	0	319	290.93	307.00
modglob	291	422	968	0.79 %	0	98	20430947.62	20740508.00
opt1217	64	769	1542	3.13 %	0	768	-20.02	-16.00
p0033	16	33	98	18.56 %	0	33	2520.57	3089.00
p0201	133	201	1923	7.19 %	0	201	6875.00	7615.00
p0282	241	282	1966	2.89 %	0	282	176867.50	258411.00
p0548	176	548	1711	1.77 %	0	548	315.29	8691.00
pk1	45	86	915	23.64 %	0	55	0.00	11.00
pp08a	136	240	480	1.47 %	0	64	2748.35	7350.00
pp08aCUTS	246	240	839	1.42 %	0	64	5480.61	7350.00
qiu	1192	840	3432	0.34 %	0	48	-931.64	-132.87
rgn	24	180	460	10.65 %	0	100	48.80	82.20
set1ch	492	712	1412	0.40 %	0	240	32007.73	54537.70
stein15	36	15	120	22.22 %	0	15	7.00	9.00
stein27	118	27	378	11.86 %	0	27	13.00	18.00
stein45	331	45	1034	6.94 %	0	45	22.00	30.00
vpm1	234	378	749	0.85 %	0	168	15.42	20.00
vpm2	234	378	917	1.04 %	0	168	9.89	13.75
bell3a	123	133	347	2.12 %	32	39	862578.64	878430.32
bell5	91	104	266	2.81 %	28	30	8608417.95	8966406.49
blend2	274	353	1409	1.46 %	33	231	6.92	7.60
flugpl	18	18	46	14.20 %	11	0	1167185.73	1201500.00
gen	780	870	2592	0.38 %	6	144	112130.04	112313.00
gt2	29	188	376	6.90 %	164	24	13460.23	21166.00
rout	291	556	2431	1.50 %	15	300	981.86	1077.56
timtab1	171	397	829	1.22 %	107	64	28694.00	764772.00
timtab2	294	675	1482	0.75 %	181	113	83592.00	1096557.00

The 40 instances are grouped into two sets, one set contains 9 MILP-Gs, the other set contains the rest 31 MILP-Bs. In Table 3.4, column "row" gives the number of rows of the instance, column "col" gives the number of columns of the instance, column "nonzero"

gives the number of nonzero coefficients in the constraint matrix of the instance, column "density" gives the density of the constraint matrix (density is defined as "row" × "col" / "nonzero" ×100%), column "#$\mathbb{Z}$" gives the number of general integer variables of the instance, column "#$\mathbb{B}$" gives the number of binary variables of the instance, column "initial" gives the objective value of the initial linear program of the instance (also denoted as v^0), column "optimal/best" gives the optimal (or best known) objective value of the MILP-G instance (also denote as v^*). The table is separated into two parts with the bottom part listing the set of MILP-G instances.

In Table 3.5, we report the computational results of the one-cut-per-iteration algorithm for the set of MILP-G and MILP-B instances described in Table 3.4. The runs stop normally either because an integer optimal solution is found or the time limit is exceeded. The time limit is 3600 seconds. When the runs stops normally, the objective function value of the linear program is recorded (denoted as v). Because the implementation delegates cut generation to the LP solver, we are not able to enforce the time limit strictly during cut generation, hence the run time may go beyond the pre-specified time limit.

For each instance, we report the run time in seconds ("time"), the number of disjunctive cuts generated ("cuts"), the number of CPT nodes (N), the number of leaf nodes (disjunctions) in CPT tree (T), and the percentage gap closure (%*gapcl*, calculated as $(v - v^0)/(v^* - v^0) \times 100\%$). The asterisk * marked in the "time" column of Table 3.5 for some instances indicate numerical difficulties that cause early termination. All runs are completed on a 3.2 GHz Sun workstation with 4 GB RAM.

We use a parameter ϵ as defining the stopping rule of the runs. When $\min\left\{x_j^k - \lfloor x_j^k \rfloor, \lceil x_j^k \rceil - x_j^k\right\} \leq \epsilon$, x_j^k is considered an integer. We set $\epsilon = 1e - 8$ in the computational runs for the one-cut-per-iteration algorithm.

Table 3.5: Computational results: one-cut-per-iteration

problem	time	cuts	N	T	% gapcl
aflow30a	3613	191	9	5	5.5
danoint	3638	254	3	2	0.3
dcmulti	3618	507	5	3	9.8
egout	3605	923	88	20	21.0
fixnet6	3601	655	5	3	4.5
glass4	1 *	3	5	3	0.0
lseu	3605	1473	53	27	13.3
markshare1	3601	514	87	44	0.0
markshare2	3605	308	87	44	0.0
mas74	3603	412	15	8	83.8
mas76	3607	529	11	6	49.3
misc03	3610	744	69	35	30.0
misc07	3617	308	29	15	4.3
mod008	3603	209	29	15	13.6
modglob	3644	675	7	4	3.1
opt1217	3813	33	31	16	0.0
p0033	1687 *	2043	56	25	10.5
p0201	3605	628	15	8	12.5
p0282	3608	556	21	11	0.4
p0548	3617	372	20	10	0.0
pk1	3606	607	45	23	0.0
pp08a	3601	741	21	11	10.9
pp08aCUTS	3608	789	11	6	5.0
qiu	3610	156	5	3	1.8
rgn	3601	750	29	15	0.0
set1ch	3617	412	15	8	0.1
stein15	2460	1814	255	128	100.0
stein27	3602	2579	35	18	7.9
stein45	3602	1906	15	8	0.0
vpm1	3621	325	37	19	0.0
vpm2	3608	460	19	10	1.4
Average MILP-B		705.7	36.5	17.8	12.5
bell3a	3605	858	55	28	7.5
bell5	3604	971	87	44	0.8
blend2	3602	382	25	12	14.8
flugpl	3600	3856	178	71	29.2
gen	3604	502	5	3	0.0
gt2	3618	470	53	27	5.1
rout	3601	417	3	2	2.1
timtab1	3602	700	18	9	4.0
timtab2	3601	597	9	5	1.4
Average MILP-G		972.6	48.1	22.3	7.2

For the set of MILP-B instances, the average percentage gap closure is 12.5%, the average number of cuts generated is 705.7. The size of the cutting plane tree (N, T) is

moderate. 3 instances *mas74, mas76, stein15* obtain around 50% gap closure. On the other hand, 9 instances report no (0%) gap closure, including 3 known hard instances: *markshare1, markshare2* and *pk1*. For these 3 instances, Bonami and Minoux (2005), Balas and Saxena (2008), Balas and Bonami (2009) and Fischetti et al. (2009) also report 0% gap closure. *stein15* reports a 100% gap closure and an integer solution is found in 2640 seconds. *glass4, p0033* terminate before the time limit is reached due to numerical difficulties, we report results recorded before they terminate.

For the set of MILP-G instances, the average percentage gap closure is 7.2%, although the average number of cuts generated, 972.6, is higher than that of the MILP-B instances. The size of the cutting plane tree (N, T) is also moderate. No instances reports more than 30% gap closure, and one instance *gen* reports no (0%) gap closure.

CHAPTER 4

AN ITERATIVE METHOD FOR CUT GENERATION

In this chapter, we introduce an iterative algorithm, Multi-cut Row Generation (MRG) for solving the CGLP with multi-term disjunctions. In this method, CGLP is first transformed into a feasibility problem, it is then decomposed into a Benders' master problem and a set of Benders' subproblems (one for each disjunction of the multi-term disjunctions). MRG solves the CGLP by iteratively solving the Benders' master and subproblems. Due to the fact that at most one leaf node on the CPT tree may be expanded and hence most bounding constraints of the disjunctions remain the same when the algorithm goes from one iteration to the next, Benders' cuts generated in previous iterations may be reused in Benders' master problems in the following iterations. A "warm-starting" procedure that reuses Benders' cuts is also devised. Computational result shows that MRG is computationally more attractive than the one-cut-per-iteration Algorithm 2 in terms of integrality gap closure.

4.1 Introduction

Besides the fact that it solves the CGLP iteratively, the MRG algorithm also differs from the one-cut-per-iteration Algorithm 2 in three other major aspects: branching variable selection, normalization of the CGLP and expansion of the cutting plane tree.

- **Branching variable selection**: First, the one-cut-per-iteration algorithm generates only one disjunctive cut in each iteration. The MRG algorithm generates multiple disjunctive cuts in each iteration, one for each fractional variable in the so-

lution x^k of the linear program solved in each iteration. The set of cuts generated for x^k is referred to as a "round-of-cuts" (or "k-th round-of-cuts"). This approach has become standard for computational experiments with lift-and-project cuts (Balas et al., 1996; Bonami and Minoux, 2005) as well as Gomory cuts with multiple source rows. Compared with previous computational results using the one-cut-per-iteration approach, the computational results in this chapter show that, the multi-term disjunctions provided by the CPT algorithm proposed in this dissertation are extremely promising. The bottleneck however, is the solution of CGLP in the extended space. More research is necessary to develop methods to solve large-scale CGLPs arising from multi-term disjunctions effectively. The method proposed in (Perregaard and Balas, 2001; Balas and Bonami, 2009) to solve the CGLP in the original space of the variables works specifically for the case of simple variable disjunctions. In contrast to the CPT, which discovers the multi-term disjunction to be used, Perregaard and Balas (2001) investigate cut generation for a fixed number of terms in a multi-term disjunction. They provide computational results which suggest that Benders' decomposition algorithm is faster in generating a fixed number of cuts than solving a large-scale CGLP when there are more than two disjunctions in a CGLP. Nevertheless, the experiments reported in Perregaard and Balas (2001) were not intended to test the effectiveness of multi-term disjunctions in solving MILP-G instances. In this sense, our results are unique for pure cutting plane methods usingmulti-term disjunctions.

- **Normalization of the CGLP**: Second, in Section 3.6, we mentioned that the one-cut-per-iteration algorithm uses normalization constraint (3.7) in the CGLP to get bounded cut coefficients. In this chapter, we introduce the normalization used by the MRG algorithm: Minimum 1-norm Cut (M1NC). M1NC normalization leads

to a feasibility CGLP problem wherein every feasible solution gives a valid cut.

- **Expansion of the cutting plane tree**: As we have seen in Chapter 3, the cutting plane tree expands (branches, i.e., to create children nodes under a leaf node) when x^k falls in a leaf node. The MRG algorithm, however, adopts a more "conservative" branching rule. In iteration k of the MRG algorithm, the cutting plane tree is expanded only when $\exists j \in \{1, 2, ..., n_1\}$ such that $x_j^{k-1} \notin \mathbb{Z}, x_j^k \in \mathbb{Z}$. If this condition is satisfied, leaf node σ^k is branched at variable x_j (i.e., $v_{\sigma^k} = j$) with branching level $q_{\sigma^k} = \lfloor x_j^{k-1} \rfloor$. Furthermore, we can make one step down the tree, so that $\sigma^k \leftarrow l_{\sigma^k}$ or r_{σ^k}. This branching rule of the cutting plane tree is also used in all the algorithms that use "round-of-cuts" approach in the rest of this dissertation. Example 4.1.1 illustrates the expansion of a cutting plane tree when the MRG algorithm goes from iteration k to $k + 1$.

Example 4.1.1. (Expansion of the cutting plane tree for "round-of-cuts")
This example shows the "conservative" expansion of the cutting plane tree introduced above. In Figure 4.1, we consider the tree expansion from iteration k to iteration $k + 1$. At iteration $k + 1$, the algorithm visits leaf node $\sigma = 5$, suppose $\exists i \in \{1, 2, ..., n_1\}$ such that $x_i^k = 2.3$ and $x_i^{k+1} = 1$, then leaf node 5 is branched and nodes 6 and 7 are created, representing $x_i \leq 2$ and $x_i \geq 3$ respectively. The solution x^{k+1} can further move down the tree to the left child of node 5, hence we have $\sigma \leftarrow 6$ (after line 8 of Algorithm 3). At this point, the cutting plane tree $\mathcal{T}$ in iteration $k + 1$ is shown as the tree on the right in Figure 4.1.

"Virtual branching" is a special operation to the cutting plane tree $\mathcal{T}$ when the "round-of-cuts" approach is involved. In such an approach, each fractional variable is allowed a slightly different cutting plane tree, temporarily. These cutting plane trees only

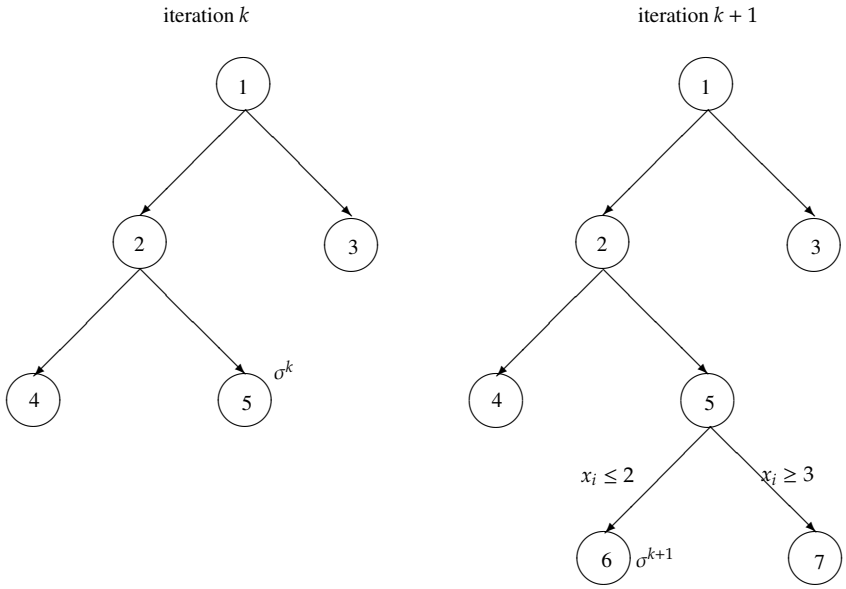

Figure 4.1: Expansion of the cutting plane tree for "round-of-cuts"

differ in the subtree rooted at the one node of $\mathcal{T}$. Let $\mathcal{T}^j$ denote the cutting plane tree used to create a disjunctive cut for fractional variable x_j. We need to consider two look-ahead scenarios for $\mathcal{T}^j$. At iteration $k+1$, the algorithm may visit

Scenario **a)** a leaf node and decide to expand the tree. After the expansion of the cutting plane tree, we get the tree $\mathcal{T}$ and a new leaf node σ. $\mathcal{T}^j$ is defined as $\mathcal{T}$ with its leaf node σ expanded with nodes l_σ and r_σ representing $x_j \leq v_\sigma$ and $x_j \geq v_\sigma + 1$ respectively.

