ON SELECTING DISJUNCTIONS FOR SOLVING MIXED INTEGER PROGRAMMING PROBLEMS

by

Ashutosh Mahajan

Presented to the Graduate and Research Committee

of Lehigh University

in Candidacy for the Degree of

Doctor of Philosophy

in

Industrial and Systems Engineering

Lehigh University

May 2009

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Date

Dr. Theodore K. Ralphs
Dissertation Director

Accepted Date

Committee Members:

Dr. Theodore K. Ralphs, Chairman

Dr. Garth T. Isaak

Dr. Jeffrey T. Linderoth

Dr. Robert H. Storer

Acknowledgments

I would like to thank Dr. Ted Ralphs for guiding me through this thesis and also through the last five years of graduate studies. I also thank Dr. Jeff Linderoth for giving valuable advice on several topics in the thesis, for introducing several computational tools to me and for pulling me in to the world of Debian. I also want to thank other members of committee, Prof. Garth Isaak and Prof. Bob Storer, for their constructive feedback and for raising interesting questions from different topics of this thesis. I gratefully acknowledge COR@L Lab and HPC-Lehigh for providing software and hardware resources necessary to undertake the experiments described here. Finally I would like to thank the faculty, staff and fellow graduate students of the Industrial and Systems Engineering Department for making my graduate studies at Lehigh a truly wonderful experience.

Contents

List of Tables

List of Figures

Abstract

Mixed integer linear programs (MIPs) offer a flexible way to model a variety of real-world problems. In this thesis, we undertake a comprehensive study of "general disjunctions" that are a fundamental concept underlying most solution techniques for MIPs, with the aim of extending the current understanding of the theory of these disjunctions and also of developing novel ways of using them so as to enhance the performance of current state of the art MIP solvers.

Solving MIPs is difficult both theoretically (it lies in the class of so called $\mathcal{NP}$-hard problems) and in practice as well. Two algorithms that have been most successful in solving MIPs are the "branch-and-bound" and the "cutting-plane" algorithm. The concept of *valid disjunctions*, which are logical conditions that will be satisfied by any feasible solution is fundamental to both these algorithms. In branch-and-bound, one recursively imposes valid disjunctions on the problem to create smaller "subproblems" and solves the LP relaxations of these subproblems until a solution is found or the problem can be shown infeasible. The cutting-plane algorithm works on the principle that any inequality valid for each subproblem resulting from the imposition of a given disjunction is also valid for the original MIP. Such valid inequalities are iteratively added until an optimal solution is exposed or the feasible region can be shown to be empty. The performance of both algorithms thus greatly depends on which valid disjunctions are selected and imposed at each step.

In this thesis, we consider the problem of selecting disjunctions from a rich class that we call “general disjunctions”. Our work is divided into two parts: theory and computation. In the first part, we analyze the computational complexity of finding the “optimal” disjunction based on specific criteria. We show that the problem is in general $\mathcal{NP}$-hard, i.e., this problem is theoretically as difficult as solving the original MIP. We also show that the problem of selecting a general disjunction remains $\mathcal{NP}$-hard when several natural restrictions are imposed. These results lead to analogous results for the branch-and-bound and the cutting-plane algorithm: the problem of selecting a general disjunction that maximally improves the bound when used to branch and the problem of finding an “elementary split inequality” that maximally improves the bound are both $\mathcal{NP}$-hard. We also develop a polynomial-time algorithm that solves the separation problem for the elementary split closure when the point to be separated lies on an edge (i.e., a one dimensional face) of the feasible region of the LP relaxation.

In the second part, we perform several computational experiments to study the effect of employing methods for finding an “optimal” general disjunction on the performance of the branch-and-cut algorithm. We start by formulating the problem of finding an “optimal” general disjunction as a MIP. In a branch-and-bound setting, the solutions obtained from exact solution of these problems lead to significant reduction in the number of steps in the recursive procedure. When the set of allowed disjunctions is further restricted, one can use explicit enumeration techniques to select the optimal disjunction. We develop a procedure that eliminates poor choices of disjunctions in such a procedure by analyzing the solutions obtained from other disjunctions. We observe in our experiments that the use of this procedure reduces significantly the number of disjunctions that are evaluated for two particular types of general disjunctions.

We also use the disjunctions obtained from the techniques described above to generate valid inequalities. We observe that the number of inequalities needed to improve the bound is significantly fewer than that required by the existing techniques that generate the

“most-violated” inequalities. In all our experiments, the time required to select the disjunctions by attempting to solve the above-mentioned formulation using an exact method was prohibitive. This underscores the importance of developing fast computational methods for solving these problems. To motivate future work, we describe some open problems related to our work.

Chapter 1

Introduction

A mixed integer linear program (MILP) is a mathematical programming problem of the form:

$$\begin{aligned}
&\text{minimize } cx \\
&\text{subject to:} \\
&\qquad Ax \geq b \qquad\qquad (1.1)\\
&\qquad x \in \mathbb{Z}^d \times \mathbb{R}^{n-d},
\end{aligned}$$

where, $A \in \mathbb{Q}^{m \times n}$, $c \in \mathbb{Q}^n$, $b \in \mathbb{Q}^m$, $m, n, d(\leq n) \in \mathbb{Z}_+$ (see Section 1.1 for notation) are given as inputs. Since this thesis looks only at problems that are linear (both objective function and constraints are linear functions of the variable x), the term *linear* will be omitted henceforth and MILPs will be referred to as MIPs. Mixed integer programming offers a flexible framework within which one can model and analyze a variety of problems encountered in science, engineering and management, such as designing systems, planning operations, scheduling, logistics, location problems, modelling inventories, designing and controlling networks, etc. MIPs also arise in several theoretical problems in graph theory and statistics. Solution of a general mixed integer program is in general difficult and the problem is known to lie in the complexity class $\mathcal{NP}$-hard. This means that as their size increases, the difficulty of solving MIP instances using current state-of-the-art algorithms can increase exponentially fast.

The two most commonly-used algorithms for solving MIPs, called, the *branch-and-bound algorithm* and the *cutting-plane algorithm* implicitly enumerate the set of all values that can be assumed by the integer variables. Both these enumeration schemes derive from a divide-and-conquer paradigm and their performance is greatly influenced by the way in which the search space is iteratively partitioned.

In this thesis, we develop and analyze new techniques for partitioning the search space using logical disjunctions with the aim of enhancing the performance of the branch-and-bound and the cutting-plane algorithm. When $d = 0$, formulation (1.1) does not have any integer-constrained variables. Such a program is called a linear program (LP). If such a problem is formed by dropping the integer-restrictions for a given MIP, then it is called the *LP relaxation* of the MIP. An LP relaxation can be solved much more efficiently than the MIP from which it was derived and hence can be used for an approximate analysis of the partitions created in the enumerative search.

Branch-and-bound is an enumeration technique first proposed by Land and Doig [1960]. This algorithm generates a bound on the optimal value of the MIP by solving an LP relaxation of the given problem. If the optimal solution obtained from solving this relaxation is feasible for the given MIP, then it is also an optimal solution for the MIP. Otherwise, the feasible region of the MIP is partitioned by imposing a logical disjunction and the process is repeated recursively on each of the resulting smaller problems, which we refer to henceforth as *subproblems*. The partitioning procedure is traditionally called *branching*. Nemhauser and Wolsey [1988, pg. 357] show that the branch-and-bound algorithm is, under mild conditions on the inputs and branching scheme, sufficient to find an optimal solution of a given MIP or to show that there is none in a finite number of steps.

Even though the branch-and-bound algorithm is, under mild conditions, sufficient to solve MIPs, naive implementations do not perform well in practice. An alternative algorithm that can be used to solve MIPs is the cutting-plane algorithm which was first introduced by Gomory [1958]. In this algorithm, an LP relaxation of the given problem is first solved to obtain a lower bound on the optimal solution of the MIP. Then *valid inequalities* are iteratively added to tighten the relaxation. Valid inequalities are those linear constraints that are satisfied by all feasible points of the given MIP. These may not however be satisfied by all feasible points of the LP relaxation, in which case the inequalities are also called *cutting planes*. In the cutting-plane algorithm, valid inequalities are usually selected in such a way that the optimal solution of the LP relaxation is *cut off* from the feasible region of the given MIP. The resulting tighter LP can be re-solved to obtain a possibly tighter lower bound. Almost all methods for generating valid inequalities for general MIPs are derived from "disjunctions" like those used for branching in branch-and-bound. Thus, the choice of disjunction can be critical for the performance of these algorithms as well.

Most MIP solvers use a combination of branch-and-bound and cutting-plane algorithms. Such an algorithm is called a *branch-and-cut* algorithm. A more detailed description of this algorithm is provided in Section 1.3. The performance of branch-and-cut algorithms has improved substantially in the last decade due to both improvements in available computational resources, and the development of more effective heuristic techniques for tightening relaxations and guiding the search [Bixby et al., 2000]. Even though all known implementations of the algorithm take exponential time in the worst case, they perform much better than this on average. This is possible because of the development of powerful techniques applied during each stage of the algorithm: efficient branching schemes for making the search faster, procedures for generation of valid inequalities, generation of good upper bounds through *primal heuristics*, enumeration through heuristic search procedures as well as tightening of relaxations through preprocessing and probing techniques. Each of these techniques can require making a *good choice* among many possible alternatives. However, making this *good choice* can be time-consuming and may be as difficult as the original problem itself.

This thesis looks at the problem of making such a choice for two fundamental techniques used in branch-and-cut: branching and generating valid inequalities. Since both these techniques rely on the selection of an underlying disjunction, a considerable portion of this thesis is devoted to the study of disjunctions. In particular, we study the computational complexity of several decision problems related to selecting disjunctions and show that these lead to similar results for both branching and generation of valid inequalities. These problems are then formulated as mathematical programs and computational experiments are carried out to understand the effects of employing exact solution methods for these selection problems in a branch-and-cut framework. We also try to understand the trade-offs in approximating these formulations. Control mechanisms that can be used to control the level of approximation at which these techniques can be used, are developed and tested. The main goals of this research can be summarized as:

1. Study the theory of disjunctions as the underlying principle for both branching decisions and generating valid inequalities within a branch-and-cut framework.

2. Formulate and analyze the computational complexity of the problems for selecting disjunctions for both branching and generation of valid inequalities.

3. Perform computational studies of the effect on the performance of the branch-and-cut algorithm of solving such selection problems to optimality.

4. Develop new heuristics for solving problems related to selecting disjunctions.

In the next section, we begin by describing the notation used throughout this thesis. We explain the concept of a *valid disjunction* in Section 1.2. In Section 1.3, we describe the branch-and-bound and the branch-and-cut algorithm and also explain the concepts of branching and generating valid inequalities. We survey the existing literature on branching and generation of valid inequalities in Section 1.4. Section 1.5 contains the description of the set up used for our computational experiments. Finally, in Section 1.6, we provide an outline of the thesis and summarize our contributions.

1.1 Notation and Terminology

This section describes the notation used in rest of the thesis. Scalars and one-dimensional vectors are represented in lowercase italics, e.g., d, α, m, x. The context makes clear the distinction between a vector and a scalar. The set of natural numbers will be denoted by $\mathbb{N}$, that of real numbers by $\mathbb{R}$, that of integers by $\mathbb{Z}$ and that of rational numbers by $\mathbb{Q}$. The sets of real, rational and integer vectors of dimension $n \in \mathbb{N}$ are represented by $\mathbb{R}^n, \mathbb{Q}^n, \mathbb{Z}^n$ respectively. For a given set $\mathcal{S}$, $\mathcal{S}_+$ denotes the set: $\{x \in \mathcal{S} \mid x \geq 0\}$. The notation $\lceil t \rceil$ and $\lfloor t \rfloor$ refer respectively to the floor and ceiling of the scalar t.

Matrices are represented by uppercase letters: A, B, etc. The i^{th} element of an n-dimensional (for some $n \in \mathbb{N}$) vector x is represented as x_i where $i \in \{1, 2, \ldots, n\}$. Similarly, a_{ij} refers to the element in the i^{th} row and j^{th} column of a matrix A. Sets are represented by uppercase letters: $\mathcal{S}, \mathcal{T}$, etc. Elements of countable sets will usually be denoted by lowercase letters with subscripts: s_i, t_i, etc. If, however, the elements of the set are themselves vectors, we will denote the elements as a^i, s^i, etc.

Throughout the thesis, a MIP is assumed to be of the form (1.1), where $m \in \mathbb{Z}_+, n \in \mathbb{Z}_+, A \in \mathbb{Q}^{m \times n}, b \in \mathbb{Q}^m, c \in \mathbb{Q}^n, d \in \mathbb{Z}_+, d \leq n$ are given as inputs. Non-negativity and any other bounds on the variables, if present, are assumed to be included in the constraint set $Ax \geq b$. Index set $I = \{1, 2, \ldots, d\}$ will be used to denote the set of indices of variables that are constrained to take integer values only. These variables are sometimes referred to as *integer variables*. Similarly, variables that are allowed to take non-integer values are referred to as *continuous variables*. Index set $C = \{d + 1, d + 2, \ldots, n\}$ is used to denote the set of indices of these variables. When the variables are restricted to the set $\{0,1\}^d \times \mathbb{R}^{n-d}$, the instance is called a mixed binary program (MBP). When $n = d$, the resulting instance is called a *pure integer program* or just an *integer program* (IP). If for a given IP, all variables are restricted to the set $\{0,1\}^n$, then it is also called a binary program (BP).