Scenario **b)** a non-leaf node σ. Here, $\mathcal{T}$ remains the same compared to the previous iteration. $\mathcal{T}^j$ is defined as $\mathcal{T} \setminus \mathcal{S}(\sigma)$ (i.e., subtree $\mathcal{S}(\sigma)$ is removed from $\mathcal{T}$) with its leaf node σ expanded with nodes l_σ and r_σ representing $x_j \leq v_\sigma$ and $x_j \geq v_\sigma + 1$ respectively.

In either scenario, two cutting plane trees $\mathcal{T}^i$ and $\mathcal{T}^j$ for $i \neq j$ only differ in the subtree rooted at the node σ. Example 4.1.2 below depicts $\mathcal{T}^j$ in different scenarios.

Example 4.1.2. (Virtual branching)

For $\mathcal{T}^j$ in scenario a) and scenario b), please refer to Figure 4.2 and Figure 4.3 respec-

tively. In Figure 4.2, the tree above the dashed line on the right is $\mathcal{T}$. in Figure 4.3, the tree above the dashed line on the right is $\mathcal{T} \setminus \mathcal{S}(\sigma)$. In both cases, the whole tree on the right (both above and below the dashed line) is $\mathcal{T}^j$.

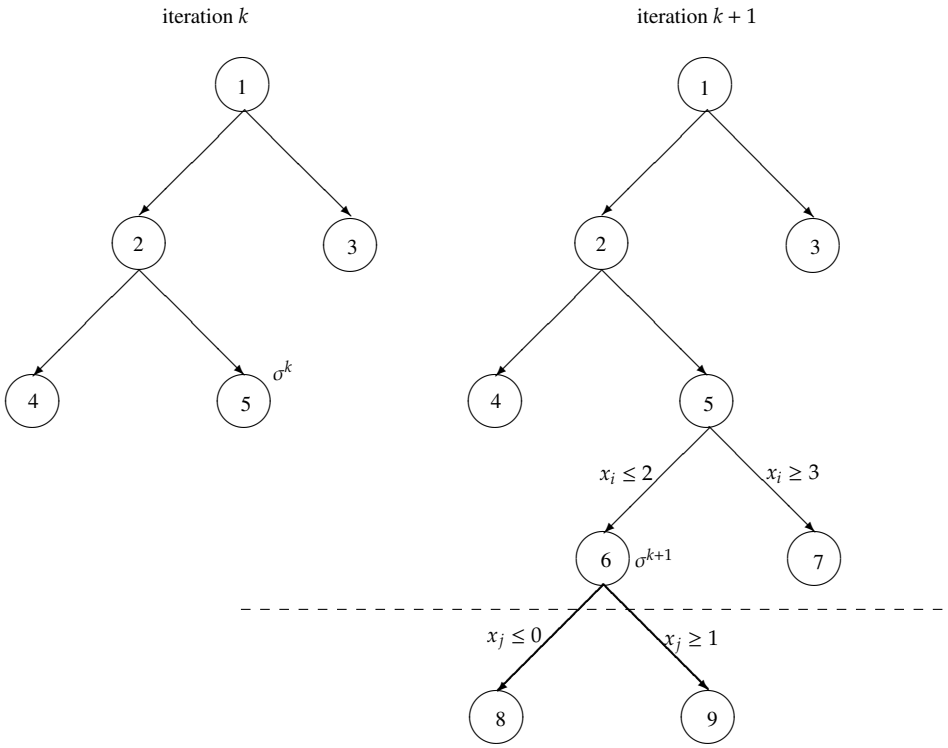

Figure 4.2: "Virtual branching" when the cutting plane tree is expanded

4.2 Multi-cut Row Generation Algorithm

At iteration k, to cut off a non-integral solution $\bar{x} = x^k$, MRG generates one disjunctive cut for each variable x_j that takes on a fractional value in $\bar{x}$. Let $\mathcal{L}^j_{k+1}$ denote the set of leaf nodes of the cutting plane tree $\mathcal{T}^j$ for variable x_j. Each leaf node $t \in \mathcal{L}^j_{k+1}$ corresponds to a region defined by $C^{k,j}_t = \{x | A^k x \ge b^k, L^{k,j}_t \le x \le U^{k,j}_t\}$, where A^k, b^k are cut-enhanced. We introduce a variable $y := x - \bar{x}$ and express the polyhedron in disjunction t as $\bar{C}^{k,j}_t = \{y | A^k y \ge \bar{b}^k, \bar{L}^{k,j}_t \le y \le \bar{U}^{k,j}_t\} = Q^{k,j}_t \cap X^k$, where $\bar{b}^k = b^k - A^k \bar{x}$, $\bar{L}^{k,j}_t = L^{k,j}_t - \bar{x}$ and

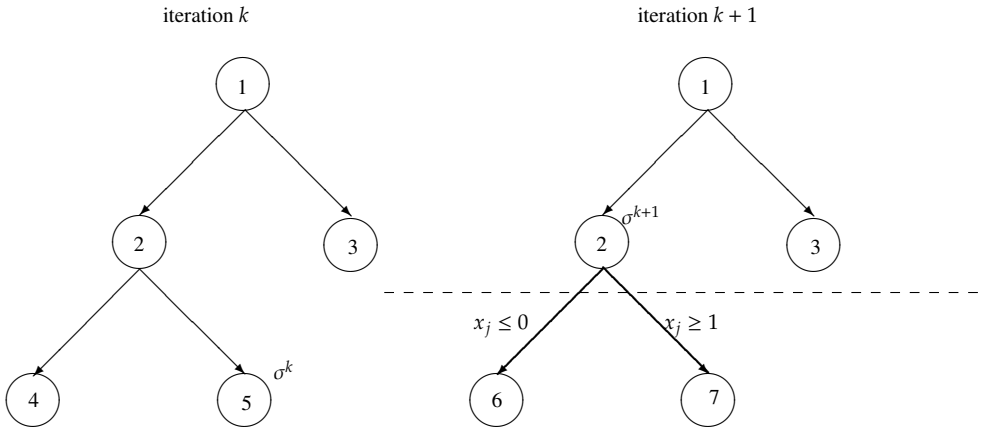

Figure 4.3: "Virtual branching" when the cutting plane tree is not expanded

$\bar{U}_t^{k,j} = U_t^{k,j} - \bar{x}$. Denote the multipliers used in the modified CGLP by $\{\lambda^t, \mu^t, \nu^t\}_{t=1}^T$, where λ^t corresponds to the multipliers associated with $A^k y \geq \bar{b}^k$ and μ^t, ν^t denote the multipliers for the lower and upper bound constraints, $\bar{L}_t^{k,j} \leq y \leq \bar{U}_t^{k,j}$, respectively. In the modified CGLP, we aim to now separate $\bar{y} = \mathbf{0}$, by generating a cut $\pi^T y \geq \pi_0$. Without loss of generality we let $\pi_0 = 1$. Note that the objective function of the CGLP becomes $\min \pi^T \bar{y} - \pi_0 = -1$. Therefore, the CGLP becomes a feasibility problem and the feasible region is defined by

$$\pi = {A^k}^T \lambda_t + \mu_t - \nu_t \qquad \forall t \in \mathcal{L}_{k+1}^j \tag{4.1}$$

$$\lambda_t^T \bar{b}^k + \mu_t^T \bar{L}_t^{k,j} - \nu_t^T \bar{U}_t^{k,j} \geq 1 \quad \forall t \in \mathcal{L}_{k+1}^j \tag{4.2}$$

$$(\lambda, \mu, \nu) \geq 0. \tag{4.3}$$

The cut that we obtain is $\pi^T y \geq 1$, or equivalently, $\pi^T x \geq 1 + \pi^T \bar{x}$.

Every feasible solution of (4.1)-(4.3) gives a valid cut. In particular, minimizing

$\sum_{j=1}^{n} |\pi_j|$ gives a valid cut, as stated in the CGLP problem below:

$$\min \quad \sum_{j=1}^{n} |\pi_j| \tag{4.4}$$

$$\text{s.t.} \quad \pi = A^{k^T} \lambda_t + \mu_t - \nu_t \qquad \forall t \in \mathcal{L}_{k+1}^{j} \tag{4.5}$$

$$\lambda_t^T \bar{b}^k + \mu_t^T \bar{L}_t^{k,j} - \nu_t^T \bar{U}_t^{k,j} \geq 1 \quad \forall t \in \mathcal{L}_{k+1}^{j} \tag{4.6}$$

$$(\lambda, \mu, \nu) \geq 0. \tag{4.7}$$

Such a CGLP is called Minimum 1-Norm Cut (M1NC) CGLP.

Next, problem (4.4) – (4.7) is decomposed into a Benders' master problem and a set of Benders' subproblems. The Benders' *Master Problem* is given by

$$\min \quad \sum_{j=1}^{n} |\pi_j| \tag{4.8}$$

$$\text{s.t.} \quad \pi^T y_t^i \geq z_t^i \quad , (y_t^i, z_t^i) \in \mathcal{R}_t^{k,j}, \forall t \in \mathcal{L}_{k+1}^{j}, \tag{4.9}$$

where $\mathcal{R}_t^{k,j}$ denotes the set of extreme rays (y_t^i, z_t^i) of

$$A^k y - \bar{b}^k z \geq 0 \tag{4.10}$$

$$y - \bar{L}_t^{k,j} z \geq 0 \tag{4.11}$$

$$-y + \bar{U}_t^{k,j} z \geq 0 \tag{4.12}$$

$$z \geq 0. \tag{4.13}$$

In the MRG algorithm, let k denote major iteration (or CPT iteration), let $x_j, j \in \mathcal{F}^k$ denote one fractional variable, where $\mathcal{F}^k$ denotes the set of integer variables in solution x^k that violates integrality constraints. For each (k, j) pair, one disjunctive cut is generated. Let s denote the Benders' iteration number.

At Benders' iteration s we only have a subset of the extreme ray sets $\mathcal{R}_t^{k,j,s} \subseteq \mathcal{R}_t^{k,j}$ for all t. We solve a restricted master problem:

$$\min \quad \sum_{j=1}^{n} |\pi_j| \tag{4.14}$$

$$\text{s.t.} \quad \pi^T y_t^i \geq z_t^i \quad , (y_t^i, z_t^i) \in R_t^{k,j,s}, \forall t \in \mathcal{L}_{k+1}^j \tag{4.15}$$

and let the optimal solution be $\bar{\pi}^{k,j,s}$. Then, for each $t \in \mathcal{L}_{k+1}^j$, we solve a Benders' subproblem

$$\min \qquad \bar{\pi}^{k,j,s^T} y - z \tag{4.16}$$

$$\text{s.t.} \quad (4.10) - (4.13), \tag{4.17}$$

which is dual to the feasibility subproblem t given by (4.1)–(4.3) for π fixed to $\bar{\pi}^{k,j,s}$. Note that if the dual subproblem is bounded (or equivalently if there is a feasible solution to (4.1)–(4.3) for π fixed to $\bar{\pi}^{k,j,s}$), its objective value is 0 with solution $(y_t, z_t) = 0$. In this case, we let $R_t^{k,j,s+1} = R_t^{k,j,s}$. However, if the dual subproblem is unbounded, then we add the a constraint (4.15) with a extreme ray solution (y_t, z_t) being the coefficients, to the Benders' master problem (4.14). In this case, $R_t^{k,j,s+1} = R_t^{k,j,s} \cup \{(y_t, z_t)\}$. The Benders' iteration stops if for all t the dual subproblems are bounded. The resulting $\bar{\pi}^{k,j,s}$ defines a valid inequality $\bar{\pi}^{k,j,s^T} y \geq 1$ violated by $\bar{y} = \mathbf{0}$.

The finite convergence of the MRG algorithm comes from the following arguments. Suppose when the Benders' subproblem (4.16) for a t is solved and found unbounded, the solution $(\hat{y}_t, \hat{z}_t)$ is an extreme ray solution. Since the number of extreme ray solutions for subproblem t is finite and there are finite number of subproblems, this Benders' method stops after a finite number of Benders' iterations. When the algorithm stops, denote the last solution from the Benders's master problem (4.14) by $\hat{\pi}$ and let the last solution for

t-th subproblem (4.16) be $(\hat{y}_t, \hat{z}_t)$. Note that all these subproblems have finite optimal solutions. The primal-dual relationship between the subproblem (4.16) and its dual (the feasibility subproblem t problem obtained from (4.5) – (4.7) when π is fixed to $\hat{\pi}$) makes $\hat{\pi}$ a feasible solution of the problem (4.4).

4.3 Warm-starting of the Algorithm

In this section, we introduce a technique of "warm-starting" of the MRG algorithm. Before that, we introduce some further notation. In order to describe the "round-of-cuts" approach, we also define the following notations:

$\mathcal{L}(t)$: the set of leaf nodes on subtree rooted at node t, on cutting plane tree $\mathcal{T}$.

$\mathcal{L}^j(t)$: the set of leaf nodes on subtree rooted at node t, on cutting plane tree $\mathcal{T}^j$.

ϕ_k: the number of disjunctive cuts generated until iteration k, hence the number of disjunctive cuts generated in iteration k is $\phi_{k+1} - \phi_k$.

For notations $\mathcal{T}$, o, σ, p_σ, l_σ, r_Σ, v_σ, q_σ, m_σ, $\mathcal{S}(\sigma)$, $\mathcal{N}(\sigma)$, $\mathcal{N}^-(\sigma)$, $\mathcal{N}^+(\sigma)$, $\mathcal{C}_\sigma$, $\mathcal{L}_k$, X_m, etc., please refer to Section 3.4.

As the MRG algorithm goes from one major iteration to the next, at most one leaf node of the CPT tree may be expanded, the bounding constraints for other leaf nodes remains unchanged. This suggests that the Benders' cuts generated in one major iteration may be reused in the Benders' master problems in the following major iterations. These reused cuts provide a "warm-starting" for the Benders' master problems.

In iteration k of the MRG algorithm, after tree expansion (line 11 of Algorithm 3), let the node visited be σ. Consider the cutting plane tree $\mathcal{T}^j$ for branched variable j after the "virtual branching" (line 15 of Algorithm 3), its set of leaf nodes $\mathcal{L}^j_{k+1}$ is divided into two sets: $\mathcal{G}^k$ and $\mathcal{E}^{k,j}$, defined by:

$$\mathcal{G}^k = \mathcal{L}_{k+1} \setminus (\mathcal{L}(\sigma) \cup \{\sigma\}) = \mathcal{L}^j_{k+1} \setminus (\mathcal{L}^j(\sigma) \cup \{\sigma\}) ,$$

$\mathcal{E}^{k,j} = \mathcal{L}^j(\sigma)$.