Polyhedra and polyhedral sets are represented by uppercase calligraphic characters like $\mathcal{P}, \mathcal{L}$, etc. The convex hull of a set $\mathcal{P}$ is denoted by $conv(\mathcal{P})$. The set $\mathcal{S} = \{x \in \mathbb{Z}^d \times \mathbb{R}^{n-d} \mid Ax \geq b\}$ will be used to denote the set of points feasible to a given MIP (1.1). A *general disjunction* will be represented as (π, π_0) or, more descriptively, as $\pi x \leq \pi_0 \vee \pi x \geq \pi_0,\ x \in \mathbb{R}^n$ where $\pi \in \mathbb{Z}^d \times \{0\}^{n-d}, \pi_0 \in \mathbb{Z}$. These concepts are described in Section 1.3. A valid inequality for a MIP of the form (1.1) with associated feasible region $\mathcal{S}$ will be denoted by an ordered pair (α, β) where $\alpha \in \mathbb{Q}^n, \beta \in \mathbb{Q}$, meaning that the relation $\alpha\tilde{x} \geq \beta$ is satisfied for any $\tilde{x} \in \mathcal{S}$.

1.2 Disjunctions

Since both the branch-and-bound and the cutting-plane algorithm are fundamentally based on the concept of *valid disjunctions* for MIPs, we describe this in detail before describing the two algorithms. A disjunction in its most general form is a logical operator that results in true whenever one or more of its operands are true. It is one of the basic operators used for describing satisfiability problems and constraint logic programming problems. In the context of linear systems, we say that a disjunction is an operator on a countable set of systems of inequalities, much like those in formulation (1.1), that results in true if and only if at least one of the systems has a feasible solution. We write a disjunction over a given set $\mathcal{S}$ as

$$\bigvee_{h \in \mathcal{Q}} A^h x \geq b^h,\ x \in \mathcal{S} \tag{1.2}$$

where $A^h \in \mathcal{Q}^{m_h \times n}$, $b^h \in \mathcal{Q}^{m_h}$, $n \in \mathbb{N}$, $m_h \in \mathbb{N}$, $h \in \mathcal{Q}$ and $\mathcal{Q}$ is an index set, and say that the disjunction is true if and only if there exist $\tilde{x} \in \mathcal{S}, h \in \mathcal{Q}$ such that $A^h\tilde{x} \geq b^h$. The set of all points $x \in \mathcal{S}$ for which the disjunction (1.2) is true, i.e.,

$$\bigcup_{h \in \mathcal{Q}} \{x \in \mathbb{R}^n \mid A^h x \geq b^h\}. \tag{1.3}$$

is called a *disjunctive set*.

The set of feasible points of a MIP in form (1.1) can be viewed as a linear system of inequalities with a disjunction because the constraint $x \in \mathbb{Z}^d \times \mathbb{R}^{n-d}$ can be expressed as a disjunction. Using the fact that for a countable index set $\mathcal{Q}$

$$\left\{x \in \mathbb{R}^n \mid Ax \geq b, \bigvee_{h \in \mathcal{Q}} A^h x \geq b^h x\right\} = \left\{x \in \mathbb{R}^n \mid \bigvee_{h \in \mathcal{Q}} (Ax \geq b, A^h x \geq b^h x)\right\}, \tag{1.4}$$

where A^h, b^h are defined as above, we can conclude that the feasible region of a MIP can be expressed as a single disjunction (with a possibly infinite number of operators).

We call a set $\{x \in \mathcal{S} \mid A^h x \geq b^h\}$, $h \in \mathcal{Q}$ associated with a disjunction of the form (1.2) a subset of the disjunctive set (1.3). In theory, the feasible region of any MIP can be expressed in the form (1.3). However, the fact that $|\mathcal{Q}|$ in such an expression could be unbounded makes it intractable to use such an expression to solve the MIP directly. In order to mitigate the effects of a large value of $|\mathcal{Q}|$, both branch-and-bound and cutting-plane algorithms resort to using disjunctive sets, the union of whose subsets contains (but is not necessarily the same as) the feasible region of the given MIP. We call a disjunction whose associated disjunctive set satisfies the above condition a *valid disjunction*. Thus, a disjunction of the form (1.2) is said to be valid for a MIP of the form (1.1) if

$$\{x \in \mathbb{Z}^d \times \mathbb{R}^{n-d} \mid Ax \geq b\} \subseteq \bigcup_{h \in \mathcal{Q}} \{x \in \mathbb{R}^n \mid A^h x \geq b^h, Ax \geq b\}. \tag{1.5}$$

Balas [1998] provide an excellent investigation of several properties of the convex hull of the union of subsets obtained from valid disjunctions, when they are available, and also establish some important theoretical results for MBPs. For each subset $\{x \in \mathbb{R}^n \mid A^h \geq b, Ax \geq b\}, h \in \mathcal{Q}$, we associate the following *subproblem* P^h

$$\begin{aligned} &\text{minimize } cx \\ &\text{subject to:} \\ &\qquad Ax \geq b \\ &\qquad A^h x \geq b^h \\ &\qquad x \in \mathbb{Z}^d \times \mathbb{R}^{n-d}, \end{aligned} \tag{1.6}$$

which is obtained by adding the constraints obtained from the disjunction to the original MIP.

Selection of valid disjunctions whose imposition can lead to a fast solution for a given MIP is the main topic of research in this thesis. We limit our attention to disjunctions of the form

$$\pi x \leq \pi_0 \vee \pi x \geq \pi_0 + 1,\ x \in \mathbb{R}^n \tag{1.7}$$

where $\pi \in \mathbb{Z}^n, \pi_0 \in \mathbb{Z}, \pi_i = 0, i \in C$. This disjunction is always valid for any MIP of the form (1.1) because all elements of $\mathbb{Z}^d \times \mathbb{R}^{n-d}$ satisfy the disjunction. Such disjunctions will henceforth be represented by the notation (π, π_0). The subsets associated with such a disjunction are

$$\{x \in \mathbb{R}^n \mid Ax \geq b, \pi x \leq \pi_0\} \text{ and } \{x \in \mathbb{R}^n \mid Ax \geq b, \pi x \geq \pi_0 + 1\} \tag{1.8}$$

The reason for restricting our study to disjunctions of this form is two-fold. First, almost all existing methods for branch-and-bound and cutting-plane algorithms can be described (see Section 1.3) in terms of generation of such disjunctions. The preliminary studies of Andersen et al. [2007] and Cornuéjols and Margot [2009] are an exception, but even these consider only disjunctions defined by two constraints. Second, even the problem of selecting "optimal" disjunctions from the set of all disjunctions of the form (1.2) are, as shown in Chapter 2, theoretically difficult and very few practical methods are available to identify these. So using such optimal disjunctions is already a leap forward in the ability to solve MIPs.

When π is a unit vector or, in other words, when the disjunction is of the form

$$x_i \leq \pi_0 \vee x_i \geq \pi_0 + 1,\ x \in \mathbb{R}^n \tag{1.9}$$

for some $i = 1, 2, \ldots, d$ and $\pi_0 \in \mathbb{Z}$, we call it a *variable disjunction*. In order to distinguish variable disjunctions from other disjunctions of the form (π, π_0), we will refer to the latter as *general disjunctions*. While both branch-and-bound and cutting-plane algorithms use general disjunctions, they do so in different ways. We now describe these algorithms.

1.3 Algorithms

The branch-and-cut algorithm is essentially a hybrid between the branch-and-bound and the cutting-plane algorithm. Both the branch-and-bound and the cutting-plane algorithm use general disjunctions to enumerate implicitly the elements of the feasible set, but they use these disjunctions very differently. Therefore, we describe the two algorithms independently, starting with the branch-and-bound algorithm.

1.3.1 The Branch-and-Bound Algorithm

Let $\mathcal{P} = \{x \in \mathbb{R}^n \mid Ax \geq b\}$ be the feasible region of the LP relaxation of the MIP (1.1) and let $\mathcal{S}^0 = \mathcal{P} \cap (\mathbb{Z}^d \times \mathbb{R}^{n-d})$ be the set of feasible points of the MIP. Let z_{IP} denote the optimal value of the MIP. The branch-and-bound algorithm starts by solving the LP relaxation of the given MIP instance. The solution value of the LP relaxation, z_{LP}, provides a lower bound on z_{IP}. Any feasible solution of the MIP instance provides an upper bound on z_{IP}. Branch-and-bound iteratively increases the lower bound and decreases the upper bound until they are equal, at which point it terminates.

If the solution to the LP relaxation is also feasible to the MIP then it is an optimal solution to the MIP as well and we are done. Otherwise, let $\mathcal{R}^0 = \mathcal{P}$ and let $\mathcal{R}^i \subseteq \mathcal{R}^0, i = 1, \ldots, k$ denote k polyhedra such that $\cup_{i=1}^{k} \mathcal{R}^i$ is a valid disjunctive set for $\mathcal{S}^0$ with disjunctive subsets $\mathcal{R}^i, i = 1, 2, \ldots, k$. Let $\mathcal{S}^i = (\mathbb{Z}^d \times \mathbb{R}^{n-d}) \cap \mathcal{R}^i$ and let P^i denote the problem of minimizing cx for $x \in S^i$. Let z_{LP}^i denote the optimal value of LP solved on $\mathcal{R}^i$ using the given objective function. Then clearly $z_{LP}^0 \leq \min_{i=1}^{k} z_{LP}^i \leq z_{IP}$. The first of these inequalities may be strict if $\cup_{i=1}^{k} \mathcal{R}^i$ does not include the optimal solution to the LP relaxation of P^0. Hence, we may obtain stronger lower bounds by such a partitioning. When this procedure is recursively applied to each of $\mathcal{S}^i, i = 1, \ldots, k$, it is called a branch-and-bound algorithm. The steps of branch-and-bound algorithm are described in Figure 1.1. At each iteration of the algorithm, following important decisions need to be made:

1. What disjunction should one use for branching in step 3 of the algorithm?

2. Which problem should be chosen from L in step 2?

Most implementations of the algorithm limit the number of subsets k at each stage to two. In such a scheme, general disjunctions of the form (π, π_0) are a natural candidate for branching.

It is often convenient to represent the evolution a branch-and-bound algorithm graphically by a *branch-and-bound tree*. In such a representation, each subproblem P^i is associated with a node of the tree, that is connected to k nodes, each denoting a subproblem resulting from the imposition of a branching disjunction on P^i. Thus, each node also corresponds to an associated LP relaxation that is solved for that particular subproblem.

1. Initialization: Set $z_{LB} \to -\infty, z_{UB} \to \infty$. $k \to 0$. Set candidate list $L \to \{P^0\}$, where $P^0 = P$.

2. If $L = \phi$, then the solution x^0 that yielded z_{UB} is optimal. If no such x^0 exists, then declare P infeasible.

3. Select and delete a problem P^i from L. Solve the LP relaxation of P^i to obtain the optimal value z^i_{LP}. Let x^i_{LP} be the optimal solution.

4. If the LP is infeasible, or if $z^i_{LP} \geq z_{UB}$, then go to Step (2). If LP solution is feasible to P and $z^i_{LP} < z_{UB}$, then $z_{UB} \to z^i_{LP}$. Delete all such P^j from L that have $z^j_{LP} \geq z_{UB}$ and go to step (2). Otherwise go to step (5).

5. Branch: Create k subsets (by adding linear inequalities) of the selected MIP such that each feasible point of the selected P^i is in one of the subsets and each feasible point in any subset is a feasible point of the selected MIP. Add each of these k MIPs to L.

6. Update $z_{LB} = \min\{z^i_{LP} \mid P^i \in L\}$. If $z_{LB} \geq z_{UB}$, stop. Otherwise, go to step (2).

Figure 1.1: Branch-and-bound algorithm, as described by Nemhauser and Wolsey [1988]

Figure 1.3 provides an illustration of such a tree for the example problem (1.10) described later.

1.3.2 The Cutting-Plane Algorithm

The cutting-plane algorithm works in a different way. An LP relaxation of a given MIP of form (1.1) is first solved to get a lower bound z_{LP} on the optimal solution value z_{IP}. Let $\mathcal{R}^0 = \mathcal{P}$ be the feasible region of the LP relaxation and let $\mathcal{S}^0 = \mathcal{P} \cap (\mathbb{Z}^d \times \mathbb{R}^{n-d})$. If the optimal solution of the LP relaxation (say x^0_{LP}) is also feasible for the MIP, then it is an optimal solution of the MIP. Otherwise, one can introduce a valid inequality that is satisfied by all points of $\mathcal{S}^0$ but not x^0_{LP}. Such a valid inequality for $\mathcal{S}^0$ is traditionally called a *cutting plane*. If a valid disjunction of form (1.2) is available for a given MIP (1.1), then any inequality that is satisfied by each of the $|\mathcal{Q}|$ subsets associated with disjunction is also satisfied by convex hull $conv(\bigcup_{h\in\mathcal{Q}}\{x \in \mathbb{R}^n \mid Ax \geq b,\ A^h x \geq b^h\})$ and hence, also by $\mathcal{S}^0$. Thus, an inequality satisfied by all subsets created by a disjunction is also satisfied by $\mathcal{S}^0$. In Section 1.4.2, we will see that in fact, all inequalities that are valid for S^0 are also valid for some disjunction of S^0. Thus, the identification of valid disjunctions implicitly underlies the cutting-plane algorithm as well.