As before, the set operation $A \setminus B$ means "remove any element in B from A, if it is in A". We have

$\mathcal{L}^j_{k+1} = \mathcal{E}^{k,j} \cup \mathcal{G}^k$.

By definition, $\mathcal{E}^{k,j}$ contains the leaf nodes expanded under σ on tree $\mathcal{T}^j$, and the branching level depends on the value x^k_j of the branching variable x_j. $\mathcal{E}^{k,j}$ is therefore called "j-dependent". It is easy to see that the size of set $\mathcal{E}^{k,j}$ is at most 2. $\mathcal{G}^k$ contains the leaf nodes of $\mathcal{T}^j$ with the "j-dependent" set $\mathcal{E}^{k,j}$ removed, therefore is considered to be "j-independent". Definitions of these sets facilitate the algorithm design of the "warm-starting". Leaf nodes in $\mathcal{G}^k$ can independently determine which Benders' cut "can not" be reused in the Benders' master problems for each $j \in \mathcal{F}^k$ and should be removed. For the rest of the Benders' cuts, leaf nodes in $\mathcal{E}^{k,j}$ further identify which cut "can" be reused in the Benders' master problem for variable x_j.

We give examples to show such dichotomy of the leaf nodes on tree $\mathcal{T}^j$, in two possible scenarios: when the cutting plane $\mathcal{T}$ tree is expanded, and when it is not expanded. Figure 4.4 is for the former scenario. When the algorithm goes to iteration $k+1$, suppose it decides to expand the CPT tree $\mathcal{T}$ under leaf node 5. Then two nodes 6 and 7 are created (line 7 of ALgorithm 3), and the visited node σ^{k+1} further moves down to node 6. "Virtual branch" tree $\mathcal{T}^j$ is created for each fractional variable $j \in \mathcal{F}^{k+1}$. Then nodes 8 and 9 are created under $\sigma^{k+1} = 6$ on tree $\mathcal{T}^j$. Finally, by definition, $\mathcal{G}^{k+1} = \{4, 7, 3\}$ and $\mathcal{E}^{k+1,j} = \{8, 9\}$. Next consider the second scenario (i.e., the cutting plane tree is not expanded). Figure 4.5 depicts the tree for this scenario. When the algorithm goes to iteration $k+1$, it decides not to expand the CPT tree $\mathcal{T}$ under leaf node $\sigma^{k+1} = 2$. "Virtual branch" tree $\mathcal{T}^j$ is created for each fractional variable $j \in \mathcal{F}^{k+1}$. Then nodes 6 and 7 are created under $\sigma^{k+1} = 2$ on tree $\mathcal{T}^j$. Finally, by definition, $\mathcal{G}^{k+1} = \{3\}$ and $\mathcal{E}^{k+1,j} = \{6, 7\}$.

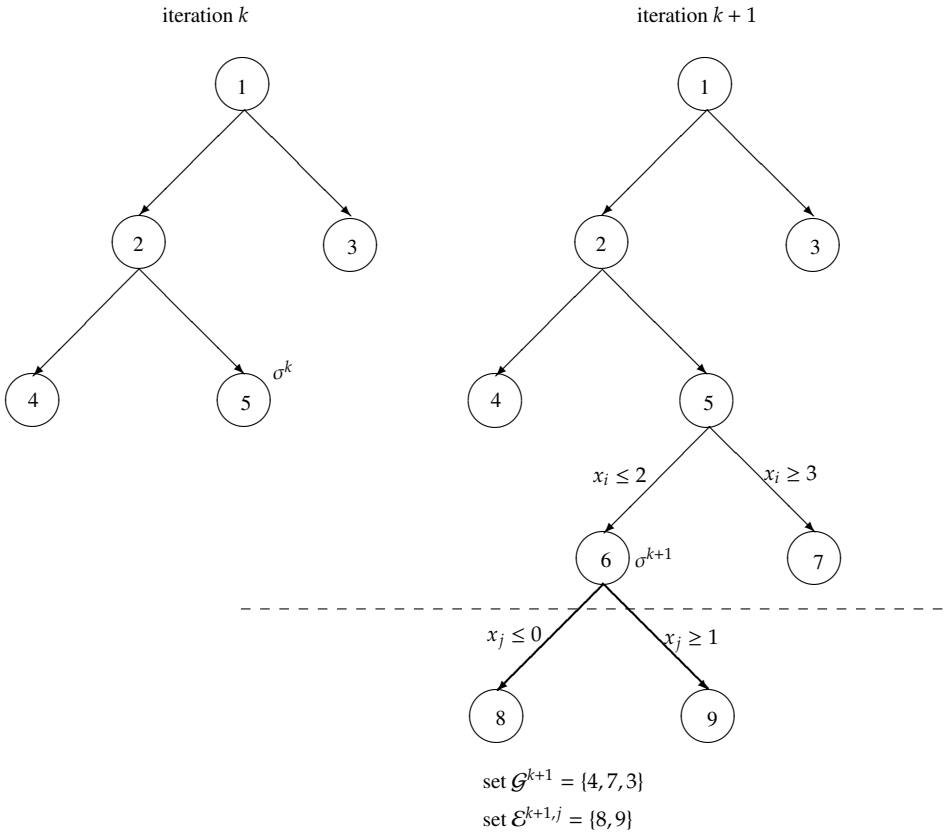

Figure 4.4: $\mathcal{G}^{k+1}$ and $\mathcal{E}^{k+1,j}$, CPT tree is expanded

The implementation of warm-starting, together with virtual branching will require an additional data-structure that we describe next. At iteration k, Let Π^{k-1} denote the set of disjunctive cuts $\left(\pi^i, \pi_0^i\right)$, $i \in \Pi^{k-1}$ that are generated in iteration $k-1$ and used in the current iteration to generate new disjunctive cuts.

$$\Pi^{k-1} = \begin{cases} \{\phi_{k-1}+1, ..., m_\sigma - 1\}, & \text{if } m_\sigma - 1 > \phi_{k-1}, \\ \emptyset, & \text{if } m_\sigma - 1 \le \phi_{k-1} \end{cases} \tag{4.18}$$

At any time in the algorithm, set $\mathcal{D}$ denotes the set of Benders' cuts available (not necessarily all cuts generated before, because some may be dropped in the algorithm). At iteration k, the algorithm makes the following two decisions.

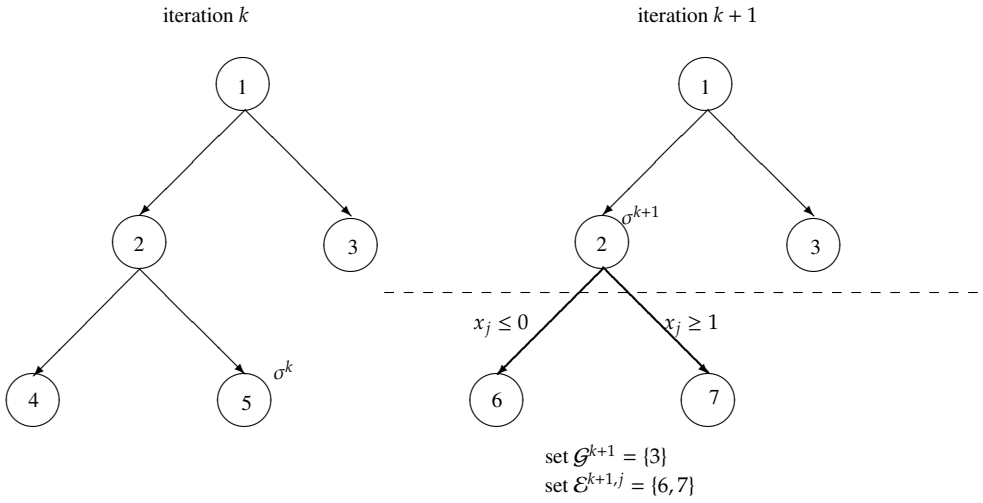

Figure 4.5: $\mathcal{G}^{k+1}$ and $\mathcal{E}^{k+1,j}$, CPT tree is not expanded

- (1) Which Benders' cut $(y^i, z^i), i \in D$ is removed from $\mathcal{D}$? This decision is made before solving any Benders' master problem.
- (2) Which Benders' cut can be reused? This decision is made before solving a Benders' master problem for each variable x_j.

A two dimensional table $\mathcal{M}^k$ is used to help make the decisions. Table $\mathcal{M}^k$ has $|\mathcal{D}|$ rows and $|\mathcal{G}^k \cup \mathcal{E}^{k,j}|$ columns, its element

$$\forall i \in \mathcal{D}, t \in \mathcal{G}^k \cup \mathcal{E}^{k,j}, \mathcal{M}^k_{i,t} = \begin{cases} 1 \quad , & \text{if } (y^i, z^i) \text{ is an unbounded} \\ & \text{(ray) solution of (4.10)-(4.13),} \\ 0 \quad , & \text{o/w} \end{cases}$$

Such a table $\mathcal{M}^k$ is illustrated in Table 4.1. In this matrix, an entry of 1 at row 2 and column 2 says that the Benders' cut (y^2, z^2) is an unbounded solution of the problem (4.10)-(4.13) for $t = 2$.

Although when Benders' cut (y^i, z^i) was generated in some previous iterations, it was an extreme ray solution of some unbounded Benders' subproblems in that iteration, it does

	$t \in \mathcal{G}^k$			$t \in \mathcal{E}^{k,j}$	
1	0	0	...	0	0
2	0	1	...	1	0
⋮	⋮	⋮	...	⋮	⋮
$i \in D$	1	0	...	0	0
⋮	⋮	⋮	...	⋮	⋮

Table 4.1: Table $\mathcal{M}^k$

not have to be an extreme solution of any of the Benders' subproblem in current iteration. However, as long as it is an unbounded (ray) solution of any subproblem defined for fractional variable x_j in this iteration, it can be used in "warm-starting" of the Benders' master problem for x_j.

Next we discuss how one decides whether previous Benders' cuts are to be dropped or not for the current CGLP.

- **Shrinking of $\mathcal{D}$**: Before the MRG algorithm generates disjunctive cuts for fractional variables (lines 13 – 26 of Algorithm 3), $\forall i \in \mathcal{D}$, if $\exists j \in \Pi^{k-1}$, such that $\pi^j(y^i/z^i) < 0$, then $\mathcal{D} \leftarrow \mathcal{D} \setminus \{i\}$, this step is also called "Shrinking of $\mathcal{D}$" and is performed in line 12 of Algorithm 3. Note that the ratio y^i/z^i is always valid because for (y^i, z^i) to be an unbounded (ray, including extreme ray) solution of a Benders' subproblem, $z^i > 0$.

- **j-independent Check**: After "Shrinking of $\mathcal{D}$", the MRG algorithm checks if any Benders' cut (y^i, z^i), $i \in \mathcal{D}$ is a unbounded (ray) solution of any subproblem for $t \in \mathcal{G}^k$. Since $\mathcal{G}^k$ is independent of branching variable x_j, this check is also called "j-independent Check" and can also be performed before the MRG algorithm generates disjunctive cuts for fractional variables (lines 13 – 26 of Algorithm 3). To this end, table $\mathcal{M}^k$ is initialized to a table of $|\mathcal{D}|$ rows and $|\mathcal{G}^k| + 2$ columns with all elements being 0s. The reason of having two additional columns (i.e. $|\mathcal{G}^k| + 2$) is

that at this point, branching variable x_j has not been chosen and therefore the actual size of set $\mathcal{E}^{k,j}$ is not known and consequently its upper bound is used. Specifically, for $i \in \mathcal{D}$, $t \in \mathcal{G}^k$, let $\mathcal{M}^k_{i,t} = 1$ if for all $l = 1, \ldots, n$, one of the following conditions holds:

$$y^i_l / z^i = 0, \bar{L}^{k,j}_{t,l} \geq -\infty \text{ and } \bar{U}^{k,j}_{t,l} \leq +\infty \tag{4.19}$$

$$y^i_l / z^i \geq 0, \bar{L}^{k,j}_{t,l} \geq -\infty \text{ and } \bar{U}^{k,j}_{t,l} = +\infty \tag{4.20}$$

$$y^i_l / z^i \leq 0, \bar{L}^{k,j}_{t,l} = -\infty \text{ and } \bar{U}^{k,j}_{t,l} \leq +\infty \tag{4.21}$$

The above "j-independent Check" is performed in line 12 of Algorithm 3.

- **j-dependent Check**: Lines 13 – 26 of Algorithm 3 generate one disjunctive cut for each fractional variable in $\mathcal{F}^k$. Once a variable x_j is chosen (line 13), the branching level and the "j-dependent" set $\mathcal{E}^{k,j}$ are determined, the corresponding Benders' subproblems (4.16) – (4.17) for $t \in \mathcal{E}^{k,j}$ are also determined. The MRG algorithm checks if any Benders' cut (y^i, z^i), $i \in \mathcal{D}$ is a unbounded (ray) solution of any subproblem $t \in \mathcal{E}^{k,j}$. Since $\mathcal{E}^{k,j}$ is dependent on branching variable x_j, this check is also called "j-dependent Check" and can only be performed after a fractional variable is selected. Specifically, for $i \in \mathcal{D}$, $t \in \mathcal{E}^{k,j}$, let $\mathcal{M}^k_{i,t} = 1$ if for all $l = 1, \ldots, n$, one of the following conditions holds:

$$y^i_l / z^i = 0, \bar{L}^{k,j}_{t,l} \geq -\infty \text{ and } \bar{U}^{k,j}_{t,l} \leq +\infty \tag{4.22}$$

$$y^i_l / z^i \geq 0, \bar{L}^{k,j}_{t,l} \geq -\infty \text{ and } \bar{U}^{k,j}_{t,l} = +\infty \tag{4.23}$$

$$y^i_l / z^i \leq 0, \bar{L}^{k,j}_{t,l} = -\infty \text{ and } \bar{U}^{k,j}_{t,l} \leq +\infty \tag{4.24}$$

The above "j-dependent Check" is performed in line 15 of Algorithm 3.

Note that in (4.19) – (4.21), for $\bar{L}_t^{k,j}, \bar{U}_t^{k,j}$, superscript j can be removed, since for $t \in \mathcal{G}^k$, these bounds are independent of any j. The same is not true for (4.22) – (4.24). The above three steps can be easily verified using the sufficient conditions for a vector d to be an unbounded (ray) of polyhedron $Ax \geq b$.

After the above steps, for $i \in \mathcal{D}$, if row i of table $\mathcal{M}^k$ contains at least one 1, Benders' cut (y^i, z^i) can be reused and is appended to the Benders' master problem for fractional variable x_j.

Algorithm 3 gives the Multi-cut Row Generation algorithm with warm-starting procedure. To simplify Algorithm 3, we define a Sub-algorithm 4 to perform tree expansion. Given a set X, a cutting plane tree $\mathcal{T}$, the set of leaf nodes $\mathcal{L}$ minus the node σ, a leaf node σ on $\mathcal{T}$, an index j for variable x_j which is fractional in $\bar{x}_j$, the Sub-algorithm 4 expands the tree $\mathcal{T}$ by creating two children nodes under node σ, with the split level $\lfloor \bar{x}_j \rfloor$, and enlarges the leaf nodes to include all leaf nodes on the new tree $\mathcal{T}$.

Lines 5 – 7 of the MRG Algorithm 3 uses the "conservative" rule of cutting plane tree expansion as described in Section 4.1. Line 7 calls the Sub-algorithm 4 to expand the cutting plane tree $\mathcal{T}$. $\mathcal{T}$ will be passed onto the next iteration after this iteration completes. The last operation in line 7 is to change the node σ visited in this iteration to either l_σ or r_σ.

In line 6, $m_\sigma = \phi_k + 1$. This is consistent to line 5 of Algorithm 2 because here ϕ_k is defined as the number of disjunctive cuts generated "before" iteration k, while in Algorithm 2, the number of disjunctive cuts generated before iteration k is indexed by $k - 1$. Similar argument applies to line 10 of Algorithm 3.

Algorithm 3 Multi-cut Row Generation algorithm

Initialization: $k = 1$, create a node o, where $p_o = null$, $m_o = 1$, $\phi_k = 0$. Let $\mathcal{T} = \{o\}$, $\mathcal{L}_k := \{o\}$ and $C_o = \{x | x_j \in [\ell_j, u_j], j = 1, \ldots, n_1\}$. Also let $X_k = X_L \cap C_o$ and $x^k \in \arg\min_{x \in X_k} c^T x$, where x^k is a vertex of X_k. Let $\mathcal{D} \leftarrow \emptyset$, $\mathcal{G}^k \leftarrow \emptyset$, $\mathcal{E}^{k,j} \leftarrow \emptyset, \forall j \in \{1, 2, ..., n_1\}$, $\Pi^{k-1} \leftarrow \emptyset$, $\mathcal{F}^k \leftarrow \emptyset$.

while $X_k \neq \emptyset$ and $x_1^k \notin \mathbb{Z}^{n_1}$ **do**

  Search for node $\sigma \in \mathcal{T}$ such that: either $\sigma \in \mathcal{L}_k$ and $x^k \in C_\sigma$; or, node $\sigma \notin \mathcal{L}_k$ and $x^k \in C_\sigma$, $x^k \notin C_{l_\sigma}$ and $x^k \notin C_{r_\sigma}$.

  if $\sigma \in \mathcal{L}_k$ **then**

    if $\exists j = 1, \ldots, n_1$ with $x_j^{k-1} \notin \mathbb{Z}$ and $x_j^k \in \mathbb{Z}$ **then**

      Let $v_\sigma = j$, $q_\sigma = \lfloor x_j^{k-1} \rfloor$, $m_\sigma = \phi_k + 1$, $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k \setminus \{\sigma\}$.

      Call function **branch**$(X_k, \mathcal{T}, \mathcal{L}_{k+1}, \sigma, j, x^{k-1})$. If $x_j^k \leq q_\sigma$, let $\sigma \leftarrow l_\sigma$, else, let $\sigma \leftarrow r_\sigma$.

    end if

  else if $\sigma \notin \mathcal{L}_k$ and $x^k \in C_\sigma$, $x^k \notin C_{l_\sigma}$ and $x^k \notin C_{r_\sigma}$ **then**

    Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k$, let $m_t \leftarrow \phi_k + 1, \forall t \in \mathcal{S}(\sigma)$.

  end if

  Update Π^{k-1} using (4.18). $\forall i \in \mathcal{D}$, if $\exists j \in \Pi^{k-1}$, such that $\pi^j(y^i/z^i) < 0$, then $\mathcal{D} \leftarrow \mathcal{D} \setminus \{i\}$ (Shrinking of $\mathcal{D}$). Let $\mathcal{G}^k \leftarrow \mathcal{L}_{k+1} \setminus (\mathcal{L}(\sigma) \cup \{\sigma\})$. Create table $\mathcal{M}^k$ with $|\mathcal{D}|$ rows and $|\mathcal{G}^k| + 2$ columns. $\forall i \in \mathcal{D}$, $\forall t = 1, \ldots, |\mathcal{G}^k|$, let $\mathcal{M}_{i,t}^k \leftarrow 1$ if for all $l = 1, \ldots, n$ one of (4.19) – (4.21) holds (j-independent Check), let $\mathcal{M}_{i,t}^k \leftarrow 0$ otherwise. $\forall i \in \mathcal{D}$, $t = |\mathcal{G}^k| + 1, |\mathcal{G}^k| + 2$, let $\mathcal{M}_{i,t}^k \leftarrow 0$. Let $\mathcal{F}^k \leftarrow \{i \in \{1, \ldots, n_1\} | x_i^k \notin \mathbb{Z}\}$, let $X_{k+1} \leftarrow X_k$.

  for all $j \in \mathcal{F}^k$ **do**

    Let $\mathcal{T}^j \leftarrow \mathcal{T} \setminus \mathcal{S}(\sigma)$, $\sigma^j \leftarrow \sigma$, $v_{\sigma^j} \leftarrow j$, $q_{\sigma^j} \leftarrow \lfloor x_j^k \rfloor$, $\mathcal{E}^{k,j} \leftarrow \mathcal{L}^j(\sigma^j)$, $\mathcal{L}_{k+1}^j \leftarrow \mathcal{L}_{k+1} \setminus \mathcal{L}(\sigma)$.

    Call function **branch**$(X_k, \mathcal{T}^j, \mathcal{L}_{k+1}^j, \sigma^j, j, x^k)$ (Virtual branching), after this call, $\mathcal{L}_{k+1}^j = \mathcal{G}^k \cup \mathcal{E}^{k,j}$. $\forall i \in \mathcal{D}$, $\forall t \in \mathcal{E}^{k,j}$, let $\mathcal{M}_{i,t}^k \leftarrow 1$ if for all $l = 1, \ldots, n$ one of (4.22) – (4.24) holds (j-dependent Check).

    Clean all rows of the Benders' master problem (4.14). Let Benders' iteration $s \leftarrow 1$. $\forall i \in \mathcal{D}$, if $\exists t \in \mathcal{G}^k \cup \mathcal{E}^{k,j}$, such that $\mathcal{M}_{i,t}^k = 1$, then add a row $\pi^T y^i \geq z^i$ to the Benders' master problem (4.14).

    while true do

      Solve Benders' master problem (4.14) for solution $\bar{\pi}^{k,j,s}$.

      for all $t = 1, \ldots, |\mathcal{G}^k| + |\mathcal{E}^{k,j}|$ **do**

        Solve t-th Benders' subproblem (4.16). If it is unbounded, let its extreme ray solution be $(\bar{y}_t, \bar{z}_t)$. Expand set $\mathcal{D}$ to include this Benders' cut $(\bar{y}_t, \bar{z}_t)$ (Expansion of $\mathcal{D}$). Add one row to $\mathcal{M}^k$ with all 0's but a 1 at $\mathcal{M}_{|\mathcal{D}|,t}^k$. Add a row $\pi^T \bar{y}_t \geq \bar{z}_t$ to Benders' master (4.14).

      end for

      If $\forall t = 1, \ldots, |\mathcal{G}^k| + |\mathcal{E}^{k,j}|$, subproblem t is bounded, go to step 24, otherwise, let $s \leftarrow s + 1$.

    end while

    Calculate disjunctive cut (π, π_0) where $\pi = \bar{\pi}^{k,j,s}$ and $\pi_0 = \pi^T x^k + 1$. Augment set X_{k+1} with this disjunctive cut.

    $\forall i \in \mathcal{D}$, $\forall t \in \mathcal{E}^{t,j}$, Let $\mathcal{M}_{i,t}^k \leftarrow 0$.

  end for

  Destroy (free) table $\mathcal{M}^k$. Let $k \leftarrow k + 1$. Let $x^k \in \arg\min_{x \in X_k} c^T x$, where x^k is a vertex of X_k.

end while

Algorithm 4 function: **branch** $(X, \mathcal{T}, \mathcal{L}, \sigma, j, \bar{x})$

Create a node l^- with $l_\sigma = l^-$, $p_{l^-} = \sigma$ and a node l^+ with $r_\sigma = l^+$, $p_{l^+} = \sigma$.
Let $C_{l^-} = \{x \in C_\sigma | x_j \leq \lfloor \bar{x}_j \rfloor\}$ and $C_{l^+} = \{x \in C_\sigma | x_j \geq \lceil \bar{x}_j \rceil\}$.
if $C_{l^-} \cap X \neq \emptyset$ **then**
    Let $\mathcal{L} \leftarrow \mathcal{L} \cup \{l^-\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^-\}$.
else
    Let $l^- = null$ (fathom).
end if
if $C_{l^+} \cap X \neq \emptyset$ **then**
    Let $\mathcal{L} \leftarrow \mathcal{L} \cup \{l^+\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^+\}$.
else
    Let $l^+ = null$ (fathom).
end if

4.4 Computational Results

The computational set-up in this section is the same as that for the one-cut-per-iteration algorithm in Section 3.7. In Table 4.2, we report our computational study of the MRG algorithm for the set of MILP-G and MILP-B instances described in Table 3.4. In order to compare with the one-cut-per-iteration algorithm, the computational results from Table 3.5 are also included in Table 4.2.

As before, integer values will be identified up to an accuracy of 1×10^{-8}. We also impose a limit on the number of Benders' iterations ($L = 1000$ denotes the number of Benders' master problems solved) for the generation of each disjunctive cut. Each Benders' master is first solved with simplex method, but if numerical difficulties are detected, barrier method is attempted. The time limit to solve any MILP instance (i.e. possibly several CGLPs) using MRG is 3600 seconds. For every instance, "time", "cuts", "N", "T" and "% gapcl" are reported, the same as we did in Section 3.7. The asterisk $*$ that appears in the "time" column for some instances indicates numerical difficulties that cause early termination of the runs. All computations are completed on a 3.2 GHz Sun workstation with 4GB RAM.

For the set of MILP-B instances, the average percentage gap closure is 22.6%, the

average number of cuts generated is 341.2. The size of the cutting plane tree (N, T) is moderate. Five instances obtain more than 50% gap closure. On the other hand, seven instances report no (0%) gap closure, including three known hard instances: *markshare1, markshare2* and *pk1*. Instances *fixnet6, mod008, p0033* terminate before the time limit is reached due to numerical difficulties, however, *p0033* achieves 78.9% gap closure before it stops. Also, for the MILP-B instances, the percentage gap closure of the MRG algorithm doubles that of the one-cut-per-iteration Algorithm 2, the average number of cuts generated by the MRG algorithm is less than half of that of the one-cut-per-iteration algorithm. In fact, for most of the instances (23 out of 31), the MRG algorithm generates less than half of the cuts that are generated by the one-cut-per-iteration algorithm.

For the set of MILP-G instances, the average percentage gap closure is 38.1%, which is a big leap compared to 7.2% of the one-cut-per-iteration Algorithm 2. The average number of cuts generated (278.3) and the size of the cutting plane tree (N, T) are small compared to the one-cut-per-iteration Algorithm 2. Except *rout*, all instances report more than 10% gap closure. The one-cut-per-iteration algorithm only wins the MRG algorithm in one instance: *flugpl*.

Overall, MRG achieves more than 15% gap closure over the one-cut-per-iteration algorithm on 16 instances. On the other hand, the one-cut-per-iteration algorithm achieves more than 15% gap closure over the MRG algorithm on three instances.

Our computational results show that compared to the one-cut-per-iteration CPT algorihtm, the MRG algorithm is a significant improvement on the given sets of MILP-B and MILP-G instances of moderate sizes.

Table 4.2: Computational results: one-cut-per-iteration vs. MRG

	one-cut-per-iteration					MRG				
problem	time	cuts	N	T	% gapcl	time	cuts	N	T	% gapcl
aflow30a	3613	191	9	5	5.5	4968	31	3	2	14.4
danoint	3638	254	3	2	0.3	3870	9	1	1	0.3
dcmulti	3618	507	5	3	9.8	3610	47	2	1	44.4
egout	3605	923	88	20	21.0	29	112	3	2	100.0
fixnet6	3601	655	5	3	4.5	165 *	60	3	2	10.4
glass4	1 *	3	5	3	0.0	3789	160	5	3	0.0
lseu	3605	1473	53	27	13.3	3630	165	5	3	32.7
markshare1	3601	514	87	44	0.0	3601	230	71	36	0.0
markshare2	3605	308	87	44	0.0	3602	163	47	24	0.0
mas74	3603	412	15	8	83.8	3780	1700	3	2	7.9
mas76	3607	529	11	6	49.3	3822	1823	13	7	8.2
misc03	3610	744	69	35	30.0	3601	1321	5	3	35.0
misc07	3617	308	29	15	4.3	3602	555	9	5	4.5
mod008	3603	209	29	15	13.6	517 *	2	1	1	0.1
modglob	3644	675	7	4	3.1	3768	31	3	2	25.8
opt1217	3813	33	31	16	0.0	3913	55	5	3	0.0
p0033	1687 *	2043	56	25	10.5	1378 *	596	11	6	78.9
p0201	3605	628	15	8	12.5	3676	92	3	2	7.1
p0282	3608	556	21	11	0.4	3619	51	5	3	62.1
p0548	3617	372	20	10	0.0	3900	50	3	2	61.3
pk1	3606	607	45	23	0.0	3601	769	19	10	0.0
pp08a	3601	741	21	11	10.9	3625	62	3	2	57.0
pp08aCUTS	3608	789	11	6.0	5	3728	31	1	1	16.3
qiu	3610	156	5	3	1.8	4095	0	1	1	0.0
rgn	3601	750	29	15	0.0	3606	205	5	3	20.0
set1ch	3617	412	15	8	0.1	5364	139	2	1	39.7
stein15	2460	1814	255	128	100.0	3604	1078	5	3	29.8
stein27	3602	2579	35	18	7.9	3607	584	5	3	7.2
stein45	3602	1906	15	8	0.0	3607	240	5	3	0.0
vpm1	3621	325	37	19	0.0	3960	141	5	3	22.1
vpm2	3608	460	19	10	1.4	4014	76	5	3	15.1
Average MILP-B		705.7	36.5	17.8	12.5		341.2	8.3	4.6	22.6
bell3a	3605	858	55	28	7.5	3628	120	19	10	70.7
bell5	3604	971	87	44	0.8	3721	90	5	2	87.2
blend2	3602	382	25	12	14.8	3666	158	5	3	23.0
flugpl	3600	3856	178	71	29.2	3602	1406	6	3	22.8
gen	3604	502	5	3	0.0	3695	34	1	1	45.4
gt2	3618	470	53	27	5.1	3949	48	7	4	55.9
rout	3601	417	3	2	2.1	3626	139	7	4	2.9
timtab1	3602	700	18	9	4.0	3680	267	5	3	21.9
timtab2	3601	597	9	5	1.4	4165	243	3	2	13.4
Average MILP-G		972.6	48.1	22.3	7.2		278.3	6.4	3.6	38.1

We also report some statistics for of the reuse of the Benders' cuts in Table 4.3.