After augmenting the initial LP relaxation with the generated valid inequality, the modified problem can be quickly re-solved by using the dual simplex method [Bertsimas and Tsitsiklis, 1997, pg. 156]. The solution of the new LP, z^1_{LP} satisfies: $z^1_{LP} \geq z_{LP}$. This process of adding a cutting plane and re-solving can be continued until either the optimal solution obtained from the augmented LP is feasible to the MIP (in which case, it is an optimal solution to the MIP) or the LP become infeasible in which case the MIP is also infeasible.

At each iteration of the cutting-plane algorithm, following important decisions need to be made:

1. What disjunctions should be used to generate valid inequalities?

1. Let z_{LB} and z_{UB} be the lower and upper bounds on the optimal value z of the given MIP P. Set $z_{LB} \rightarrow -\infty, z_{UB} \rightarrow \infty$. $k \rightarrow 0$. Set candidate list $L \rightarrow \{P^0\}$, where $P^0 = P$.
2. If $L = \phi$, stop. $i \rightarrow i+1$. Choose an MIP problem P^i from the list L and remove it from L.
3. Solve the LP relaxation of P^i to obtain the optimal value z_{LP}. If LP is infeasible, or if $z_{LP} \geq z_{UB}$, go to step (2). If LP solution is feasible to the P and $z_{LP} < z_{UB}$, then $z_{UB} \rightarrow z_{LP}$ and go to step (2).
4. Optionally, add cutting-planes to the LP. If cutting-planes are added, then go to step (3). Otherwise go to step (5).
5. Branch: Create k subsets (by adding linear inequalities) of the selected MIP such that each feasible point of the selected P^i is in one of the subsets and each feasible point in any subset is a feasible point of the selected MIP. Add each of these k MIPs to L.
6. Update $z_{LB} = \min\{z^i_{LP} \mid P^i \in L\}$. If $z_{LB} \geq z_{UB}$, stop. Otherwise, go to step (2).

Figure 1.2: Branch-and-cut algorithm, as described by Nemhauser and Wolsey [1988]

2. Given a disjunction, which valid inequalities should be generated and how?

Unlike the branch-and-bound algorithm, there is only one LP under consideration and the aforementioned decisions can affect its size, difficulty and the number of iterations and hence the time required to solve the problem. Neither branch-and-bound nor cutting-plane algorithms have been found in practice to be effective on their own. As such, most solvers use an algorithm that combines both of them: the branch-and-cut algorithm. The steps of a generic branch-and-cut algorithm are outlined in Figure 1.2. Under mild conditions on the input data and the branching scheme employed, the branch-and-cut algorithm either terminates and provides an optimal solution to the problem or proves that it is infeasible [Nemhauser and Wolsey, 1988, pg. 357].

Before proceeding further, we show by means of an example how both branch-and-bound and the cutting-plane algorithm make use of disjunctions. Consider as an example, the following integer program:

$$\begin{aligned} \min -x_1 - 4x_2 & \\ 2x_1 - 2x_2 &\geq -1 \\ -x_1 - 2x_2 &\geq -2.5 \\ -x_1 + 4x_2 &\geq 0 \\ x_1, x_2 &\in \{0, 1\}. \end{aligned} \tag{1.10}$$

A branch-and-bound tree showing a solution procedure for this problem is shown in Figure 1.3. The root node of the tree corresponds to the LP relaxation of the problem. The optimal value of this relaxation is -4.5. This initial lower bound increases to -4.0 after branching on the disjunction $x_1 \leq 0 \vee x_1 \geq 1,\ x \in \mathbb{R}^2$. After branching, we get two smaller problems denoted by nodes 2 and 3 in the figure. Each of these subproblems are solved by branching on the disjunction $x_2 \leq 0 \vee x_2 \geq 1, x \in \ \mathbb{R}^2$. The branch-and-bound tree shown corresponds to solving seven LP relaxations, one for each subproblem, in order to solve the problem.

Figure 1.4 shows pictorially the feasible region of the LP relaxation and the valid inequalities that may be used to solve the problem. $x^0 = (0.5, 1)$ is the solution to the original LP relaxation with solution value -4.5. Using the same disjunction $x_1 \leq 0 \vee x_1 \geq 1,\ x \in \mathbb{R}^2$ for generating a valid inequality, we obtain $x_1 - 4x_2 \geq -2$ that is valid for both subsets created by the disjunction. The objective value increases to -4.25 after adding this inequality and $x^2 = (1, 0.75)$ is the optimal solution for the tighter LP relaxation. Now imposing the disjunction $x_2 \leq 0 \vee x_2 \geq 1,\ x \in \mathbb{R}^2$, we can similarly obtain the valid inequality $-x_2 \geq 0$. Solving the LP relaxation after adding this inequality gives the required optimal solution $x^2 = (0, 0)$. Thus, we obtain the solution using the same

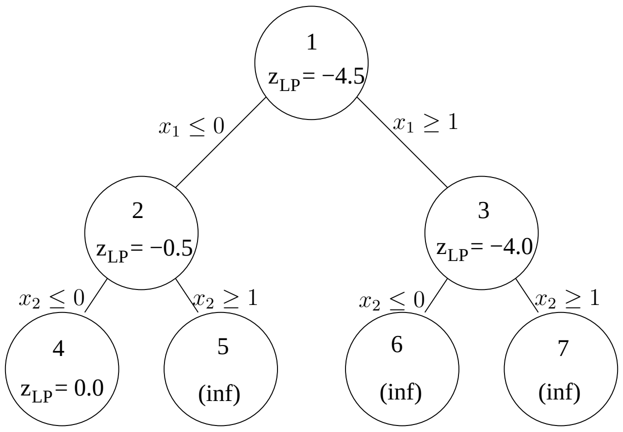


Figure 1.3: A branch-and-bound tree depicting a solution procedure for solving problem (1.10).

disjunctions, albeit in different ways.

1.4 Techniques for Branching and Generating Valid Inequalities

Following the previous discussion on the important role disjunctions play in both branch-and-bound and the cutting-plane algorithm, we briefly explain the problems related to identifying such disjunctions and using them in either algorithm. We review the fact that existing methods for both algorithms can be seen as special cases of using general disjunctions and then highlight the relationship between branching and generating valid inequalities.

1.4.1 Branching Methods

Suppose one is given a MIP instance of form (1.1) with set $\mathcal{S}^0$ of feasible points. When branching, one seeks to create k MIP instances $P^1, P^2, \ldots, P^k$ with the same objective function as that of (1.1) and with sets of feasible solutions $\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^k$ such that $\bigcup_1^k S^i =$

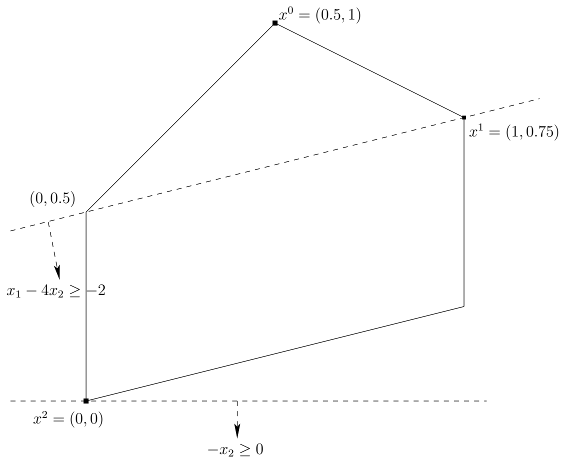


Figure 1.4: A cutting-plane procedure for solving solving problem (1.10).

$\mathcal{S}^0$. Even though any such branching rule may be used without affecting the correctness of the algorithm, the choice of branching rule can nonetheless severely affect its performance [Jeroslow, 1974]. It is trivial to construct branching rules such that the branch-and-bound algorithm never terminates. Practically, most implementations branch in such a way that $\mathcal{S}^1, \mathcal{S}^2, \ldots, \mathcal{S}^k$ are pairwise disjoint. The most common method of branching in such a way is to *branch on a hyperplane* by choosing a general disjunction $(\pi, \pi_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}$, creating two MIP instances P^1, P^2 where P^1 is constructed by adding the constraint $\sum_{i=1}^{d} \pi_i x_i \leq \pi_0$ to the original MIP and P^2 by adding the constraint $\sum_{i=1}^{d} \pi_i x_i \geq \pi_0 + 1$.

Even though any choice of (π, π_0) is valid for partitioning $\mathcal{S}$, a good choice is one that makes P^1, P^2 *easy* to solve. Note that the *ease* of solving P^1, P^2 again depends on the branching decisions and other techniques used to solve them. The *ease* or difficulty of solving the problem is function of the difficulty of proving that the problem is infeasible or of finding an optimal solution. A branching strategy can address either or both of these criteria. Let $f(S^1, S^2)$, denote a measure of the difficulty of solving the subproblems associated with feasible region $\mathcal{S}^0$. Then, the branching problem may be formulated as

$$\begin{aligned} &\min\{f(S^1, S^2)\} \\ s.t.\ &S^1 = S^0 \cap \{x \mid \pi x \leq \pi_0\} \\ &S^2 = S^0 \cap \{x \mid \pi x \geq \pi_0 + 1\} \\ &(\pi, \pi_0) \in \mathbb{Z}^{d+1}. \end{aligned} \tag{1.11}$$

Several different measures of f may be used and additional restrictions on the set (π, π_0) may be assumed. In fact, most of the existing branching techniques can be described using the above formulation. We review these techniques next.

When the branching disjunctions are limited to unit vectors, i.e., $\pi = e_i, i \in 1, 2, \ldots, d$, such a branching scheme is known as branching on variables or simply *variable branching*. This is the simplest and most widely used branching scheme for solving MIPs. If the optimal solution to the LP relaxation of the current subproblem is $\tilde{x}$, then a variable-branching could be of the form $x_i \leq \lfloor \tilde{x}_i \rfloor \vee x_i \geq \lceil \tilde{x}_i \rceil$ for some $i \in 1, 2, \ldots, d$ and $\tilde{x}_i$ fractional. The last condition ensures that $\tilde{x}$ is not a feasible solution in either subset created by the disjunction. Several rules have been proposed to select a branching variable out of the at most d available candidates. These are discussed next.

Dribeek [1966] suggested selecting a variable that would lead to maximum increase in the lower bound for the subproblem after branching. They proposed doing a few dual-simplex iterations to estimate this change. This technique later came to be known as *strong branching*. Later, Benichou et al. [1971] introduced the idea of using *pseudo-costs* to estimate this change in the lower bound. More specifically, suppose $\tilde{x}^k$ is the optimal solution of the LP relaxation of subproblem k and that $\tilde{x}_i^k$ has a fractional value for some $i = 1, 2, \ldots, d$. Further, suppose that the objective function value of the LP relaxation of the k^{th} subproblem is z^k and that this value changes to z^{k+1}, z^{k+2} after branching on variable i (i.e., imposing the disjunction $x_i \leq \lfloor \tilde{x}_i \rfloor \vee x_i \geq \lceil \tilde{x}_i \rceil,\ x \in \mathbb{R}^n$). Then the upper and lower pseudo-costs for variable i are defined as

$$PCL_i = \frac{z^{k+1} - z^k}{f_i^k}, \quad PCU_i = \frac{z^{k+1} - z^k}{1 - f_i^k}, \tag{1.12}$$

where $f_i^k = \tilde{x}_i^k - \lfloor \tilde{x}_i^k \rfloor$. Benichou et al. [1971] observed that the pseudo-cost of a variable does not usually change by more than an order of magnitude throughout execution of branch-and-bound algorithm and hence could be used to quickly estimate the changes in objective function values. Thus, if z^k is the objective value of the LP relaxation of the current subproblem k then the objective values after branching on a variable, say i, could be estimated as

$$\hat{z}_i^{k+1} = (PCL_i) f_i^k + z^k, \quad \hat{z}_i^{k+2} = (PCU_i)(1 - f_i^k) + z^k. \tag{1.13}$$

These pseudo-costs are initially unavailable and hence cannot be used when they are not initialized. Eckstein [1994] suggest that when pseudo-costs of only a few variables are available, the average of these could be used as an estimate. Linderoth and Savelsbergh [1999] performed extensive experiments to show that even though selecting variables after performing strong branching leads to much smaller branch-and-bound trees as compared to those obtained when using pseudo-costs, the overall time taken is less when pseudo-costs are used. They also suggest choosing the variable such that the convex combination of the estimated objective function values is maximized, i.e., select a variable i such that the expression

$$\alpha \min\{\hat{z}_i^{k+1}, \hat{z}_i^{k+2}\} + (1 - \alpha) \max\{\hat{z}_i^{k+1}, \hat{z}_i^{k+2}\} \tag{1.14}$$

has the maximum value for a constant $\alpha \in (0, 1)$.

Achterberg et al. [2005] combined the ideas of strong branching and pseudo-cost branching into a new method called *reliability-branching*. They suggest performing explicit strong branching on each variable at least a fixed number of times, called the reliability parameter, before using its pseudo-costs for selection. Their approach was shown to perform much better than using pseudo-costs alone both in the terms of the size of the tree and the time taken to solve. While most variable selection schemes have been limited to the criteria of objective function value, a notable exception is the approach of Patel and Chinneck [2007]. They select a variable on the basis of its coefficients in the so called active constraints, or the constraints that are satisfied as equalities by the optimal solution of the LP relaxation of the current subproblem. They show that their approach leads to faster solution times and fewer nodes as compared to a state-of-the-art commercial solvers.