- $\mathcal{I}$: the maximal number of Benders' cuts that have been reused in one Benders' master problem during the run.

- $||\mathcal{D}_r^*||$: Since the set of Benders' cuts may shrink (Algorithm 3, line 12) or expand (Algorithm 3, line 20), we collect statistics only for the set of Benders' cuts remained in set $\mathcal{D}^k$ when the runs stop. Let $\mathcal{D}^*$ denote the set of Benders' cuts that remain upon termination and let $\mathcal{D}_r^*$ denote the subset of Benders' cuts in $\mathcal{D}^*$ that are reused by some Benders' master problems. $|\mathcal{D}_r^*|$ records the size of the set $\mathcal{D}_r^*$. For each cut $i \in \mathcal{D}_r^*$, we assign a frequency number which records the number of times that this cut is reused (or equivalently, how many Benders' master problems reuse this cut), denoted as f_i. We collect the min, max and weighted average values of these frequency numbers.

- f_{min}: $\min_{i \in \mathcal{D}_r^*} f_i$.

- f_{max}: $\max_{i \in \mathcal{D}_r^*} f_i$.

- f_{avg}: $\sum_{i \in \mathcal{D}_r^*} f_i / |\mathcal{D}_r^*|$.

The statistics are given in Table 4.3. For about half of the instances, among the set of Benders' cuts that remain upon termination, more than 1000 cuts are reused by some Benders' master problems. For 9 instances, the weighted average frequency of being reused is above 10. For 11 instances including the three early-terminated instances *fixnet6, mod008, p0033*, no cuts are reused by any Benders' master problem.

Table 4.3: Statistics of MRG

problem	$\mathcal{I}$	$\|\mathcal{D}_r^*\|$	f_{mim}	f_{max}	f_{avg}
aflow30a	1391	1391	1	2	1.3
danoint	0	0			
dcmulti	0	0			
egout	215	328	1	19	1.9
fixnet6	*				
glass4	1975	15416	1	22	3.6
lseu	3081	12304	1	36	10.8
markshare1	2625	2082	1	140	22.8
markshare2	3810	2900	1	106	24.9
mas74	330	653	1	29	7.0
mas76	368	1309	1	173	34.3
misc03	34	108	1	7	1.9
misc07	74	351	1	13	4.7
mod008	*				
modglob	2211	4791	1	12	3.4
opt1217	2628	7589	1	15	3.9
p0033	*				
p0201	2194	4410	1	10	3.7
p0282	5644	12379	1	19	7.5
p0548	5750	10886	1	8	2.2
pk1	2149	3520	1	219	29.4
pp08a	2887	6515	1	11	2.9
pp08aCUTS	0	0			
qiu	0	0			
rgn	302	0			
set1ch	0	0			
stein15	228	6	1	12	5.8
stein27	875	0			
stein45	3170	8489	1	55	10.6
vpm1	2047	6491	1	17	3.8
vpm2	7264	14879	1	31	8.3
bell3a	2127	2512	1	43	14.1
bell5	2519	1521	1	11	3.1
blend2	1350	6301	1	8	2.2
flugpl	428	999	1	365	122.9
gen	0	0			
gt2	5615	15500	1	18	4.6
rout	219	3624	1	9	2.0
timtab1	4575	9467	1	93	27.1
timtab2	3411	7334	1	12	2.6

CHAPTER 5

NORMALIZATIONS CONSTRAINTS

Chapter 4 discusses the normalization constraint for the cut generation LP (CGLP) used by the MRG algorithm: Minimum 1-Norm Cut (M1NC). In this chapter, we first look into more details of the M1NC normalization constraint. Then we introduce another CGLP normalization constraint: Weighted Cut Coefficients (WCC). We show that WCC and M1NC normalizations, together with "round-of-cuts" approach, enhance the CPT algorithm greatly and are able to close a significant portion of the integrality gap without resorting to branch-and-cut. This chapter is based on a submitted paper by Chen et al. (2010).

5.1 Introduction

One important aspect of the cut generation LP is its normalization. The feasible set defined by (3.4) – (3.6) is a cone, and its extreme rays correspond to valid inequalities including all facets of the disjunctive program at hand. To truncate the feasible set and obtain a bounded solution, various normalization constraints are often introduced. Literature (Balas, 1979; Balas and Bonami, 2009; Balas and Perregaard, 2002; Cadoux, 2010; Cornuéjols and Lemaréchal, 2006; Fischetti et al., 2009; Rey and Sagastizábal, 2002) provides studies of various normalization constraints. The computational results presented in Balas et al. (1996), Fischetti et al. (2009), Balas and Bonami (2009) show that the choice of normalization has some impact on the performance and stability of the algorithm.

In this chapter, two normalization methods for the CGLP are studied. Algorithm 5 is obtained by modifying Algorithm 2 from generating one disjunctive cut per iteration to generating one disjunctive cut for each fractional variable in one iteration, the latter is referred to as "round-of-cuts" (we have touched this in Chapter 4). Section 5.3 discusses the WCC and M1NC normalizations for the CGLP, where the M1NC discussion in this section constitutes a complement to the derivation of M1NC in Section 4.2. Algorithm 5 hence is called WCC algorithm or M1NC algorithm, depending on the normalization scheme used. Section 5.4 discusses the implementation details of Algorithm 5 and the computational results of it.

5.2 Multi-cut Cutting Plane Tree Algorithm: Round-of-cuts

Some notations used in this chapter are defined in previous chapters: Chapter 3 and Chapter 4. For notations $\mathcal{T}$, o, σ, p_σ, l_σ, r_Σ, v_σ, q_σ, m_σ, $\mathcal{S}(\sigma)$, $\mathcal{N}(\sigma)$, $\mathcal{N}^-(\sigma)$, $\mathcal{N}^+(\sigma)$, C_σ, $\mathcal{L}_k$, X_m, please refer to Section 3.4. For notation $\mathcal{T}^j, j \in \mathcal{F}^k$, please refer to Section 4.1. For notations $\mathcal{L}^j_{k+1}, \mathcal{F}^k, \mathcal{L}(\sigma), \phi_k$, please refer to Section 4.3.

The pseudo-code of the cutting plane tree algorithm is given in Algorithm 5. At iteration k, if the current extreme point solution to $\min_{x \in X_k} c^T x$, given by x^k is integral, then we have found the optimal solution to the MILP-G. Otherwise, we search the cutting plane tree, to find the last node σ on the path from the root node such that $x^k \in C_\sigma$. There are two cases: Case (1) σ is a leaf node ($\sigma \in \mathcal{L}_k$), Case (2) σ is not a leaf node ($\sigma \notin \mathcal{L}_k$, $x^k \in C_\sigma$, $x^k \notin C_{l_\sigma}$ and $x^k \notin C_{r_\sigma}$). In Case (1), for each fractional variable x_j, let the split variable be $v_\sigma = j$. We create two new nodes: left (l_σ) and right (r_σ) children of σ at the split level $q_\sigma = \lfloor x_j \rfloor$. We let $C^j_{l_\sigma} = \{x \in C_\sigma | x_j \leq \lfloor x^k_j \rfloor\}$ and $C^j_{r_\sigma} = \{x \in C_\sigma | x_j \geq \lceil x^k_j \rceil\}$ and create an updated list of leaf nodes, $\mathcal{L}^j_{k+1}$ that includes $C^j_{l_\sigma}(C^j_{r_\sigma})$, unless $C^j_{l_\sigma} \cap X_k = \emptyset$ ($C^j_{r_\sigma} \cap X_k = \emptyset$). In this case, we also let $m_\sigma = \phi_k + 1$, as this is the first time the tree search

for a fractional solution stops at node σ. In Case (2), the cutting plane tree and m_σ are unchanged. However, in this case, we update $m_t = \phi_k + 1$ for all successors of σ, $t \in \mathcal{S}(\sigma)$. We generate a facet of the set $\text{clconv}\{\cup_{t \in \mathcal{L}^j_{k+1}} (X_{m_\sigma} \cap C^j_t)\}$ that cuts off x^k (using CGLP). We describe the CGLP in greater detail in Section 5.3. The new valid inequality given by this CGLP is included along with those defining X_k, and the resulting set is denoted X_{k+1}. This process continues until the stopping criterion is satisfied.

Algorithm 5 Cutting Plane Tree algorithm: WCC and M1NC

Initialization: $k = 1$, create a node o, where $p_o = null$, $m_o = 1$, $\phi_k = 0$. Let $\mathcal{T} = \{o\}$, $\mathcal{L}_k := \{o\}$ and $C_o = \{x | x_j \in [\ell_j, u_j], j = 1, \ldots, n_1\}$. Also let $X_k = X_L \cap C_o$ and $x^k \in \arg\min_{x \in X_k} c^T x$, where x^k is a vertex of X_k.
while $X_k \neq \emptyset$ and $x^k_1 \notin \mathbb{Z}^{n_1}$ **do**
  Search for node $\sigma \in \mathcal{T}$ such that: either $\sigma \in \mathcal{L}_k$ and $x^k \in C_\sigma$; or, node $\sigma \notin \mathcal{L}_k$ and $x^k \in C_\sigma$, $x^k \notin C_{l_\sigma}$ and $x^k \notin C_{r_\sigma}$.
  if $\sigma \in \mathcal{L}_k$ **then**
    if $\exists j = 1, \ldots, n_1$ with $x^{k-1}_j \notin \mathbb{Z}$ and $x^k_j \in \mathbb{Z}$ **then**
      Let $v_\sigma = j$, $q_\sigma = \lfloor x^{k-1}_j \rfloor$, $m_\sigma = \phi_k + 1$, $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k \setminus \{\sigma\}$.
      Call function **branch**$(X_k, \mathcal{T}, \mathcal{L}_{k+1}, \sigma, j, x^{k-1})$.
      if $x^k_j \leq q_\sigma$ **then**
        $\sigma \leftarrow l_\sigma$.
      else
        $\sigma \leftarrow r_\sigma$.
      end if
    end if
  else if $\sigma \notin \mathcal{L}_k$ and $x^k \in C_\sigma$, $x^k \notin C_{l_\sigma}$ and $x^k \notin C_{r_\sigma}$ **then**
    Let $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k$, let $m_t \leftarrow \phi_k + 1, \forall t \in \mathcal{S}(\sigma)$.
  end if
  for all $j = 1, \ldots, n_1$ such that $x^k_j \notin \mathbb{Z}$ **do**
    Let $\mathcal{T}^j \leftarrow \mathcal{T} \setminus \mathcal{S}(\sigma)$, $\sigma^j \leftarrow \sigma$, $v_{\sigma^j} \leftarrow j$, $q_{\sigma^j} \leftarrow \lfloor x^k_j \rfloor$, $\mathcal{L}^j_{k+1} \leftarrow \mathcal{L}_{k+1} \setminus \mathcal{L}(\sigma)$.
    Call function **branch**$(X_k, \mathcal{T}^j, \mathcal{L}^j_{k+1}, \sigma^j, j, x^k)$, after this call, $\mathcal{L}^j_{k+1}$ contains all leaf nodes of tree $\mathcal{T}^j$.
    Generate a valid inequality for the set $\text{clconv}\{\cup_{t \in \mathcal{L}^j_{k+1}} (X_{m_\sigma} \cap C_t)\}$ that cuts off x^k (using CGLP). Call the set with the additional valid inequality X_{k+1}.
  end for
  Let $k \leftarrow k + 1$. Let $x^k \in \arg\min_{x \in X_k} c^T x$, where x^k is a vertex of X_k.
end while

Algorithm 6 function: **branch** $(X, \mathcal{T}, \mathcal{L}, \sigma, j, \bar{x})$

Create a node l^- with $l_\sigma = l^-$, $p_{l^-} = \sigma$ and a node l^+ with $r_\sigma = l^+$, $p_{l^+} = \sigma$.
Let $C_{l^-} = \{x \in C_\sigma | x_j \leq \lfloor \bar{x}_j \rfloor\}$ and $C_{l^+} = \{x \in C_\sigma | x_j \geq \lceil \bar{x}_j \rceil\}$.
if $C_{l^-} \cap X \neq \emptyset$ **then**
    Let $\mathcal{L} \leftarrow \mathcal{L} \cup \{l^-\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^-\}$.
else
    Let $l^- = null$ (fathom).
end if
if $C_{l^+} \cap X \neq \emptyset$ **then**
    Let $\mathcal{L} \leftarrow \mathcal{L} \cup \{l^+\}$. Let $\mathcal{T} \leftarrow \mathcal{T} \cup \{l^+\}$.
else
    Let $l^+ = null$ (fathom).
end if

5.3 WCC and M1NC Normalizations

At iteration k for fractional variable x_k, $j = 1, \ldots, n_1$, define the sets $X_m \cap C_t = \{x | A_m x \geq b_m, L_t \leq x \leq U_t\}$ for $t \in \mathcal{L}^j_{k+1}$, where A_m, b_m are the cut-enhanced matrix and right-hand-side vector of the MILP-G, with $m = m_\sigma$. Let the multipliers used in the CGLP be $\{\lambda_t, \mu_t, \nu_t\}_{t \in \mathcal{L}^j_{k+1}}$, where λ_t corresponds to the vector of multipliers associated with $A_m x \geq b_m$, and μ_t, ν_t denote the vectors of multipliers for the lower and upper bound constraints, $L^k_t \leq x \leq U^k_t$, respectively. The CGLP has the form:

$$\bar{z} = \max \quad \pi_0 - \pi^T x^k \tag{5.1}$$

$$\text{s.t.} \quad \pi = \lambda_t^T A_m + \mu_t - \nu_t \qquad \forall t \in \mathcal{L}^j_{k+1} \tag{5.2}$$

$$\pi_0 \leq \lambda_t^T b^k_m + \mu_t^T L^k_t - \nu_t^T U^k_t \quad \forall t \in \mathcal{L}^j_{k+1} \tag{5.3}$$

$$(\lambda, \mu, \nu) \geq 0. \tag{5.4}$$

where $x^k \in C_\sigma$ is the current fractional solution. The optimal solution to the CGLP yields an inequality $\pi x \geq \pi_0$ valid for X that cuts off x^k.

5.3.1 Weighted Cut Coefficients (WCC)

This normalization constraint is a variant of that of Balas and Perregaard (2002), with $w^\top \pi = 1$, where $w = \hat{x} - x^k$, and $\hat{x}$ is some feasible solution to clconv$\{\cup_{t \in \mathcal{L}^j_{k+1}} (X_{m_\sigma} \cap C^j_t)\}$. In contrast to this choice of $\hat{x}$, Bonami and Minoux (2005) use a mixed-integer feasible solution to X. Since the latter would require our experiments to use either a pre-processing function, or a branch-and-cut algorithm, we modified the recommendation of Balas and Perregaard (2002) to one that fits more naturally in the CPT algorithm. In our implementation, we choose $\hat{x}$ to be an optimal extreme point of either $X_k \cap C^j_{l_\sigma}$ or $X_k \cap C^j_{r_\sigma}$, whichever gives a better bound. As shown in Balas and Perregaard (2002), including the constraint $w^\top \pi = 1$ to (5.1)–(5.4) renders the CGLP objective bounded.

5.3.2 Minimum 1-Norm Cut (M1NC)

Another way to ensure that the CGLP is bounded is to translate the coordinates to the point x^k, and then to find a minimum norm hyperplane that separates the origin from the closure of the convex hull of the disjunctive set. As a result, such cuts may be referred to as "deep" cuts (Cornuéjols and Lemaréchal, 2006; Cadoux, 2010; Rey and Sagastizábal, 2002). As one can recognize, this idea can be traced back to the very origins of optimization, i.e., separating hyperplane theorems. However, care must be taken to implement such ideas in a computationally effective way. It turns out that the choice of the norm used in defining the projection is critical to the success of a computer implementation of the CGLP. Computational experiments reported in Cadoux (2010) suggest that cuts which minimize the ℓ_2 norm do not lead to an effective computational approach. We have confirmed this conclusion in our own experiments. In this chapter, we use the M1NC normalization with ℓ_1 norm of π minimized. The derivation of the M1NC normalization can be found in Section 4.2. (5.5) – (5.8) gives the formulation of the CGLP using M1NC

normalization used in this chapter.

$$\min \quad \sum_{j=1}^{n} |\pi_j| \tag{5.5}$$

$$\text{s.t.} \quad \pi = \lambda_t^\top A_m + \mu_t - \nu_t \qquad t \in \mathcal{L}_{k+1}^j \tag{5.6}$$

$$\lambda_t^\top \bar{b}_m^k + \mu_t^\top \bar{L}_t^k - \nu_t^\top \bar{U}_t^k \geq 1 \quad t \in \mathcal{L}_{k+1}^j \tag{5.7}$$

$$(\lambda, \mu, \nu) \geq \mathbf{0} \tag{5.8}$$

where $\bar{b}_m^k = b_m - A_m x^k$, $\bar{L}_t^k = L_t - x^k$ and $\bar{U}_t^k = U_t - x^k$. The cut that we obtain is $\pi^\top(x - x^k) \geq 1$.

Below we construct the duality between the projection of the origin on to the convex hull (closure) of a disjunctive set using the ℓ_∞ norm and the M1NC (5.5) – (5.8).

For a fractional variable x_j^k, $j = 1, \ldots, n_1$, consider the CGLP (in the translated space) that maximizes the violation, subject to an upper bound constraint on the ℓ_1 norm of the cut coefficients:

$$\max \quad \pi_0 - \pi^\top \mathbf{0} \tag{5.9}$$

$$\text{s.t.} \quad \pi = \lambda_t^\top A_m + \mu_t - \nu_t \qquad t \in \mathcal{L}_{k+1}^j \tag{5.10}$$

$$\pi_0 - \lambda_t^\top \bar{b}_m^k - \mu_t^\top \bar{L}_t^k + \nu_t^\top \bar{U}_t^k \leq 0 \quad t \in \mathcal{L}_{k+1}^j \tag{5.11}$$

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{5.12}$$

$$\pi \leq \alpha \tag{5.13}$$

$$-\pi \leq \alpha \tag{5.14}$$

$$(\lambda, \mu, \nu, \alpha) \geq \mathbf{0}. \tag{5.15}$$

Because of the positive homogeneity of the the cut generation (set valued) mapping (Cornuéjols and Lemaréchal, 2006; Cadoux, 2010), this CGLP gives a cut $\pi'^\top(x - x^k) \geq$

π'_0, which is an optimum to M1NC up to a positive scalar multiple. Let y, z, η, y^+, y^- be the dual multipliers of the constraints (5.10)–(5.14) in that order. Then the dual problem is:

$$\begin{aligned}
\min \quad & \eta \\
\text{s.t.} \quad & \eta - y_i^+ - y_i^- \geq 0 \qquad i = 1, \ldots, n \\
& y^+ - y^- - \textstyle\sum_{t \in \mathcal{L}_{k+1}^j} y_t = 0 \\
& A_m y_t - \bar{b}_m^k z_t \geq 0 \\
& y_t - \bar{L}_t^k z_t \geq 0 \\
& -y_t + \bar{U}_t^k z_t \geq 0 \\
& \textstyle\sum_{t \in \mathcal{L}_{k+1}^j} z_t = 1 \\
& (y^+, y^-, z) \geq \mathbf{0}.
\end{aligned}$$

This is the problem of projecting the origin on to the disjunctive set under the ℓ_∞ norm, and hence the resulting cut is the "deepest" cut as measured by the ℓ_∞ norm.

5.4 Computational Results

In this section, we present the computational study of the "round-of-cuts" CPT Algorithm 5 with WCC/M1NC normalizations. Our results show that with the WCC/M1NC normalizations, the "round-of-cuts" CPT algorithm can close a significant portion of the integrality gap in many of the instances of the given set of MILPs.

We use the same set of test problems described in Table 3.4, as in the previous chapters. Integer values will be identified up to an accuracy of 1×10^{-8}. The time limit for the runs is 3600 seconds. In Table 5.1, we compared WCC and M1NC normalizations. For each instance using each of the two normalizations, we report "time", "cuts", "N",

"T" and "% gapcl", they are the same as in Section 3.7. The asterisk $*$ that appears in the "time" column for some instances indicates numerical difficulties that cause early termination of the runs. All computations are completed on a 3.2 GHz Sun workstation with 4GB RAM. Because the implementation delegates cut generation to the LP solver, we are not able to enforce the time limit very strictly during cut generation, the running time may actually exceed the time limit by several minutes. In table 5.1, we also include the computational results for the MRG algorithm from Table 4.2 in order to compare MRG with M1NC.

We can see that compared to the one-cut-per-iteration CPT Algorithm 2, the WCC algorithm and M1NC algorithm have big improvement in percentage gap closure for both MILP-B and MILP-G instances given in Table 3.4 within the same time limit.

Compared to the MRG Algorithm 3, M1NC algorithm still has obvious improvement. For MILP-B instances, the average percentage gap closure of the M1NC algorithm is 51.4, more than two times of that of the MRG algoritm. For MILP-G instances, the average percentage gap closure of the M1NC algorithm is 53.6%, bringing more than 15% (of the total gap) gap closure compared to 38.1% of the MRG algorithm. The average number of cuts generated by the M1NC algorithm for MILP-B instances is 1147.2, more than three times that of the MRG algorithm. The average number of cuts generated by the M1NC algorithm for MILP-G instances is more than four times ((1236 of M1NC vs. 278.3 of MRG)) that of the MRG algorithm. The size of the cutting plane tree (N, T) are similar for M1NC and MRG. MRG loses on all instances except on *mas76*, where MRG has 8.2% gap closure, while M1NC has 0% gap closure because numerical issues cause M1NC terminate after running for 6 seconds.

It is also quite remarkable that for several of these MILP instances, both WCC and M1NC algorithms are able to close more than 90% of the integrality gap with a pure

Table 5.1: Computational results: MRG vs. WCC vs. M1NC

	MRG					WCC					M1NC				
problem	time	cuts	N	T	% gapcl	time	cuts	N	T	% gapcl	time	cuts	N	T	% gapcl
aflow30a	4968	31	3	2	14.4	3601	360	3	2	31.9	3604	496	3	2	**49.8**
danoint	3870	9	1	1	0.3	3634	161	3	2	1.1	3703	129	7	4	1.7
dcmulti	3610	47	2	1	44.4	3618	514	3	2	95	3612	408	7	3	70.1
egout	29	112	3	2	100	27	163	3	2	100	21	129	5	3	100
fixnet6	165 *	60	3	2	10.4	5166	428	5	3	65.2	3636	508	5	3	**81.9**
glass4	3789	160	5	3	0	64 *	145	3	2	0	114 *	9	9	5	**25.0**
lseu	3630	165	5	3	32.7	3601	398	3	2	43.1	3604	1792	5	3	**59.9**
markshare1	3601	230	71	36	0	3620	405	19	10	0	3603	882	33	17	0
markshare2	3602	163	47	24	0	3609	353	17	9	0	3646	1215	27	14	0
mas74	3780	1700	3	2	7.9	27 *	10	3	2	11.9	11 *	1	1	1	9.9
mas76	3822	1823	13	7	8.2	25 *	10	3	2	10.5	6 *	0	1	1	0
misc03	3601	1321	5	3	35	3604	931	5	3	28.3	3600	3817	3	2	**50.3**
misc07	3602	555	9	5	4.5	3610	703	3	2	5.2	3600	2234	3	2	8.8
mod008	517 *	2	1	1	0.1	3601	252	5	3	20.8	3603	885	3	2	28.9
modglob	3768	31	3	2	25.8	3622	513	3	2	87.9	4635	409	3	2	96.9
opt1217	3913	55	5	3	0	3607	657	5	3	0.5	3612	528	15	8	0.5
p0033	1378 *	596	11	6	78.9	3601	475	7	4	75.3	3603	2616	9	5	84.3
p0201	3676	92	3	2	7.1	3607	575	3	2	76.6	3600	1034	3	2	65.4
p0282	3619	51	5	3	62.1	3611	780	3	2	96.7	3606	1398	3	2	97.9
p0548	3900	50	3	2	61.3	3627	551	3	2	45.1	3603	1264	3	2	**100.0**
pk1	3601	769	19	10	0	3604	525	11	6	0	3604	1605	7	4	0
pp08a	3625	62	3	2	57	3627	831	3	2	97.4	3605	1320	3	2	99.3
pp08aCUTS	3728	31	1	1	16.3	3602	706	3	2	90.4	3635	1015	5	3	97.8
qiu	4095	0	1	1	0	3636	103	1	1	42.7	3609	237	3	2	**63.5**
rgn	3606	205	5	3	20	3780	354	3	2	50.4	3601	1209	5	3	60.4
set1ch	5364	139	2	1	39.7	3600	1020	4	2	97.9	3608	1274	12	6	99.9
stein15	3604	1078	5	3	29.8	3603	2394	3	2	23.9	3601	2501	3	2	37
stein27	3607	584	5	3	7.2	3603	1893	3	2	11.3	3601	2098	3	2	20
stein45	3607	240	5	3	0	3606	1488	3	2	0	3601	1802	3	2	0
vpm1	3960	141	5	3	22.1	3604	951	3	2	84.2	3600	1628	3	2	**99.9**
vpm2	4014	76	5	3	15.1	3618	566	3	2	67.8	3603	1120	3	2	**85.3**
Average MILP-B		341.2	8.3	4.6	22.6		619.8	4.6	2.8	43.9		1147.2	6.4	3.6	51.4
bell3a	3628	120	19	10	70.7	15 *	58	9	5	70.7	3615	576	15	8	73.6
bell5	3721	90	5	2	87.2	3605	891	3	2	95.3	3621	1333	6	2	97.4
blend2	3666	158	5	3	23	3607	368	5	3	30.1	3602	1231	3	2	37.8
flugpl	3602	1406	6	3	22.8	3601	1144	5	3	23.3	3602	3369	8	4	25.9
gen	3695	34	1	1	45.4	3611	467	4	2	82.9	3600	844	3	2	88.2
gt2	3949	48	7	4	55.9	3709	194	11	6	92.9	3610	787	27	14	93
rout	3626	139	7	4	2.9	3601	319	3	2	1.2	3611	601	3	2	4.2
timtab1	3680	267	5	3	21.9	3611	818	3	2	37.8	3600	1173	5	3	36.6
timtab2	4165	243	3	2	13.4	3609	775	3	2	21.6	3601	1210	3	2	26
Average MILP-G		278.3	6.4	3.6	38.1		559.3	5.1	3	50.6		1236	8.1	4.3	53.6

cutting plane approach (especially for *bell5* and *gt2* which contain general integer variables). Overall, the average percentage gap improvements are 44.8% and 51.4%, for the WCC and M1NC normalizations, respectively. For over 20% of the test instances (highlighted in bold in the %*gapcl* column for M1NC), there is over 15% increase in the gap improvement using the M1NC normalization compared with the WCC normalization. In particular, for *p0548* the alternative normalization provides an additional 55% gap improvement within approximately the same running time. On the other hand, for instances *dcmulti, mas76* and *p0201*, the WCC normalization provides more than 10% gap improvement over the M1NC normalization. We also observe that *markshare1, markshare2* and *pk1* are hard instances for which (Bonami and Minoux, 2005; Balas and Saxena, 2008; Balas and Bonami, 2009) and Fischetti et al. (2009) also report 0% gap closure. If these instances are excluded, the average percentage gap improvements rise to 49.1% and 56.1%, for the WCC and M1NC normalizations, respectively.

Table 5.1 also demonstrates that for moderately sized instances (e.g., less than 1000 variables and less than 1000 constraints), the number of disjunctions explored in the CPT is far less than the worst case (full expansion of integer variables), and for such instances, ordinary LP solvers may suffice for cut generation. Moreover, the guidance provided by the objective function and intermediate fractional solutions help limit the number of leaf nodes explored by the CPT algorithm, and does not come close to its worst case bound.

CHAPTER 6

COMPARISIONS TO OTHER ALTERNATIVE ALGORITHMS

In this chapter, we introduce three other variants of the CPT algorithm. First, we consider disabling the expansion capability of the cutting plane tree, hence each disjunctive cut is generated from a CGLP using only two disjunctive terms, i.e., $x_j \leq v$ and $x_j \geq 1 + v$ (such disjunctions are traditionally called "simple disjunctions"). We called such variants of the CPT algorithm "Multi-cut Simple Disjunction (MSD)" algorithm. Depending on the normalization (WCC or M1NC) used, we have "MSD using WCC (MSD0)" or "MSD using M1NC (MSD1)".