On the other hand, the criteria for selecting general disjunctions for branching have primarily been based on the "integer width" of the feasible region associated with the LP relaxation. Given a polytope $\mathcal{P}$ and a direction $\pi \in \mathbb{Z}^n$, the width of $\mathcal{P}$ along the direction π, $w^\pi(\mathcal{P})$ is given by

$$w^\pi(\mathcal{P}) = \max_{x,y \in \mathcal{P}} \pi(x - y)$$

and it integer width is

$$w(\mathcal{P}) = \min_{\pi \in \mathbb{Z}^n, \pi \neq 0} w^\pi(\mathcal{P})$$

In their survey, Aardal and Eisenbrand [2004] discussed the fact that when the dimension is fixed, polynomial time algorithms for solving integer programs can be obtained by branching on general disjunctions obtained by determining the so-called *thin directions* of the feasible region, i.e., disjunctions along which the integer width of the feasible region is small. These polynomial time algorithms are derived from the seminal work of Lenstra [H.W. Lenstra, 1983] and mostly vary in the way they approximate the direction of the LP relaxation. It has also been shown, for instance by Krishnamoorthy and Pataki [2006], that certain specific problems can be solved "easily" if one branches on particular general disjunctions.

Some heuristics that may enhance the computational efficiency of branch-and-cut algorithms by branching on general disjunctions have also been proposed recently. Derpich and Vera [2006] obtain the direction of minimum width and then give score to each variable, the score being higher if the direction of branching on that variable is closer to that of minimum width. They then select a branching variable on the basis of this score. Two commonly found special structures are exploited by Beale and Tomlin [1970] to generate branching rules called SOS-1 and SOS-2. SOS-1 is used when only one variable out of a given set of binary variables may take a non-zero value, while SOS-2 can be used when only two adjacent variables from a given set of binary variables may be non-zero. Fischetti and Lodi [2003] proposed an improvement heuristic called *local-branching* in which the branching disjunction is of the form $\sum_{i \in I_1} x_i - \sum_{i \in I_2} \leq k \vee \sum_{i \in I_1} x_i - \sum_{i \in I_2} \leq k+1,\ x \in \mathbb{R}^n$, where $I_1 \subseteq I, I_2 \subseteq I$ are sets of the integer-constrained variables that are constructed on the basis of the best known solution and the solution of the LP relaxation of the current subproblem. Instead of solving it within the current branch-and-bound tree, the first subproblem is given to a MIP solver in the hope that it will find good solutions of that small problem quickly.

Owen and Mehrotra [2001] developed a heuristic based on strong branching in which they repeatedly add more variables to a disjunction with coefficients in {-1,0,1}, if by doing so the lower bound is improved. Karamanov and Cornuéjols [2007] use the fact that general disjunctions that are used to generate valid inequalities can be used for branching as well. They select those general disjunctions as candidates for branching that could be used to generate the Gomory Mixed Integer (GMI) inequalities (a type of valid inequality). They then explicitly evaluate the improvement in objective function value to select from a set of such disjunctions. Cornuéjols et al. [2008] try to improve the quality of GMI disjunctions used above by employing a method similar to that used by Andersen et al. [2005] to generate reduce-and-split inequalities.

1.4.2 Valid Inequalities

In a fashion similar to finding the best disjunction for branching as described above, one can formulate an optimization problem to generate valid inequalities. Suppose, for a given MIP of the form (1.1) with feasible region $\mathcal{S}^0$ and with the feasible region of its LP relaxation $\mathcal{P}$, we generate a valid inequality (α, β) from a given class $\mathcal{C}$ of valid inequalities. Let $g(\mathcal{P}, \alpha, \beta, d)$ be the measure of difficulty of solving the integer program after adding this inequality. Then the problem of selecting the best inequality may be written as

$$\begin{aligned} &\min g(\mathcal{P}, \alpha, \beta, d) \\ &s.t. \quad (\alpha, \beta) \in \mathcal{C}. \end{aligned} \tag{1.15}$$

Before describing some commonly used classes of valid inequalities, we first show that an inequality (α, β) is valid for $\mathcal{S}^0$ if and only if it is valid for some disjunction of the form (1.3) valid for $\mathcal{S}^0$. Since $\mathcal{S}^0 \subseteq \bigcup_{h\in\mathcal{Q}}\{x \in \mathbb{R}^n \mid A^h x \geq b^h\}$, it trivially follows that if (α, β) is a valid inequality for each subset created by a valid disjunction of $\mathcal{S}^0$, then it is also valid for $\mathcal{S}^0$. We now show that conversely, all valid inequalities for $\mathcal{S}^0$ can be derived from some disjunction. Since the variables x_i, $i = 1, 2, \ldots, d$ can assume only integer values in $\mathcal{S}^0$, the projection set $\mathcal{T} = \{t \in \mathbb{Z}^d \mid \exists x \in \mathcal{S}^0 \ s.t. \ x_i = t_i, i = 1, 2, \ldots, d\}$ is countable. Let $\mathcal{Q}$ be an index set of elements of $\mathcal{T}$ such that $t^h \in \mathbb{Z}^d$ is the h^{th} element of $\mathcal{T}$ for $h \in \mathcal{Q}$. Further, let $\mathcal{W}_{h\in\mathcal{Q}} = \{x \in \mathcal{S}^0 \mid x_i = t_i^h, i = 1, 2, \ldots, d\}$. Clearly, $\mathcal{S}^0 = \bigcup_{h\in\mathcal{Q}} \mathcal{W}_h$. It is also easy to see that $\mathcal{W}_h$ is a polyhedral set for each $h \in \mathcal{Q}$. This leads to the conclusion that $\mathcal{S}^0$ can be represented as a disjunction $\bigvee_{h\in\mathcal{Q}} \mathcal{W}_h$ and that any inequality that is valid for $\mathcal{S}^0$ is also valid for $\bigcup_{h\in\mathcal{Q}} \mathcal{W}_h$. Once it is realized that any valid inequality can be derived from some disjunction of $\mathcal{S}^0$, we can study specific families of valid inequalities as though derived from special restrictions of allowed disjunctions. We now describe two commonly used such families.

Probably the oldest and also the simplest class of valid inequalities for MIPs are the

Chvátal-Gomory (C-G) inequalities introduced by Gomory [1958]. Consider a pure IP in the standard form (1.1) with the set of feasible solutions $\mathcal{S}^0$ and the associated LP relaxation $\mathcal{P}$. Then for any $u \in \mathbb{R}^m_+$, $uA \geq ub$ is trivially a valid inequality for $\mathcal{P}$. For the particular case when uA assumes integer values, $uA\tilde{x}$ must also be integer valued for any point $\tilde{x} \in \mathcal{S}^0$ since both $uA, \tilde{x}$ are integer valued. Consider, for such a $u \in \mathbb{R}^m_+$, the general disjunction

$$uAx \leq \lceil ub \rceil - 1 \vee uAx \geq \lceil ub \rceil,\ x \in \mathbb{R}^n. \tag{1.16}$$

Since this is a valid disjunction for $\mathcal{S}^0$, $\mathcal{S}^0 \subseteq \{x \mid uAx \leq \lceil ub \rceil - 1\} \cup \{x \mid uAx \geq \lceil ub \rceil\}$. Then, the observation that $\mathcal{P} \cap \{x \in \mathbb{R}^n \mid uAx \leq \lceil ub \rceil - 1\} = \phi$ leads to the result that $\mathcal{S}^0 \subseteq \{x \mid uAx \geq \lceil ub \rceil\}$. Hence, $(uA, \lceil ub \rceil)$ is a valid inequality for $\mathcal{S}^0$ even though it may not be valid for $\mathcal{P}$ (when $ub \notin \mathbb{Z}$). Such an inequality is called a C-G inequality. So, an inequality $(\alpha, \lceil \beta \rceil), \alpha \in \mathbb{Z}^n, \beta \in \mathbb{R}$ is a C-G inequality if (α, β) is a valid inequality for $\mathcal{P}$. We further say that such an inequality is called an *elementary* C-G inequality if (α, β) is valid for $\mathcal{P}$. Let $\mathcal{P}^1_{CG}$ be the feasible region obtained after adding all possible elementary C-G cuts to $\mathcal{P}$. It is known that $\mathcal{P}^1_{CG}$ is also a polyhedron [Cook et al., 1998] and is called the elementary C-G closure of $\mathcal{P}$. All inequalities that are valid for $\mathcal{P}^1_{CG}$ but not for $\mathcal{P}$ are said to have C-G rank one. In general, inequalities valid for $\mathcal{P}^{i+1}_{CG}$ but not for $\mathcal{P}^i_{CG}$ are said to have rank i.

C-G cuts can also be interpreted in a slightly different manner. Consider again the pure IP as above. Now consider the following LP

$$\beta = \min_{x \in \mathcal{P}} \alpha x \tag{1.17}$$

for some $\alpha \in \mathbb{Z}^n$. Clearly, (α, β) is a valid inequality for $\mathcal{P}$. Hence, any point $\tilde{x} \in \mathcal{S}^0$ must satisfy $\alpha\tilde{x} \geq \lceil \beta \rceil$. Cook et al. [1990] generalized this idea in the following way. Consider the following problem

$$\begin{aligned} \beta = \min_{x \in \mathcal{P}} & \ \alpha x \\ s.t. & \quad \pi x \in \mathbb{Z} \end{aligned} \tag{1.18}$$

where $\pi \in \mathbb{Z}^n$ is fixed. Clearly all points in $\mathcal{S}^0$ must be feasible to the program (1.18) and thus the inequality $(\alpha, \lceil \beta \rceil)$ is again valid for $\mathcal{S}^0$. Cook et al. [1990] relaxed this problem to

$$\begin{aligned} \beta = \min_{x \in \mathcal{P}} & \ \alpha x \\ s.t. & \quad \pi x \leq \pi_0 \vee \pi x \geq \pi_0 + 1 \end{aligned} \tag{1.19}$$

where $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$. Thus, (α, β) is a valid inequality for $\mathcal{P} \cap \{x \in \mathbb{R}^n \mid \pi x \leq \pi_0\}$ and also $\mathcal{P} \cap \{x \in \mathbb{R}^n \mid \pi x \geq \pi_0 + 1\}$ even when $\alpha \notin \mathbb{Z}^n$. Since, (π, π_0) is a valid general disjunction for $\mathcal{S}^0$, it follows that (α, β) is a valid inequality for $\mathcal{S}^0$. Such an inequality is called a split inequality. These are sometimes also referred to as disjunctive inequalities since they are derived from some valid disjunction (π, π_0). However, we will use the term split inequality to refer to such inequalities to emphasize the fact that these were generated from some general disjunctions. The elementary split closure $\mathcal{P}_s^1$ is defined as the intersection of $\mathcal{P}$ and all possible elementary split inequalities. Cook et al. [1990] showed that $\mathcal{P}_s^1$ is also a polyhedron.

It is easy to see, by considering the disjunction $uAx \leq \lfloor ub \rfloor \vee uAx \geq \lceil ub \rceil,\ x \in \mathbb{R}^n$, that C-G inequalities are a special case of split inequalities. In fact, many other known classes of inequalities are also special cases of split inequalities: Gomory Mixed Integer Inequalities [Nemhauser and Wolsey, 1988], Lift and Project Cuts [Balas et al., 1993], Intersection Cuts [Balas, 1971], Mixed Integer Rounding Cuts [Nemhauser and Wolsey, 1990], reduce-and-split inequalities [Andersen et al., 2005] etc.

Most practical methods for generating valid inequalities first generate a set of valid inequalities using heuristics and then select the best ones from this set. The most commonly used criterion for selecting a valid inequality is the amount by which the current LP solution violates the given inequality after the coefficients (α, β) have been scaled appropriately. The higher the measure of violation, the "closer" it is likely to be to $conv(\mathcal{S}^0)$ and hence the "better" the inequality. We visit the criteria for selecting valid inequalities in Chapter 4 in greater detail.

A valid disjunction $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ that is used to derive a split inequality is a valid disjunction for branching as well. Since the main goal of this research is to study the problem of finding good disjunctions, we will repeatedly use this relation between branching on a general disjunction and generating split inequalities in our analysis and computational experiments.

1.5 Set up for Computational Experiments

We now describe the set up used to perform computational experiments for identifying and selecting general disjunctions for branching and generating valid inequalities. The actual experiments are described in Chapter 3 and Chapter 4. The instances used in the experiments have been collected from well-known libraries of MIP instances: MIPLIB 3.0 [Bixby et al., 1998], MIPLIB 2003 [Achterberg et al., 2006], the Mittelmann test set [Mittelmann, 2008]. Our test set has 177 instances in total. Some characteristics of size of these problems are described in Tables A.1. All experiments described in the thesis were performed on these instances only. We used only a subset of these instances in experiments that were too time consuming or when the experiment was not applicable to certain instances (like, for instance, if the best known upper bound is provided as an input, the LP relaxation becomes infeasible). In such cases, we list the subset of instances used.

All experiments were performed using a cluster of 64-bit machines, each with 16GB RAM, 8 Intel-Xeon 1.86GHz cores and 4MB cache. Each node of the cluster was running the CentOS operating system release-5. All tests were run by first submitting them to the CONDOR job scheduler that ensured there was at most one test running on each processor. The computational time reported in the experiments does not include the time used by the scheduler. The implementations used the ILOG CPLEX 10.2 callable library with appropriate callbacks, in order to override the default methods. The CoinUtils library available through COIN-OR was used for creating and modifying the sparse matrices used in the implementation. The program code was written in C and was compiled using the GCC compiler, version 4.1.2 with optimization level "-O".

1.6 Outline of Thesis and Contributions

We start in Chapter 2 by looking at the computational complexity of identifying the "best" possible disjunction for a given subproblem in a branch-and-bound algorithm. This requires a discussion about the criteria by which such disjunctions should be selected. Through this analysis, we provide several results for the computational complexity of choosing such disjunctions and also for that of generation of split inequalities. In Chapter 3, we show the computational effectiveness of using the disjunctions generated according to criteria outlined in Chapter 2. We also show effectiveness of some heuristic methods developed to speed up the performance of such a branching scheme. In Chapter 4 we study the effects of generating valid inequalities using the disjunctions generated with the techniques mentioned in Chapter 2 and compare them to other approaches used previously. Finally in Chapter 5, we describe some important questions for future research. The salient contributions of this work can be summarized as follows.

1. We show that the problem of selecting a general disjunction such that the LP relaxation becomes infeasible after branching is $\mathcal{NP}$-complete.

2. We show that the above problem is $\mathcal{NP}$-complete in the presence of several natural restrictions on the original problem and/or on the set of disjunctions considered.

3. The above results are shown to result in the following observation: Proving that a given inequality is an elementary split inequality is $\mathcal{NP}$-complete. We also show that this problem is different from the one of showing if the split rank of a given inequality is one.

4. We show that as a consequence of previous results, the problem of finding an elementary split inequality such that the lower bound on the problem is increased by the maximum is $\mathcal{NP}$-complete.

5. We provide a polynomial time algorithm that either generates an elementary split inequality to separate a given point on an edge of the LP relaxation of the problem or shows that no such inequalities exist.

6. A MIP is formulated to find a disjunction that when used for branching will result in a given lower bound. Extensive computational experiments show that such disjunctions can reduce drastically the size of branch-and-bound tree.

7. We develop a method to enumerate disjunctions that have two variables that is orders of magnitude faster than explicit enumeration. We also use the same idea to show that strong branching on variables can also be improved substantially.

8. Experiments described in Chapter 4 suggest that selecting C-G cuts that improve the lower bound by the maximum rather than those that are violated the most by the given solution of the LP relaxation.

Chapter 2

Theory of general disjunctions

2.1 Introduction

In this chapter, we study the computational complexity of the problem of selecting "optimal" general disjunctions for solving MIPs using branch-and-bound and cutting-plane algorithms. The motivation for studying the complexity of these problems is that such models may be useful in guiding branching decisions and generating valid inequalities in a practical context. In particular, these problems can, in principle, be solved at different stages of the branch-and-cut algorithm to select the "optimal" disjunction for partitioning the feasible region associated with a subproblem. In Chapters 3 and 4, we show by means of several experiments that the number of iterations for both branch-and-bound and cutting-plane algorithms can be reduced significantly by employing such selection procedures. However, the time required to solve these problems, when formulated as straightforward optimization problems, using a generic solver, is prohibitively large.

In this chapter, we show that these problems in fact lie in the complexity class $\mathcal{NP}$-hard, even for binary MIPs and even when certain restrictions are imposed on the structure of the disjunctions. Recall from Section 1.4.2 that the disjunctions used for branching may also be used to generate valid inequalities, called *split inequalities*. We show that the problem of deciding whether a given inequality is an elementary split inequality can be reduced to a problem related to determining an optimal branching disjunction. This immediately leads to a proof that the problem of deciding whether a given inequality is an elementary split inequality is $\mathcal{NP}$-complete. It is well known that one can find in polynomial time a split inequality (if any exists), that separates an extreme point of the feasible region of the LP relaxation from the elementary split closure of the given MIP. It is also well known that the same separation problem for any given point is $\mathcal{NP}$-complete. We show that a split inequality that separates any point on an edge of the feasible region of the LP relaxation from the elementary closure can be found in time polynomial in the size of inputs.

Before defining the problem of selecting a disjunction, it is necessary to describe the criterion for it. In its simplest form, the efficiency of the branch-and-bound procedure depends mainly on the number of subproblems generated. The goal of selecting a branching disjunction is then to minimize the total number of subproblems to be solved. Liberatore [2000] showed in the context of Satisfiability Problems (SATs) that the problem of finding an optimal variable disjunction (according to the criteria of minimizing the overall size of the search tree) is $\mathcal{NP}$-hard. Since SATs are reducible, in polynomial time, to MIPs, a similar result may be expected for the case of MIPs. In light of this, the problem of selecting an optimal general branching disjunction appears to be difficult. The approach taken by most solution procedures, and the one we shall take here, is then to evaluate candidate branching disjunctions by assessing their effect using more myopic criteria. The criterion of maximum bound improvement has been shown effective for selecting variable disjunctions (see Section 1.4.1) and we will use the same for selecting general disjunctions also. We will also compare this criterion with that of minimizing integer width, which has often been used for selecting general disjunctions.

In a similar vein, one would like to use as few disjunctions as possible for generating split inequalities in a cutting plane algorithm. The choice of branching disjunctions that lead to a small branch-and-bound tree may also be expected to generate valid inequalities that can solve the problem using only a few iterations of the cutting-plane algorithm. We therefore study the complexity of generating "best" valid inequalities based on the same criterion of maximum bound improvement.

The remainder of the chapter is organized as follows. We describe the problem of finding the optimal disjunction with respect to both the criteria of maximal bound improvement and minimum integer width in Section 2.2 and also provide MIP formulations that could be used to obtain an optimal disjunction based on these criteria. The computational complexity of these problems is discussed in Section 2.3. The results obtained there are then used to derive, in Section 2.4, analogous results for the problem of generating an elementary split inequality. In Section 2.6 we construct a polynomial time algorithm to separate points on an edge of an LP relaxation from the elementary split closure. Finally, in Section 2.7, we present conclusions.

2.2 Selecting Branching Disjunctions

Before addressing the complexity of the problem of selecting a branching disjunction, we first describe the problem of selecting a general branching disjunction to maximize the bound improvement and show how to construct a sequence of MIPs to solve it. We then show, using a similar approach, that the problem of selecting the thinnest integer direction can be also formulated as a MIP.

2.2.1 Maximizing Bound Improvement

The problem of selecting a disjunction to maximize bound improvement can be stated as follows. Consider the MIP (1.1) and assume that the associated polyhedron $\mathcal{P}$ is nonempty. Let $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1$ be a given general disjunction. Then the LPs associated with the subproblems created after branching are

$$\begin{aligned} z_L^* = \min\, & cx \\ \text{s.t.} \quad & Ax \geq b \\ & \hat{\pi}x \leq \hat{\pi}_0 \end{aligned} \qquad \text{and} \qquad \begin{aligned} z_R^* = \min\, & cx \\ \text{s.t.} \quad & Ax \geq b \\ & \hat{\pi}x \geq \hat{\pi}_0 + 1. \end{aligned} \tag{2.1}$$

Problem 1 (Greatest lower bound from branching on a general disjunction)**.** *Given a mathematical program of the form (1.1), find* $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1$ *such that* $\min\{z_L^*, z_R^*\}$ *is maximized, where* z_L^*, z_R^* *are as defined in (2.1).*

Problem 1 is an optimization problem and the associated decision problem is as follows.

Problem 2 (Lower bound from branching on a general disjunction)**.** *Given a mathematical program of the form (1.1) and* $z_l \in \mathbb{R}$*, does there exist* $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1$ *such that the LP relaxation associated with each subproblem (2.1) created after branching on* $(\hat{\pi}, \hat{\pi}_0)$ *has an optimal objective value of at least* z_l*, i.e.,* $\min\{z_L^*, z_R^*\} \geq z_l$*?*

Here, we describe a procedure to solve Problem 2 and then show in Section 2.3 that the problem is $\mathcal{NP}$-complete. We first consider a special case of this problem when the feasible region of the LPs associated with each subproblem is empty (i.e., $z_l = \infty$) and then extend the results to the general case. Consider the following problem.

Problem 3 (Disjunctive infeasibility)**.** *Given a mathematical program of the form (1.1), does there exist* $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1$ *such that the feasible region of the LP relaxations associated with each subproblem (2.1) created after branching on* $(\hat{\pi}, \hat{\pi}_0)$ *is empty, i.e.,* $\min\{z_L^*, z_R^*\} = \infty$*?*

The solution to Problem 3 does not depend upon the cost vector c because it is only desired to prove that the problem (1.1) is infeasible. The problem of finding a desired disjunction $(\hat{\pi}, \hat{\pi}_0)$ can be formulated as follows. Assume again that $\mathcal{P}$ is nonempty. Suppose $(\hat{\pi}, \hat{\pi}_0)$ is chosen such that both LPs (2.1) become infeasible. Then consider the following problems:

$$\begin{aligned} \zeta_L^* = \min\, & \hat{\pi}x \\ \text{s.t.} \quad & Ax \geq b \end{aligned} \qquad \text{and} \qquad \begin{aligned} \zeta_R^* = \min\, & -\hat{\pi}x \\ \text{s.t.} \quad & Ax \geq b. \end{aligned} \tag{2.2}$$

The dual of each of the above two programs (2.2) can be written, respectively, as:

$$\begin{aligned} \zeta_L^* = \max\, & pb \\ \text{s.t.} \quad & pA = \hat{\pi} \\ & p \geq 0 \end{aligned} \qquad \text{and} \qquad \begin{aligned} \zeta_R^* = \max\, & qb \\ \text{s.t.} \quad & qA = -\hat{\pi} \\ & q \geq 0. \end{aligned} \tag{2.3}$$

The programs (2.1) are infeasible if and only if $\zeta_L^* > \hat{\pi}_0$ and $\zeta_R^* > -(\hat{\pi}_0 + 1)$. By using this condition and combining the above two dual formulations, one can get the desired formulation for giving an answer to Problem 3. More precisely, the LPs in (2.1) are infeasible if and only if the system

$$\begin{aligned} pA - \pi &= 0 \\ qA + \pi &= 0 \\ pb - \pi_0 &> 0 \\ qb + \pi_0 &> -1 \\ p &\geq 0 \\ q &\geq 0 \\ (\pi, \pi_0) &\in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1, \end{aligned} \tag{2.4}$$

has a feasible solution with $\pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$.

Once we have a formulation that may be solved in order to answer Problem 3, we can extend it in the usual way to address Problem 2 as well. More details about this procedure are described in Chapter 3. Problem 3 is equivalent to that of finding $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1$ such that $\mathcal{P} \subseteq \{x \in \mathbb{R}^n \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$. If such a $(\hat{\pi}, \hat{\pi}_0)$ exists, then the “width” $\mathcal{P}$ is less than one. In the next section, we study the problem of minimizing the integer width of $\mathcal{P}$ and show that the general framework described above can be extended to this problem as well.

2.2.2 Minimizing Integer Width

Assuming that $\mathcal{P}$ is full dimensional, the *width* of $\mathcal{P}$ in direction π is $\max_{x,y\in\mathcal{P}}(\pi x - \pi y)$, while the *integer width* of $\mathcal{P}$ is

$$w(\mathcal{P}) = \min_{\pi} \max_{x,y\in\mathcal{P}} (\pi x - \pi y), \pi \in \mathbb{Z}^d \times \{0\}^{n-d}, \pi \neq 0.$$

Then, a vector π that is obtained from the above optimization problem, along with a scalar $\pi_0 = \lfloor \pi x^* \rfloor$, where x^* is the optimal solution of the LP relaxation (1.1), can be used to determine a disjunction for branching. Sebő [1999] showed that the problem of determining whether $w(\mathcal{P}) \leq 1$ is $\mathcal{NP}$-complete, even when $\mathcal{P}$ is a simplex. It is also known, from a result of Banaszczyk et al. [1999], that if $\mathcal{P} \cap \mathbb{Z}^{n-d}$ is empty, then $w(\mathcal{P}) \leq Cn^{\frac{3}{2}}$, where C is a constant. Derpich and Vera [2006] approximate the direction of the minimum integer width in order to assign priorities for branching on variables and use this to select variable disjunctions.

The width of $\mathcal{P}$ in a given direction $\hat{\pi}$ can be obtained by solving the LP

$$\begin{aligned} \max\ & \hat{\pi}x - \hat{\pi}y \\ \text{s.t.}\ & Ax \geq b \\ & Ay \geq b. \end{aligned} \tag{2.5}$$

The dual associated with the LP (2.5) is

$$\begin{aligned} \min\ & -qb - pb \\ \text{s.t. } & pA = \hat{\pi} \\ & qA = -\hat{\pi} \\ & p, q \geq 0. \end{aligned} \tag{2.6}$$

Therefore, the problem of finding $w(\mathcal{P})$ can be written as

$$\begin{aligned} \min\ & -qb - pb \\ \text{s.t. } & pA - \pi = 0 \\ & qA + \pi = 0 \\ & \pi \neq 0 \\ & \pi \in \mathbb{Z}^d \times \{0\}^{n-d} \\ & p, q \geq 0. \end{aligned} \tag{2.7}$$

Since the disjunction (π, π_0) is the same as the disjunction $(-\pi, -\pi_0 - 1)$, the condition $\pi \neq 0$ can be replaced by the inequality $\sum_{i=1}^{n} \pi_i \geq 1$. Problem (2.7) can now be solved as a MIP.