In Section 5.3 we have used ℓ_1 norm of π as the objective function of the M1NC normalization for the CGLP, which generates the "deepest" cut in ℓ_∞ norm. In this chapter, instead of generating the "deepest" cut in ℓ_∞ norm, we use a $(\theta, 1-\theta)$ convex combination of two norms, the ℓ_1 norm and the ℓ_∞ norm of π, as the objective function of the CGLP. By perturbing the parameter θ, we can generate more than one cut for each fractional variable. Notice that the reason we can change the objective function of the CGLP and not lose the validity of the cuts is because when M1NC normalization is used, every feasible solution of the CGLP gives a valid inequality.

6.1 Multi-cut Simple Disjunctions Algorithm

The CPT algorithm without tree expansion is implemented by removing lines 3 – 16 in Algorithm 5 so that only the root node of the tree remains. The computational results for MSD0 (MSD using WCC normalization) and MSD1 (MSD using M1NC normalization)

are shown in Table 6.1.

6.2 Multi-cut for One Tree Algorithm

Since any feasible solution of (5.6) – (5.8) defines a valid disjunctive cut, we can use the following objective function, a convex combination of ℓ_1 norm and ℓ_∞ norm of π, parameterized by θ,

$$\min \theta \sum_{j=1}^{n} |\pi_j| + (1-\theta) \max_j |\pi_j|, \tag{6.1}$$

where $0 \leq \theta \leq 1$. By perturbing parameter θ, we can generate more than one cut for each fractional variable, per iteration. This method is named as "Multi-cut for One Tree (MFOT)", because more than one cut are generated for one unique cutting plane tree, which corresponds to one fractional variable.

We equally divide the interval $[0, 1]$ into P segments, and assign $P + 1$ values to θ, as the end values of each segment. In Table 6.1, we use $P = 5$, and hence $\theta = 0.0, 0.2, 0.4, 0.6, 0.8, 1.0$. In each iteration, for each fractional variable and each θ value, we solve one CGLP using M1NC normalization and objective function (6.1). The computational result is grouped under heading "MFOT-P5" in Table 6.1.

6.3 Computational Results

We use the same set of test problems described in Table 3.4, as in the previous chapters. In our implementation, integer values are identified up to an accuracy of 1×10^{-8}. The time limit for the runs remains 3600 seconds. In Table 6.1, we compare MSD0, MSD1 and M1NC normalizations. For each instance, we report "time", "cuts", "N", "T" and "%

gapcl" (see Section 3.7) for "MFOT-P5", and we report only "time", "cuts" and "% gapcl" for MSD0 and MSD1 because the cutting plane tree in MSD0 and MSD1 does not grow. The asterisk $*$ that appears in the "time" column for some instances indicates numerical difficulties that cause early termination of the runs. All computations are completed on a 3.2 GHz Sun workstation with 4GB RAM.

Table 6.1: Computational results: MSD0 vs. MSD1 vs. MFOT-P5

	MSD0			MSD1			MFOT-P5				
problem	time	cuts	% gapcl	time	cuts	% gapcl	time	cuts	N	T	% gapcl
aflow30a	3879	363	32.3	3607	533	53.7	3618	588	3	2	28.3
danoint	3634	195	1.8	3612	272	1.7	3893	342	3	2	1.7
dcmulti	3620	468	94.7	3601	788	97.4	3617	894	3	2	65.4
egout	33	172	100.0	9 *	83	100.0	40	546	3	2	100.0
fixnet6	3614	492	80.5	3606	738	90.6	3614	570	3	2	14.8
glass4	67 *	171	87.2	249 *	28	0.0	7 *	8	1	1	0.0
lseu	3605	1290	48.0	3610	1762	64.4	3607	3576	3	2	51.1
markshare1	3609	1347	0.0	3603	2509	0.0	3625	1998	19	10	0.0
markshare2	3601	1481	0.0	3601	2403	0.0	3625	1938	13	7	0.0
mas74	18 *	6	10.3	13 *	1	0.0	11 *	0	1	1	0.0
mas76	6 *	2	4.9	7 *	0	0.0	77 *	0	1	1	0.0
misc03	3602	889	33.9	3601	3527	47.3	3617	3276	3	2	25.9
misc07	3606	649	4.5	3601	2149	7.6	3609	2328	5	3	4.3
mod008	3607	902	25.2	3600	876	28.9	3636	1446	3	2	22.7
modglob	3636	344	75.6	3687	576	99.8	3608	746	3	2	79.5
opt1217	3609	684	0.6	3600	1457	0.5	3610	828	5	3	0.0
p0033	3605	1426	75.3	4549	2282	90.8	3605	5022	7	4	75.6
p0201	3613	589	77.3	3608	1006	62.0	3618	1590	3	2	30.9
p0282	3603	928	96.5	3606	1308	97.7	3625	2988	3	2	95.9
p0548	3618	573	39.5	3789	1140	99.2	3620	2466	3	2	97.5
pk1	3607	1044	0.0	3604	1912	0.0	3662	1866	9	5	0.0
pp08a	4790	717	96.8	3602	1225	99.2	3618	2184	3	2	94.7
pp08aCUTS	3601	574	95.0	3601	1151	98.0	3618	1248	3	2	65.6
qiu	3601	86	37.6	3626	279	69.2	3626	294	1	1	29.2
rgn	3602	514	30.6	3606	1450	53.7	3608	1926	5	3	27.8
set1ch	3605	1159	98.8	2317	1386	100.0	3635	1698	4	2	65.5
stein15	3601	2332	22.8	3601	2514	36.6	3607	7350	3	2	34.3
stein27	3604	1908	11.2	3602	2125	20.0	3604	4705	3	2	14.2
stein45	3604	1483	0.0	3604	1781	0.0	3613	3715	3	2	0.0
vpm1	3624	1008	84.6	843	962	100.0	3610	2925	3	2	82.1
vpm2	3603	699	74.9	3606	1133	86.3	3628	2045	3	2	69.2
Average MILP-B		790.2	46.5		1269.5	51.8		1971.2	4.1	2.5	37.9
bell3a	88 *	151	71.1	3631	1425	72.5	3612	648	19	10	72.4
bell5	3601	412	95.1	3675	1171	97.3	3606	1253	4	2	94.0
blend2	3618	534	34.4	3619	1283	39.2	3634	1654	3	2	29.4
flugpl	3600	802	23.1	3602	3846	27.1	3610	4782	5	3	24.5
gen	3613	518	84.9	3604	775	87.2	3616	912	3	2	62.7
gt2	3614	885	98.1	3617	1469	100.0	3608	1110	17	9	92.8
rout	3626	327	3.9	3602	819	5.3	3695	756	7	4	2.9
timtab1	3604	737	36.9	3603	1524	42.0	3605	1890	3	2	28.8
timtab2	3624	784	27.5	3604	1325	30.1	3615	2100	3	2	16.2
Average MILP-G		572.2	52.8		1515.2	55.6		1678.3	7.1	4.0	47.1

Compared to the WCC and M1NC algorithms, the MSD0 and MSD1 algorithms give

slightly higher average percentage gap closure (the difference is less than 3%). Compared to WCC, MSD0 generates approximately 200 more cuts on average, for both MILP-B and MILP-G instances. On the other hand, MSD1 generates approximately 400 more cuts on average than the M1NC.

For MSD0, the overall average percentage gap closure is 47.9%, whereas the average for MILP-Bs is 46.5% and for MILP-Gs is 52.8%. Although instance *glass4* terminated before time limit is reached, MSD0 still achieves 87.2% gap closure, much more than 0.0% gap closure of achieved by WCC. Three instances *dcmulti, fixnet6* and *glass4* achieve more than 15% gap closure over the WCC. *rgn* is the only instance on which WCC achieves more than 15% gap closure over the MSD0. Four instances stop before the time limit is reached due to numerical difficulties: *glass4, mas74, mas76* and *bell3a*.

For MSD1, the overall average percentage gap closure is 52.6%, whereas the average for MILP-Bs is 51.8% and for MILP-Gs is 55.6%. Although with slightly higher average percentage gap closure, MSD1 sees *dcmulti* as the only instance that achieves more than 15% gap closure over the M1NC algorithm. M1NC gives 25% gap closure on *glass4*, compared to the 0.0% given by the MSD1. Three instances for MSD1 stop before the time limit is reached: *glass4, mas74* and *mas76*. It is also worth mentioning that four instances *egout, set1ch, vpm1* and *gt2* achieve 100% gap closure, and two of them find an optimal integer solution within the time limit.

For MFOT-P5, the overall average percentage gap closure is 40.0% whereas the average for MILP-Bs is 37.9% and for MILP-Gs is 47.1%. Although the average number of cuts generated for MFOT-P5 are higher than that for MSD0 and MSD1, the average percentage gap closure of MFOT-P5 is lower (by more than about 8.5%) compared to MSD0 and MSD1. This can happen because redundant cuts are generated in MFOT-P5, as θ is perturbed.

For each instance, we compute a "baseline" percentage gap closure, which is the average of all the percentage gap closure values obtained by running all algorithms of one-cut-per-iteration, MRG, WCC, M1NC, MSD0, MSD1 and MFOT-P5, on this instance. Figures 6.1, 6.2 and 6.3 shows the difference of the percentage gap closure and its corresponding baseline percentage gap closure of all instances. These figures give the reader an overall picture of how each algorithm behaves relative to other algorithms, in terms of percentage gap closure. In these figures, the horizontal line represents the test instances in the same order as in the Table 3.4, the vertical line represents, for each instance, the difference between the percentage gap closure of one algorthm and its baseline percentage gap closure.

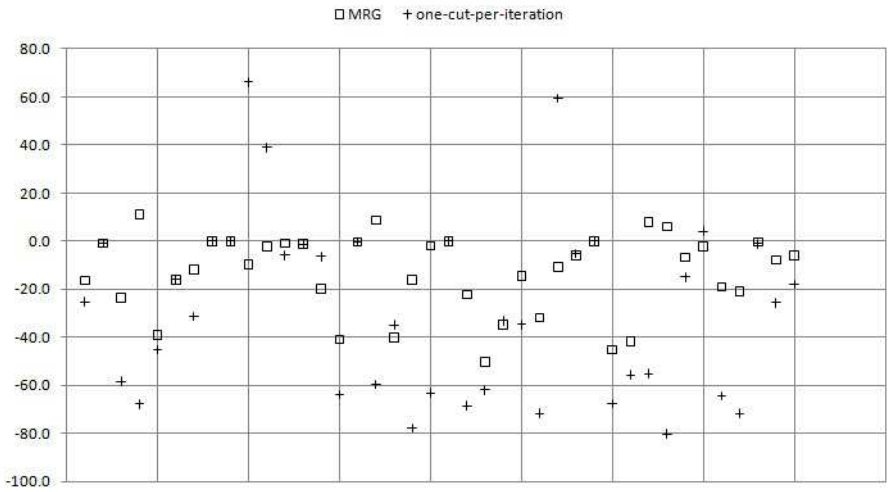

Figure 6.1: Difference between the percentage gap closure and its baseline: one-cut-per-iteration vs. MRG

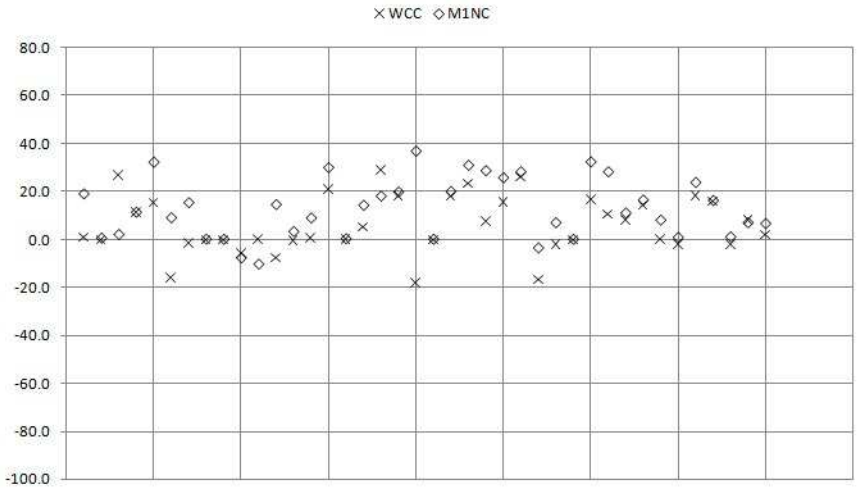

Figure 6.2: Difference between the percentage gap closure and its baseline: WCC vs. MRG

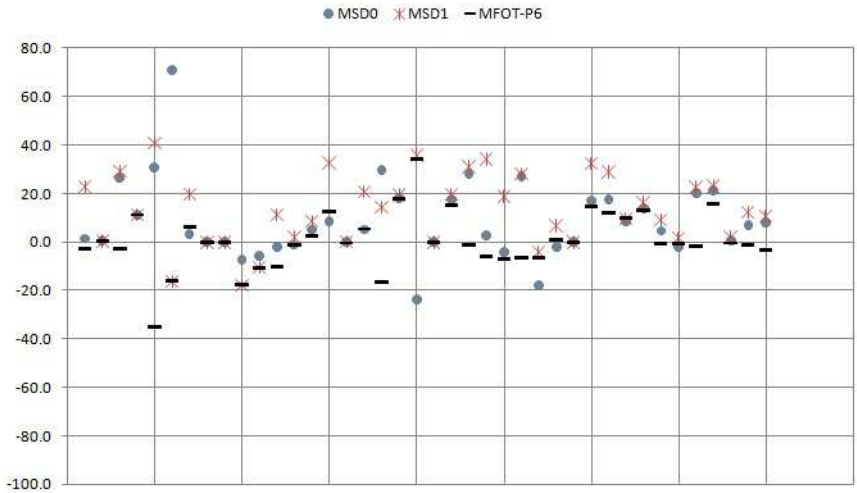

Figure 6.3: Difference between the percentage gap closure and its baseline: MSD0 vs. MSD1 vs. MFOT-P5

Table 6.2 divided the interval $[-100\%, 100\%]$ into 10 equal ranges, and shows the distribution of the number of instances whose percentage gap closure "wins" (or "losses" if negative) its baseline percentage gap closure in the 10 ranges, for each algorithm we have discussed. We can see that in the range (20% 0%], M1NC algorithm has the most

instances that "win" against the baseline, whereas in range (40% 20%], MSD1 algorithm has the most instances that "win" against the baseline. In all ranges that are above 40%, all algorithms have at most 1 instance that "wins" against the baseline.

Table 6.2: Distribution of the number of the percentage gap closure that "wins/losses" against its baseline

% range	one-cut-per -iteration	MRG	WCC	M1NC	MSD0	MSD1	MFOT-P5
[100% 80%]							
(80% 60%]	1				1		
(60% 40%]	1					1	
(40% 20%]	1		5	10	7	12	1
(20% 0%]	5	8	24	27	23	23	18
(0% -20%]	10	21	11	3	8	4	20
(-20% -40%]	6	6			1		1
(-40% -60%]	5	5					
(-60% -80%]	10						
(-80% -100%]	1						

CHAPTER 7

SUMMARY

Over the past five decades, there has been significant progress in the polyhedral study of MILP-B. For these problems the facial disjunctive property is well accepted to be critical for finite convergence of the sequential convexification process using disjunctive programming. However, such property does not hold for MILP-Gs. Although disjunctive cuts are natural to use for MILP-Gs, a cutting plane procedure using disjunctive cuts has not been proved to be finitely convergent. This dissertation proposes an algorithm which solves this problem, it also develops various techniques that greatly enhance the computational power the proposed algorithm.

In this dissertation, we propose a finitely convergent disjunctive cutting plane procedure, the Cutting Plane Tree (CPT) algorithm, to solve the MILP-Gs. This algorithm creates partitions of the feasible region that result in multiple disjunctions, as guided by the objective function. This dissertation has made algorithmic and computational contributions to the field of general mixed-integer programming as follows.

- The CPT algorithm is the first cutting plane algorithm that can solve MILP-Gs with bounded integer variables in finite steps.
- The work in this dissertation gives the first computational study of disjunctive cutting plane procedures using multi-term disjunctions without pre-specified hierarchy of the disjunctive structures. The multi-term disjunctions discovered in each iteration is based on the fractionality of the solution of the linear program solved in each iteration.

- We provide a wide-ranging investigation of computational tactics for implementing multi-term disjunctive cuts.

- We compared the impact of two normalization constraints for the CGLP to the computational performance of the CPT algorithm. We conclude that one of the normalization constraint seems to have an edge over the others.

As for future work, there is much more that should be investigated. First, it is natural to combine the branch-and-bound and the cutting plane tree. In such an algorithm, there will be two trees: one is from the CPT process, the other is from the B&B process. An identical mapping between the two trees is not computationally viable because the B&B tree grows very quickly, as a result, the large number of leaf nodes on the B&B tree (hence the CPT tree) makes the cut generation using the CGLP impractical. A key issue in this line of research is how to coordinate between the B&B tree and a CPT tree with moderate size.

Another potential area that one might pursue is the development of specialized simplex and barrier type methods to solve the multi-term CGLP. For two-term disjunctions, Balas and Perregaard (2003) have already reported exceptional progress by using the special structure. Perhaps, an extension of such work may be worth investigating.

Finally, we can also foresee the use of these tools in Stochastic Mixed-Integer Programming for which polyhedral characterizations could be used successfully, just as they have been used for Stochastic Combinatorial Optimization using ideas based on Disjunctive Decomposition (Yuan and Sen, 2009).

APPENDIX A

CUTTING PLANE TREES OF THREE EXAMPLES

Example 3.2.1 OM02. (cont.) Figure A.1 depicts the cutting plane tree at termination. The blank nodes (without a number) in Figure A.1 indicate nodes for which $X_k \cap C_\sigma = \emptyset$.

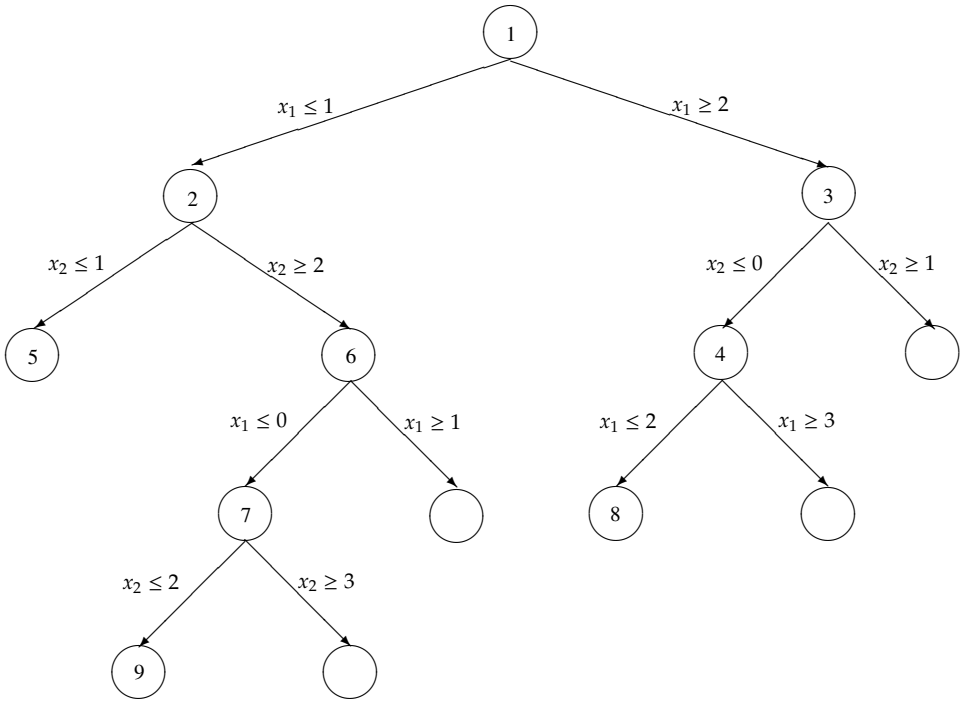

Figure A.1: OM01: The final cutting plane tree.

Example 3.5.1 CKS90. (cont.) Because it takes two iterations to solve this example, we illustrate the cutting plane tree at each iteration in Figure A.2.

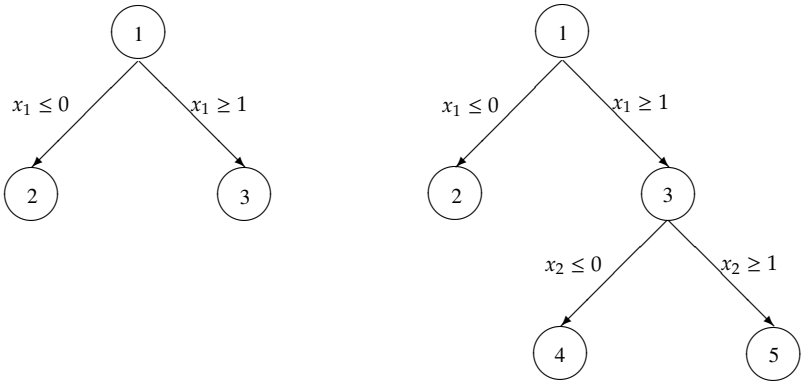

Figure A.2: CKS90: The cutting plane tree of iterations 1 and 2.

Example 3.5.2 SS85. (cont.) Figure A.3 depicts the cutting plane tree at termination.

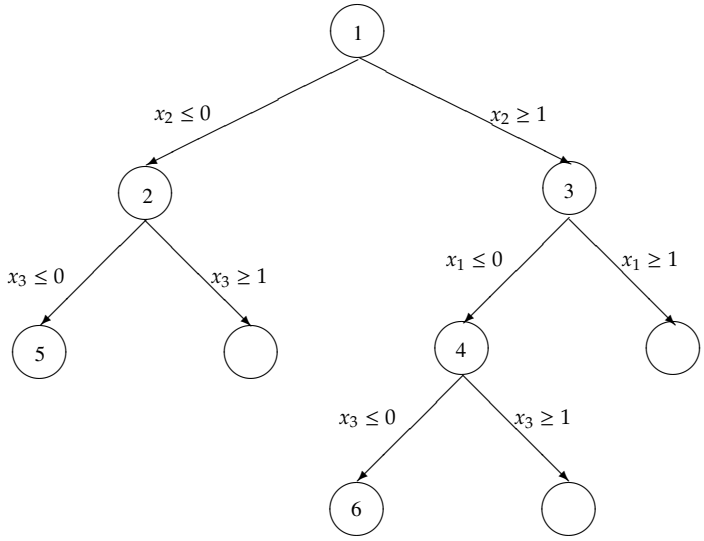

Figure A.3: SS85: The final cutting plane tree.

REFERENCES

Achterberg, T., T. Koch, and A. Martin (2006). MIPLIB 2003. *Operations Research Letters*, **34**(4), pp. 361–372. doi:10.1016/j.orl.2005.07.009.

Adams, W. P. and H. D. Sherali (2005). A hierarchy of relaxations leading to the convex hull representation for general discrete optimization problems. *Annals of Operations Research*, **140**(1), pp. 21–47.

Andersen, K., Q. Louveaux, R. Weismantel, and L. A. Wolsey (2007). Inequalities from Two Rows of a Simplex Tableau. In *IPCO '07: Proceedings of the 12th International Conference on Integer Programming and Combinatorial Optimization*, pp. 1–15. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-72791-0. doi:http://dx.doi.org/10.1007/978-3-540-72792-7_1.

Balas, E. (1971). Intersection Cuts - A New Type of Cutting Planes for Integer Programming. *Operations Research*, **19**, pp. 19–39.

Balas, E. (1979). Disjunctive Programming. *Annals of Discrete Mathematics*, **5**, pp. 3–51.

Balas, E. and P. Bonami (2009). Generating lift-and-project cuts from the LP simplex tableau: open source implementation and testing of new variants. *Mathematical Programming Computation*, **1**, pp. 165–199.

Balas, E., S. Ceria, and G. Cornuéjols (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, **58**(1–3), pp. 295–324.

Balas, E., S. Ceria, and G. Cornuéjols (1996). Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, **42**(9), pp. 1229–1246.

Balas, E. and M. Perregaard (2002). Lift-and-project for mixed 0-1 programming: recent progress. *Discrete Applied Mathematics*, **123**, pp. 129–154.

Balas, E. and M. Perregaard (2003). A precise correspondence between Lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. *Mathematical Programming*, **94**, pp. 221–245.

Balas, E. and A. Saxena (2008). Optimizing over the split closure. *Mathematical Programming*, **113**, pp. 219–240.

Bixby, R. E., E. A. Boyd, and R. R. Indovina (1992). MIPLIB: A test set of mixed integer programming problems. *SIAM News*, **25**, p. 16.

Bixby, R. E., S. Ceria, C. M. McZeal, and M. W. P. Savelsbergh (1996). An updated mixed integer linear programming library: MIPLIB 3.0. `http://www.caam.rice.edu/~bixby/miplib/miplib.html`.

Bonami, P., L. T. Biegler, A. R. Conn, G. Cornuéjols, I. E. Grossmann, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wachter (2008). An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization*, **5**(2), pp. 186–204.

Bonami, P. and M. Minoux (2005). Using rank-1 lift-and-project closures to generate cuts for 0-1 MIPs, a computational investigation. *Discrete Optimization*, **2**, pp. 288–307.

Cadoux, F. (2010). Computing deep facet-defining disjunctive cuts for mixed-integer programming. *Mathematical Programming*, **122**, pp. 197–223.

Caprara, A. and A. Letchford (2003). On the separation of split cuts and related inequalities. *Mathematical Programming*, **94**, pp. 279–294.

Chen, B., S. Küçükyavuz, and S. Sen (2010). A Computational Study of the Cutting Plane Tree Algorithm for General Mixed-Integer Linear Programs. Submitted to *Operations Research Letters*.

Chen, B., S. Küçükyavuz, and S. Sen (2011). Finite Disjunctive Programming Characterizations for General Mixed-Integer Linear Programs. *Operations Research*, **59**, pp. 202–210.

Cook, W., R. Kannan, and A. Schrijver (1990). Chvátal closures for mixed integer programming problems. *Mathematical Programming*, **47**(1–3), pp. 155–174.

Cornuéjols, G. (2008). Valid inequalities for mixed integer linear programs. *Mathematical Programming*, **112**(1), pp. 3–44.

Cornuéjols, G. and C. Lemaréchal (2006). A convex-analysis perspective on disjunctive cuts. *Mathematical Programming*, **106**, pp. 567–586.

Fischetti, M., A. Lodi, and A. Tramontani (2009). On the separation of disjunctive cuts. *Mathematical Programming*, pp. 1–26. ISSN 0025-5610.

Gomory, R. E. (1963). An algorithm for integer solutions to linear programs. In Graves, P. and P. Wolfe (eds.) *Recent Advances in Mathematical Programming*, pp. 269–302. McGraw-Hill, New York.

Jörg, M. (2007). K-disjunctive cuts and a finite cutting plane algorithm for general mixed integer linear programs. Retrieved June 19, 2009. `http://arxiv.org/PS_cache/arxiv/pdf/0707/0707.3945v1.pdf`.

Li, Y. and J.-P. Richard (2008). Cook, Kannan and Schrijver's example revisited. *Discrete Optimization*, **5**(4), pp. 724–734.

Lovász, L. and A. Schrijver (1991). Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal of Optimization*, **1**(2), pp. 166–190.

Owen, J. H. and S. Mehrotra (2001). A disjunctive cutting plane procedure for general mixed-integer linear programs. *Mathematical Programming*, **89**(3), pp. 437–448.

Owen, J. H. and S. Mehrotra (2002). On the value of Binary expansions for general mixed-integer programs. *Operations Research*, **50**(5), pp. 810–819.

Padberg, M. W., T. J. van Roy, and L. A. Wolsey (1985). Valid linear Inequalities for fixed charge problems. *Operations Research*, **33**(4), pp. 842–861.

Perregaard, M. and E. Balas (2001). Generating Cuts from Multiple-Term Disjunctions. In *Proceedings of the 8th International IPCO Conference on Integer Programming and Combinatorial Optimization*, pp. 348–360. Springer-Verlag, London, UK. ISBN 3-540-42225-0.

Rey, P. A. and Sagastizábal (2002). Convex normalizations in lift-and-project methods for 0-1 programming. *Annals of Operations Research*, **116**, pp. 91–112.

Sen, S. and H. D. Sherali (1985). On the convergence of cutting plane algorithms for a class of nonconvex mathematical programs. *Mathematical Programming*, **31**(1), pp. 42–56.

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

Sherali, H. D. and W. P. Adams (1994). A hierarchy of relaxations and convex hull representations for mixed-integer zero-one programming problems. *Discrete Applied Mathematics*, **52**(1), pp. 83–106.

Sherali, H. D. and C. M. Shetty (1980). *Optimization with Disjunctive Constraints*. Springer Verlag, Berlin.

Van Roy, T. J. and L. A. Wolsey (1985). Valid Inequalities and Separation for uncapacitated fixed charge networks. *Operations Research Letters*, **4**(3), pp. 105–112.

Yuan, Y. and S. Sen (2009). Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS Journal on Computing*, **21**(3), pp. 480–487.

Zanette, A., M. Fischetti, and E. Balas (2008). Can pure cutting plane algorithms work? In *Proceedings of IPCO 2008*, pp. 416–434. Springer-Verlag, London, UK.