Note that if there exists a $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$ that satisfies the formulation (2.4), then $w(\mathcal{P}) < 1$. However, the converse is not true. To see this, consider as an example $\mathcal{P} = \{x \in \mathbb{R}^2_+ \mid 3 \leq 4x_1 + 4x_2 \leq 5\}$ and $d = 2$. Then, even though $w(\mathcal{P}) \leq \frac{1}{2} < 1$, (1.1) is still feasible. Comparing formulations (2.4) and (2.7), one can see that (2.4) is more constrained than (2.7). As a result, there may be some benefit to using solutions to the formulation (2.4) to generate branching disjunctions over those of (2.7). A feasible solution to formulation (2.4) guarantees that the LP relaxations associated with both subproblems created after branching are infeasible and therefore gives a short proof of infeasibility, provided that such a short proof exists. Branching along a direction of minimum width does not guarantee this. As an example, consider a MIP with feasible region $\{x \in \mathbb{Z}^2_+ \mid 7 \leq 8x_1 + 8x_2 \leq 9, -3 \leq 4x_1 - 4x_2 \leq 3\}$. Branching on the disjunction $x_1 \leq 0 \vee x_1 \geq 1$ immediately makes LP relaxation of each subproblem infeasible while branching along a direction of minimum width ($w(\mathcal{P}) = 0.25$), $x_1 + x_2 \geq 2 \vee x_1 + x_2 \leq 1$ results in two subproblems out of which one still has a feasible LP relaxation and needs further processing. Krishnamoorthy [2008] showed that, in general, branching along a direction of minimum width need not result in a small branch-and-bound tree, even in higher dimensions.

Even though there are some similarities in the formulations (2.4) and (2.7), it is not easy to reduce the problem of finding $w(\mathcal{P})$ to Problem 3. Therefore, we use a different approach to address the complexity of the latter.

2.3 Complexity of Selecting Branching Disjunctions

For the case when (π, π_0) is restricted to variable disjunctions only, Problem 3 can be solved in time polynomial in the size of the input by solving the two LPs associated with each of the n possible variable disjunctions. The following results show that the problem becomes difficult in the case of general disjunctions. We first show that Problem 3 is $\mathcal{NP}$-complete. We then show that the problem remains $\mathcal{NP}$-complete even when several common restrictions are introduced.

Lemma 2.3.1. *If $(\hat{\pi}, \hat{\pi}_0, \hat{p}, \hat{q})$ is a feasible solution to (2.4), then $\hat{\pi}_0 < \hat{p}b \leq -\hat{q}b < \hat{\pi}_0 + 1$.*

Proof. The first and last inequalities come directly from the formulation (2.4). Let $\zeta^*_L = \min_x\{\hat{\pi}x \mid Ax \geq b\}, \zeta^*_R = \max_x\{\hat{\pi}x \mid Ax \geq b\}$. Then $\zeta^*_L \leq \zeta^*_R$. Also, $p = \hat{p}$ and $q = \hat{q}$ are feasible solutions to the dual programs (2.3). By using weak duality on the associated LPs (2.2), we get that $\zeta^*_L \geq \hat{p}b$ and $\zeta^*_R \leq -\hat{q}b$. Thus, $\hat{\pi}_0 < \hat{p}b \leq \zeta^*_L \leq \zeta^*_R \leq -\hat{q}b <$

$\hat{\pi}_0 + 1$. $\square$

We assume for the remainder of the section that the mathematical program (1.1) is a pure integer program, i.e., that $n = d$. This assumption is made for notational convenience only—the results are also valid for the case when $d < n$.

We first show that the Problem 3 is in the complexity class $\mathcal{NP}$. If the matrices A, b have integer entries only, then we claim that constraints $p < \mathbf{1}$, where $\mathbf{1}$ is the vector of all ones, may be added to (2.4) without any loss of generality. In order to see this, suppose the formulation (2.4) has a feasible solution with $p = \hat{p}, q = \hat{q}, \pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$. Further suppose that $\hat{p}_i \geq 1$ for some $i, 1 \leq i \leq m$. Then $p = \hat{p} - e_i, q = \hat{q} + e_i, \pi = \hat{\pi} - a_i, \pi_0 = \hat{\pi}_0 - b_i$ is also a feasible solution. Here, e_i is the i^{th} unit vector and a_i the i^{th} row of the matrix A. This process can be applied repeatedly until p is component-wise less than 1. If we assume that $p < \mathbf{1}$, then $|pb| \in [0, \sum_{i=1}^{m} |b_i|)$. Also, using (2.4), $|\pi_j| \in [0, \sum_{i=1}^{m} |a_{ij}|), j = 1, \ldots, n$. Using Lemma 2.3.1, this implies $|\pi_0| \leq |pb| \leq \sum_{i=1}^{m} |b_i|$. So, if the system (2.4) is feasible, then a feasible solution may be expressed in size that is polynomial in the size of the input. Also, given a $(\hat{\pi}, \hat{\pi}_0)$, one can determine whether a disjunction on $(\hat{\pi}, \hat{\pi}_0)$ will make the LPs (2.1) infeasible in time that is polynomial in the size of the input by solving the two linear programs. This shows that Problem 3 lies in the complexity class $\mathcal{NP}$.

Before further addressing the complexity of Problem 3, we consider the same problem applied to a system of linear Diophantine equations in place of the system of form (1.1). Suppose we are given a system of linear Diophantine equations of the form,

$$\begin{aligned} Ax &= b \\ x &\in \mathbb{Z}^n. \end{aligned} \tag{2.8}$$

Such equations can be solved in time polynomial in the size of the input [Nemhauser and Wolsey, 1988, pg. 191]. A branching disjunction $(\hat{\pi}, \hat{\pi}_0)$ that can make the associated LP relaxations of (2.8) infeasible can be shown, by using the approach above, to satisfy (along with a suitable $\hat{p}, \hat{q}$) the system

$$\begin{aligned} pA &= \pi, \\ -qA &= \pi, \\ pb &> \pi_0, \\ -qb &< 1 + \pi_0, \text{ and} \\ (\pi, \pi_0) &\in \mathbb{Z}^{n+1}. \end{aligned} \tag{2.9}$$

We claim that the system (2.9) can be solved in time polynomial in the size of the input. The system of Diophantine equations (2.8) is infeasible if and only if there exists a λ such that $\lambda A \in \mathbb{Z}^m$ and $\lambda b \notin \mathbb{Z}$ [Nemhauser and Wolsey, 1988, pg. 191]. Further, if (2.8) is infeasible, then such a λ can be found in polynomial time. In such a case, $p = -q = \lambda, \pi = \lambda A, \pi_0 = \lfloor \lambda b \rfloor$ is a feasible solution to (2.9). Conversely, suppose that (2.8) has a feasible solution x^0. Such a feasible solution can be found in polynomial time. Then for any $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$, $\pi x^0 = pAx^0 = pb$ and $\pi x^0 = -qAx^0 = -qb$. Since $\pi x^0 \in \mathbb{Z}$, there is no $\pi_0 \in \mathbb{Z}$ such that $\pi_0 > \pi x^0$ and $\pi_0 < \pi x^0 + 1$. Thus, in this case, the existence of the solution x^0 is sufficient to show that (2.9) is infeasible. So a feasible solution for the system (2.9) can be found or it can be shown that no such solution exists in time polynomial in the size of the input.

Now consider the problem (1.1) again (recall the assumption that $n = d$). We just have shown that the problem of finding $\lambda \in \mathbb{R}^m$ such that $\lambda A \in \mathbb{Z}^n, \lambda b \notin \mathbb{Z}$ is easy. Existence of such a λ is a necessary condition for the feasibility of a given program of the form (2.4). To see this, suppose $p = \hat{p}, \pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$ are feasible for (2.4) and substitute $\lambda = \hat{p}$. Then $\lambda A = \hat{\pi} \in \mathbb{Z}^n$, but $\hat{\pi}_0 < \lambda b < \hat{\pi}_0 + 1$. Existence of such a λ is not, however, sufficient for feasibility of (2.4). For instance, consider the set: $\mathcal{P} \cap \mathbb{Z}^2 = \{x \in \mathbb{Z}^2 \mid 3x_1 + 6x_2 \geq 2\}$ and $\lambda = \frac{1}{3}$. Clearly $\lambda A \in \mathbb{Z}^2, \lambda b \notin \mathbb{Z}$. Still, $\mathcal{P} \cap \mathbb{Z}^n$ has at least one feasible point $(1, 0)$. This provides a hint that Problem 3 may not be easy.

We now show that Problem 3 is $\mathcal{NP}$-complete by reducing the well-known number partitioning problem to Problem 3. The Number Partitioning Problem $PARTITION$ is defined as follows

Problem 4 ($PARTITION$, [Garey and Johnson, 1979])**.** *Given a finite set $\mathcal{S}$ and a size $a^i \in \mathbb{Z}_+$ for each $i \in \mathcal{S}$. Is there a subset $K \subseteq \mathcal{S}$ such that $\sum_{i \in K} a^i = \sum_{i \in \mathcal{S} \setminus K} a^i$?*

Proposition 2.3.1. *Problem 3 is $\mathcal{NP}$-complete.*

Proof. The proof is a modification of the approach used by Sebő [1999] for the problem of finding integer width. Consider the Problem 4 above, which is known to be $\mathcal{NP}$-complete. Let $n \in \mathbb{N}$, $\mathcal{S} = \{1, 2, \ldots, (n-1)\}, a^i \in \mathbb{Z}_+, i \in \mathcal{S}$ be inputs for Problem 4. Let $s = \frac{1}{2}\sum_{i \in \mathcal{S}} a^i$. An instance of Problem 4 can be answered "yes" if and only if there exists a set $K \subseteq \{1, 2, \ldots, n-1\}$ such that $\sum_{i \in K} a^i = s$. Since multiplying each a^i by 4 results in a problem equivalent to Problem 4, it is assumed, without loss of generality, that $s \in \mathbb{Z}_+, s \geq 2$. Problem 4 can be reduced to Problem 3 as follows. Consider the simplex $\mathcal{P}_s$ of points $v^i, i = 1 \ldots n+1$ in n dimensions, with the coordinates of v^i defined as

$$v^i_j = \begin{cases} \frac{1}{2n} & \text{if } j \neq i, i = 1, 2, \ldots, n, \\ \frac{1}{2n} + \frac{1}{2} & \text{if } j = i, i = 1, 2, \ldots, n, \\ a^j & \text{if } i = n+1, j = 1, 2, \ldots, n-1, \\ -\frac{1}{2}\sum_{k=1}^{n-1} a^k + \frac{1}{2} & \text{if } i = n+1, j = n. \end{cases}$$

So, $v^1 = (\frac{1}{2n} + \frac{1}{2}, \frac{1}{2n}, \frac{1}{2n}, \ldots, \frac{1}{2n}), v^2 = (\frac{1}{2}, \frac{1}{2} + \frac{1}{2n}, \frac{1}{2n}, \ldots, \frac{1}{2n}), \ldots, v^n = (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2n} + \frac{1}{2}), v^{n+1} = (a^1, a^2, \ldots, a^{n-1}, -s + \frac{1}{2})$. We will show that the desired subset K exists if and only if there exists $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$, such that $\mathcal{P}_s \subseteq \{x \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$.

Suppose the desired subset K exists, i.e., K is a set such that $\sum_{i \in K} a^i = s$. Let

$\hat{\pi}_i = 1, i \in K$, $\hat{\pi}_n = 1$, $\hat{\pi}_i = 0, i \notin K \cup \{n\}$, $\hat{\pi}_0 = 0$. Then, $0 < \hat{\pi} v^i < 1, i = 1, 2, \ldots, n$. Also, $v^{n+1}\hat{\pi} = \frac{1}{2}$. Since all vertices of $\mathcal{P}_s$ satisfy the condition $\hat{\pi}_0 < \hat{\pi} x < \hat{\pi}_0 + 1$, $\mathcal{P}_s \subseteq \{x \mid \hat{\pi}_0 < \hat{\pi} x < \hat{\pi}_0 + 1\}$. $(\hat{\pi}, \hat{\pi}_0)$ is then the required disjunction.

Conversely, suppose there exists some $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$ such that $\mathcal{P}_s \subseteq \{x \mid \hat{\pi}_0 < \hat{\pi} x < \hat{\pi}_0 + 1\}$. Then, $\hat{\pi}_0 < \hat{\pi} v^i < \hat{\pi}_0 + 1, i = 1, 2, \ldots, (n+1)$ and $|\hat{\pi}(v^i - v^k)| < 1, i = 1, 2, \ldots, n, k = 1, 2, \ldots, n$. Substituting the coordinates of v^i and v^k, one gets that $|\frac{\hat{\pi}_i - \hat{\pi}_k}{2}| < 1$. Since $\hat{\pi}_i, \hat{\pi}_k \in \mathbb{Z}$, this means that $|\hat{\pi}_i - \hat{\pi}_k| \leq 1$ for each pair $(i,k) \in \{1, 2, \ldots, n\}^2$. So, $\hat{\pi}_i \in \{t, t+1\}, i = 1, 2, \ldots, n$ for some $t \in \mathbb{Z}$. Since disjunction $(\hat{\pi}, \hat{\pi}_0)$ is equivalent to disjunction $(-\hat{\pi}, -\hat{\pi}_0 - 1)$, it can be assumed without loss of generality that $t \geq 0$. Let $K = \{i \mid \hat{\pi}_i = t + 1\}$. Substituting the coordinates of v^1 and $\hat{\pi}$ into the inequalities $\hat{\pi}_0 < v^1\hat{\pi} < \hat{\pi}_0 + 1$, one gets

$$\begin{aligned} \hat{\pi}_0 < & \sum_{i=1}^{n} v_i^1 \hat{\pi}_i < \hat{\pi}_0 + 1 \\ \Rightarrow \hat{\pi}_0 < & \sum_{i=1}^{n} \frac{\hat{\pi}_i}{2n} + \frac{\hat{\pi}_1}{2} < \hat{\pi}_0 + 1 \\ \Rightarrow \hat{\pi}_0 < & \frac{t}{2} + \sum_{i \in K} \frac{1}{2n} + \frac{\hat{\pi}_1}{2} < \hat{\pi}_0 + 1. \end{aligned}$$

Since $\frac{\hat{\pi}_1}{2} \in \{\frac{t}{2}, \frac{t+1}{2}\}$, the only integer value of $\hat{\pi}_0$ that satisfies the above condition is $\hat{\pi}_0 = t$. Thus $\hat{\pi} \in \{t, t+1\}^n, \hat{\pi}_0 = t$. Also, $K = \phi$ would mean that $\hat{\pi}_0 < t < \hat{\pi}_0 + 1$. This is not possible for any integers $t, \hat{\pi}_0$. Hence K is not empty. The condition $\hat{\pi}_0 < \hat{\pi} v^{n+1} < \hat{\pi}_0$

implies:

$$\begin{aligned}
\hat{\pi}_0 <& \sum_{i=1}^{n} v_i^{n+1}\hat{\pi}_i < \hat{\pi}_0 + 1 \\
\Rightarrow \hat{\pi}_0 <& \sum_{i=1}^{n-1} \hat{\pi}_i a^i - \hat{\pi}_n s + \frac{\hat{\pi}_n}{2} < \hat{\pi}_0 + 1 \\
\Rightarrow t \ <& \ t\sum_{i=1}^{n-1} a^i + \sum_{i\in K} a^i - \hat{\pi}_n s + \frac{\hat{\pi}_n}{2} < t+1 \\
\Rightarrow t \ <& \ 2ts + \sum_{i\in K} a^i - \hat{\pi}_n s + \frac{\hat{\pi}_n}{2} < t+1.
\end{aligned}$$

Now there are two cases. Suppose $\hat{\pi}_n = t$. Then the above condition implies that $t < ts + \sum_{i\in K} a^i + \frac{t}{2} < t+1$. This is not possible because $s \geq 2$ and $K \neq \phi$. Thus, $\hat{\pi}_n$ must equal $t+1$. In this case, the above condition becomes:

$$\begin{aligned}
t <& \ 2ts + \sum_{i\in K} a^i - ts - s + \frac{t+1}{2} < t+1 \\
\Rightarrow t <& \ (t-1)s + \sum_{i\in K} a^i + \frac{t+1}{2} < t+1.
\end{aligned}$$

Since $s \geq 2$ and $K \neq \phi$, the only value that t may assume is $t = 0$. That means $0 < \sum_{i\in K} a^i - s + \frac{1}{2} < 1$. Thus $\sum_{i\in K} a^i = s$ and K is the required subset for the Problem 4.

Thus, given a simplex $\mathcal{P}_s$, the problem of finding $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$ is $\mathcal{NP}$-complete. Since $\mathcal{P}_s$ is a simplex, its description can be transformed into form (1.1) in time polynomial in the size of the description of $\mathcal{P}_s$. This completes the required proof. □

Even though the above proof did not assume any restrictions on values of $(\hat{\pi}, \hat{\pi}_0)$, the reduction from Problem 4 imposed the conditions $\hat{\pi} \in \{0,1\}^n$. This shows that several restrictions of Problem 3 are also $\mathcal{NP}$-complete. Some of these are listed below

Proposition 2.3.2. *The following restrictions of Problem 3 are $\mathcal{NP}$-complete.*

1. *Given a mathematical program of the form (1.1), does there exist* $(\hat{\pi}, \hat{\pi}_0) \in \{0,1\}^{n+1}$ *such that the LP relaxation of both the subproblems (2.1) created after branching on* $(\hat{\pi}, \hat{\pi}_0)$ *are infeasible.*

2. *Given a mathematical program of the form (1.1), does there exist* $\hat{\pi} \in \{0,1\}^n$ *such that the LP relaxation of both the subproblems (2.1) created after branching on* $(\hat{\pi}, 0)$ *are infeasible.*

3. *Given a mathematical program of the form (1.1), does there exist* $\hat{\pi} \in \{0,1\}^n, \hat{\pi}_0 \in \mathbb{Z}$ *such that the LP relaxation of both the subproblems (2.1) created after branching on* $(\hat{\pi}, \hat{\pi}_0)$ *are infeasible.*

4. *Given a mathematical program of the form (1.1), does there exist* $\hat{\pi} \in \mathbb{Z}^n_+$ *such that LP relaxation of both the subproblems (2.1) created after branching on* $(\hat{\pi}, 0)$ *are infeasible.*

5. *Given a mathematical program of the form (1.1), does there exist* $\hat{\pi} \in \mathbb{Z}^n$ *such that the LP relaxation of both the subproblems (2.1) created after branching on* $(\hat{\pi}, 0)$ *are infeasible.*

6. *Given a mathematical program of the form (1.1), does there exist* $\hat{\pi} \in \{0,1,-1\}^n, \hat{\pi}_0 \in \mathbb{Z}$ *such that the LP relaxation of both the subproblems (2.1) created after branching on* $(\hat{\pi}, \hat{\pi}_0)$ *are infeasible. (This problem is mentioned because Owen and Mehrotra [2001] developed a greedy heuristic for the optimization version of this problem, without addressing the complexity of the problem).*

Proof. The proof of each of the above propositions follows directly from the proof of Proposition 2.3.1 above. □

If Q is a polytope, then the fact that $Q \subseteq \{x \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$, for some $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$, is sufficient to show that $w(Q) < 1$. The proof provided above settles the question of complexity of finding such a sufficient condition. If a program of form (1.1) has only binary variables, i.e., it is of the form:

$$\begin{aligned} &\min cx \\ \text{s.t. } &Ax \geq b \\ &x \in \{0,1\}^n, \end{aligned} \tag{P_b}$$

then the width of the associated polyhedron $\mathcal{P}_b$ is trivially at most one. The following proposition shows that the problem of deciding whether there exists a disjunction $(\hat{\pi}, \hat{\pi}_0)$ that will prove the infeasibility of a binary program is also $\mathcal{NP}$-complete.

Problem 5 (Disjunctive infeasibility for binary programs)**.** *Given a mathematical program of the form (P_b), does there exist a disjunction $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$, that proves infeasibility?*

Problem 5 is a special case of Problem 3 and hence the proof of $\mathcal{NP}-$completeness of the latter follows from that of the former. However, we address the complexity of Problem 5 separately because the proof is easier to understand having seen that of Problem 3.

Proposition 2.3.3. *Problem 5 is $\mathcal{NP}$-complete.*

Proof. The proof is similar to that of Proposition 2.3.1. Let $n \in \mathbb{Z}_+$, $\mathcal{S} = \{1, 2, \dots, (n-1)\}, a^i \in \mathbb{Z}_+, i \in \mathcal{S}$ be inputs for an instance of Problem 4. We need to modify our previous transformation because coordinates of the feasible region of $\mathcal{P}$ can only lie in $[0, 1]$, while $a^i \in \mathbb{Z}_+, i \in \mathcal{S}$. Let $M = \sum_{i \in \mathcal{S}} a^i$ and $m = \frac{1}{M}$. If each $a^i, i \in \mathcal{S}$ is divided by M, then the problem of partitioning remains the same. Let $\tilde{a}^i (= \frac{a^i}{M}) \in \mathbb{Q}_+, i \in \mathcal{S}$ so that $\sum_{i \in \mathcal{S}} \tilde{a}^i = 1$. The answer to an instance of Problem 4 is "yes" if and only if there exists a set $K \subseteq \{1, 2, \dots, n-1\}$ such that $\sum_{i \in K} \tilde{a}^i = \frac{1}{2}$. Since each $\tilde{a}^i$ is an integer multiple of $\frac{1}{M}$, there is no $K \subseteq \mathcal{S}$ such that $\sum_{i \in K} \tilde{a}^i \in [\frac{1}{2} - \frac{1}{2M}, \frac{1}{2})$ or $\sum_{i \in K} a^i \in (\frac{1}{2}, \frac{1}{2} + \frac{1}{2M}]$. This observation will be useful later.

Problem 4 can now be reduced to Problem 5 as follows. Let $\epsilon = \frac{1}{2M} = \frac{m}{2}$. Consider the convex hull, $\mathcal{P}_s$, of points $v^i, i = 1 \dots n+3$ in $n+1$ dimensions, where the coordinates of v^i are defined as

$$v^i_j = \begin{cases} \frac{1}{2n} & \text{if } j \neq i, j \neq n, j \neq n+1, i = 1, 2, \dots, n-1 \\ \frac{1}{2n} + \frac{1}{2} & \text{if } j = i, i = 1, 2, \dots, n-1 \\ 0 & \text{if } j = n, n+1, i = 1, 2, \dots, n-1 \\ & \\ \tilde{a}^j & \text{if } j = 1, 2, \dots, n-1, i = n \\ 1 & \text{if } j = n, n+1, i = n \\ & \\ \tilde{a}^j & \text{if } j = 1, 2, \dots, n-1, i = n+1 \\ \frac{1}{2} - \epsilon & \text{if } j = n, i = n+1 \\ 0 & \text{if } j = n+1, i = n+1 \\ & \\ \tilde{a}^j & \text{if } j = 1, 2, \dots, n-1, i = n+2 \\ 0 & \text{if } j = n, i = n+2 \\ \frac{1}{2} - \epsilon & \text{if } j = n, i = n+2 \\ & \\ \frac{1}{2} & \text{if } j = n, i = n+3 \\ 0 & \text{if } j \neq n, i = n+3 \end{cases}$$

This means $v^1 = (\frac{1}{2n} + \frac{1}{2}, \frac{1}{2n}, \frac{1}{2n}, \dots, 0, 0)$, $v^2 = (\frac{1}{2n}, \frac{1}{2n} + \frac{1}{2}, \frac{1}{2n}, \frac{1}{2n}, \dots, 0, 0)$ etc. $v^{n-1} = (\frac{1}{2n}, \dots, \frac{1}{2n} + \frac{1}{2}, 0, 0)$, $v^n = (\tilde{a}_1, \tilde{a}_2, \dots, \tilde{a}_{n-1}, 1, 1)$, $v^{n+1} = (\tilde{a}_1, \dots, \tilde{a}_{n-1}, \frac{1}{2} - \epsilon, 0)$, $v^{n+2} = (\tilde{a}_1, \dots, \tilde{a}_{n-1}, 0, \frac{1}{2} - \epsilon)$, $v^{n+3} = (0, 0, \dots, 0, \frac{1}{2}, 0)$. Clearly, $\mathcal{P}_s \subseteq \{x \in \mathbb{R}^{n+1} \mid 0 \leq x_i \leq 1, i = 1, 2, \dots, n\}$. It will now be shown that a $K \subseteq \mathcal{S}$ such that $\sum_{i \in K} \tilde{a}^i = \frac{1}{2}$ exists if and only if there exists $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$, such that $\mathcal{P}_s \subseteq \{x \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$. Suppose $K \subseteq \mathcal{S}$ such that $\sum_{i \in K} \tilde{a}^i = \frac{1}{2}$. Let $\hat{\pi}_i = 1, i \in K$, $\hat{\pi}_n = 1, \hat{\pi}_{n+1} = -1$, $\hat{\pi}_i = 0, i \notin K \cup \{n, n+1\}$,

$\hat{\pi}_0 = 0$. Then, $0 < \hat{\pi}v^i < 1, i = 1, 2, \dots, n+3$. Since all vertices of $\mathcal{P}_s$ satisfy the condition $\hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1$, $\mathcal{P}_s \subseteq \{x \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$. $(\hat{\pi}, \hat{\pi}_0)$ is then the required disjunction.

Conversely, suppose there exists some $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$ such that $\mathcal{P}_s \subseteq \{x \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$. Then, $\hat{\pi}_0 < \hat{\pi}v^i < \hat{\pi}_0 + 1, i = 1, 2, \dots, (n+3)$. This also means that $|\hat{\pi}(v^i - v^k)| < 1, i = 1, 2, \dots, n-1, k = 1, 2, \dots, n-1$. Substituting the coordinates of v^i and v^k, one gets: $|\frac{\hat{\pi}_i - \hat{\pi}_k}{2}| < 1$. Because $\hat{\pi}_i, \hat{\pi}_k \in \mathbb{Z}$, this means that $|\hat{\pi}_i - \hat{\pi}_k| \leq 1$ for each pair $(i, k) \in \{1, 2, \dots, n-1\}^2$. This means that $\hat{\pi}_i \in \{t, t+1\}, i = 1, 2, \dots, n-1$ for some $t \in \mathbb{Z}$. Let $K = \{i \in \mathcal{S} \mid \pi_i = t+1\}$. Substituting the coordinates of v^1 and $\hat{\pi}$ into the inequalities $\hat{\pi}_0 < v^1\hat{\pi} < \hat{\pi}_0 + 1$, one gets:

$$\begin{aligned} \hat{\pi}_0 < & \sum_{i=1}^{n+1} \frac{\hat{\pi}_i}{2n} + \frac{\hat{\pi}_1}{2} < \hat{\pi}_0 + 1 \\ \Rightarrow \hat{\pi}_0 < \frac{t}{2} + & \sum_{i \in K} \frac{1}{2n} + \frac{\hat{\pi}_1}{2} < \hat{\pi}_0 + 1. \end{aligned}$$

Since $\frac{\hat{\pi}_1}{2} \in \{\frac{t}{2}, \frac{t+1}{2}\}$, the only integer value of $\hat{\pi}_0$ that satisfies the above condition is $\hat{\pi}_0 = t$. Thus, $\hat{\pi} \in \{t, t+1\}^n, \hat{\pi}_0 = t$. Also, $K = \phi$ would mean that $\hat{\pi}_0 < t < \hat{\pi}_0 + 1$. This is not possible for any integers $t, \hat{\pi}_0$. Hence, K is not empty. The condition $\hat{\pi}_0 < \hat{\pi}v^n < \hat{\pi}_0 + 1$ implies:

$$\begin{aligned} \hat{\pi}_0 < & \sum_{i=1}^{n-1} \hat{\pi}_i \tilde{a}^i + \hat{\pi}_n + \hat{\pi}_{n+1} < \hat{\pi}_0 + 1 \\ \Rightarrow \quad t < t & \sum_{i=1}^{n-1} \tilde{a}^i + \sum_{i \in K} \tilde{a}^i + \hat{\pi}_n + \hat{\pi}_{n+1} < t + 1 \\ \Rightarrow & \qquad \hat{\pi}_{n+1} = -\hat{\pi}_n \end{aligned}$$

The condition $\hat{\pi}_0 < \hat{\pi} v^{n+1} < \hat{\pi}_0 + 1$ implies:

$$\begin{aligned} t < \sum_{i=1}^{n-1} \hat{\pi}_i \tilde{a}^i + \hat{\pi}_n (\frac{1}{2} - \epsilon) < t+1 \\ \Rightarrow 0 < \quad \sum_{i \in K} \tilde{a}^i + \hat{\pi}_n (\frac{1}{2} - \epsilon) < 1. \end{aligned} \tag{2.10}$$

The condition $\hat{\pi}_0 < \hat{\pi} v^{n+2} < \hat{\pi}_0 + 1$ implies:

$$\begin{aligned} t < t + \sum_{i \in K} \tilde{a}^i + \hat{\pi}_{n+1} (\frac{1}{2} - \epsilon) < t+1 \\ \Rightarrow 0 < \quad \sum_{i \in K} \tilde{a}^i - \hat{\pi}_n (\frac{1}{2} - \epsilon) < 1. \end{aligned} \tag{2.11}$$

Finally, the condition $\hat{\pi}_0 < \hat{\pi} v^{n+3} < \hat{\pi}_0$ gives:

$$\begin{aligned} t < \frac{\hat{\pi}_n}{2} < t+1 \\ \Rightarrow \hat{\pi}_n = 2t+1. \end{aligned}$$

Since the disjunction $(\hat{\pi}, \hat{\pi}_0)$ is the same as the disjunction $(-\hat{\pi}, -\hat{\pi}_0 - 1)$, we assume without loss of generality that $\hat{\pi}_n \geq 0$. The condition $\hat{\pi}_n = 2t+1$ implies that $\hat{\pi}_n \geq 1, t \geq 0$. These, along with the conditions $K \neq \phi, M > 3$, and equation (2.10) imply that $\hat{\pi}_n = 1$. This along with equations (2.10, 2.11) gives,

$$\begin{aligned} \sum_{i \in K} \tilde{a}^i < \frac{1}{2} + \epsilon \quad \text{and} \\ \sum_{i \in K} \tilde{a}^i > \frac{1}{2} - \epsilon. \end{aligned}$$

These conditions, along with the choice of ϵ, imply respectively that $\sum_{i \in K} \tilde{a}^i \leq \frac{1}{2}$ and $\sum_{i \in K} \tilde{a}^i \geq \frac{1}{2}$. Therefore, $\sum_{i \in K} \tilde{a}^i = \frac{1}{2}$ and K is the desired subset of $\mathcal{S}$.

To complete the proof, we show that a description of $\mathcal{P}_s$ in the form (P_b) can be obtained in polynomial time from the finite list of points $v^1, v^2, \ldots, v^{n+2}$. Note that the convex hull of $v^1, v^2, \ldots, v^{n+2}$ is a simplex in $(n+1)$ dimensions (say Q), and can be expressed in form (P_b) in time polynomial in the size of the input as follows. If the point $v^{n+3} \in Q$, then $\mathcal{P}_s = Q$. Otherwise, delete from Q any such inequalities that are violated by v^{n+3}. Call this description $\mathcal{P}'$. Consider each of the $\frac{1}{2}(n+1)(n+2)$ hyperplanes passing through v^{n+3} and any n extreme points of Q. If all of the extreme points of Q lie on one side of this hyperplane, add this to the description $\mathcal{P}'$. Once all such hyperplanes are considered, the region $\mathcal{P}'$ is the same as $\mathcal{P}_s$. This process takes time polynomial in the size of the input and yields a description of $\mathcal{P}_s$ in form (P_b). The proof is now complete. □

The proof provided above is not sufficient to prove a similar result for the restriction of Problem 5 in which $\pi \in \{0,1\}^n$ because one of the components of the vector π in the proof above is restricted to the value of -1. However, the following proof shows that the problem remains $\mathcal{NP}$-complete even in the presence of this restriction. The proof uses a reduction of the ONE-IN-THREE-3SAT problem [Garey and Johnson, 1979; Schaefer, 1978], which is known to be $\mathcal{NP}$-complete, to this problem.

Problem 6 (ONE-IN-THREE-3SAT [Garey and Johnson, 1979])**.** *Given a set U of variables and a collection C of clauses over U such that each clause $c \in C$ has $|c| = 3$ and c does not contain a negated literal. Is there a truth assignment for U such that each clause in C has exactly one true literal?*

Problem 7 (Disjunctive infeasibility of binary programs using 0-1 hyperplanes)**.** *Given a mathematical program of the form (P_b), does there exist $\hat{\pi} \in \{0,1\}^n, \hat{\pi}_0 \in \mathbb{Z}$ such that the feasible region of each LP associated with the subproblems (2.1) created after branching on $(\hat{\pi}, \hat{\pi}_0)$ (with additional constraints $x \in [0,1]^n$) is empty?*

Proposition 2.3.4. *Problem 7 is $\mathcal{NP}$-complete.*

Proof. We reduce Problem 6 to Problem 7 as follows. Associate variables $\hat{\pi}_i, i = 1, 2, \ldots, n-1$ with each variable u_i in U (where $n = |U| + 1$). Let $\hat{\pi}_i = 1$ if u_i is assigned TRUE in a truth assignment and $\hat{\pi}_i = 0$ otherwise. Clearly, an instance of Problem 6 has a required truth assignment if and only if $\hat{\pi}$ satisfies the following constraints,

$$\sum_{\{i|u_i \in c\}} \hat{\pi}_i = 1, \quad \forall c \in C$$
$$\hat{\pi}_i \in \{0,1\}^n \tag{2.12}$$

Let $A^{\hat{\pi}}$ be the coefficient matrix associated with the above program (with elements $a_{ij} = 1$ if and only if clause i contains variable u_j, 0 otherwise). If $rank(A^{\hat{\pi}}) < rank(A^{\hat{\pi}}, \mathbf{1})$, then the system (2.12) is infeasible and there does not exist any truth assignment for Problem 6. Also, any such infeasibility can be detected in polynomial time by calculating the rank of the above matrices. Hence, it may be assumed that $rank(A^{\hat{\pi}}) = rank(A^{\hat{\pi}}, \mathbf{1})$. It may also be assumed that the rows of $A^{\hat{\pi}}$ are linearly independent. Otherwise, one may drop a redundant row from (2.12) (or equivalently, a redundant clause from Problem 6). Using these facts, one can assume without loss of generality that $|C| = rank(A^{\hat{\pi}}) = rank(A^{\hat{\pi}}, \mathbf{1}) \leq |U| = n$.

Consider the convex hull $\mathcal{P}_s$ of $m = |C| + 1$ points: $v^i, i = 1, 2, \ldots m \in \mathbb{R}^n$. Let the coordinate j, v^i_j, of each point v^i assume a value 0 if $a_{ij} = 0$ and a value $\frac{1}{2}$ if $a_{ij} = 1, i = 1, 2, \ldots, m-1, j = 1, 2, \ldots, n$. Let $v^i_n = 0, i = 1, 2, \ldots, m-1$. Let v^m be chosen such that $v^m_j = 0, j = 1, 2, \ldots, n-1, v^m_n = \frac{1}{2}$. There exists a $\hat{\pi} \in \{0,1\}^n$ such that $\mathcal{P}_s \subseteq \{x \in \mathbb{R}^n \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$ if and only if $v^i \in \{x \in \mathbb{R}^n \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}, i = 1, 2, \ldots, m$. This is true if and only if $(\hat{\pi}, \hat{\pi}_0)$ satisfy the following conditions

$$\pi_0 < \frac{1}{2}\sum_{u_j \in c} \hat{\pi}_j < \pi_0 + 1, \quad c \in C$$
$$\pi_0 < \quad \frac{1}{2}\hat{\pi}_n < \hat{\pi}_0 + 1,$$

or equivalently, if and only if $(\hat{\pi}, \hat{\pi}_0)$ satisfy the following conditions

$$\sum_{u_j \in c} \hat{\pi}_j = 2\pi_0 + 1, \quad c \in C$$

$$\hat{\pi}_n = 2\hat{\pi}_0 + 1.$$

Since $\hat{\pi} \in \{0,1\}^n$, the above conditions are satisfied if and only if $\hat{\pi}_0 = 0, \hat{\pi}_n = 1$ and $\hat{\pi}$ satisfies the system of equations (2.12). Hence, an instance of Problem 6 has a required truth assignment if and only if $\mathcal{P}_s \subseteq \{x \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$ for some $\hat{\pi} \in \{0,1\}^n$. Since it was assumed that the rows of $A^{\hat{\pi}}$ are linearly independent, the points v^i are also linearly independent. Hence, the dimension of $\mathcal{P}_s$ is exactly $m - 1 (= |C|)$. In order to obtain a description of $\mathcal{P}_s$ in the standard form (1.1), one has to find $|C|$ facets of $\mathcal{P}_s$. This can be done by making $|C|$ sets, each with $|C| - 1$ extreme points of $\mathcal{P}_s$ and finding a plane that passes through these. This can be done in time polynomial in the size of the input by solving $|C|$ systems of equations, each in $|C| - 1$ variables. These $|C|$ facets can be used to describe $\mathcal{P}_s$ in standard form (1.1). Thus, Problem 7 is $\mathcal{NP}$-complete. □

The complexity results for Problem 1 follow directly from those for Problem 3. In particular, Problem 1 is $\mathcal{NP}-$hard and remains so even when the restrictions described in Proposition 2.3.2 are applied and even for the case of binary programs.

2.4 Generating Split Inequalities

Recall from Section 1.4.2 that given a MIP of the form (1.1) and the associated polyhedron $\mathcal{P}$, we say that an inequality $(\alpha, \beta) \in \mathbb{R}^{n+1}$ is an elementary split inequality for $\mathcal{P}$ with respect to variables with indices $i = 1, 2, \ldots, d$ if both polyhedra $\{x \in \mathcal{P} \mid \pi x \leq \pi_0\}$ and $\{x \in \mathcal{P} \mid \pi x \geq \pi_0 + 1\}$ (and hence also the union of these polyhedra) are subsets of $\{x \in \mathbb{R}^n \mid \alpha x \geq \beta\}$ for some $\pi \in \mathbb{Z}^d \times 0^{n-d}$ and $\pi_0 \in \mathbb{Z}$. Since such a (π, π_0) is a valid disjunction for (1.1), such inequalities are also valid for the problem (1.1). In this section, we study the problem of selecting disjunctions for generating split inequalities. The way split inequalities are generated from a given disjunction is important in understanding this question. This is discussed next.

When a general disjunction $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}$ is given, one may generate many valid split inequalities from it. Any such valid inequality (α, β) must be valid for both the following linear systems.

$$\begin{array}{lcl} Ax \geq b & & Ax \geq b \\ \hat{\pi}x \leq \hat{\pi}_0 & \text{and} & \hat{\pi}x \geq \hat{\pi}_0 + 1 \\ x \in \mathbb{R}^n & & x \in \mathbb{R}^n. \end{array} \tag{2.13}$$

Using the theory of duality for linear programming (see Hadley [1961], for instance), there must exist multipliers u_L, $u_R \in \mathbb{R}^m_+$, $s_L, s_R \in \mathbb{R}_+$ such that the following conditions hold.

$$\begin{aligned} u_L A - s_L \hat{\pi} &= \alpha \\ u_L b - s_L b &\geq \beta \\ u_R A + s_R \hat{\pi} &= \alpha \\ u_R b + s_R b &\geq \beta \end{aligned} \tag{2.14}$$

Conversely, a split inequality $(\hat{\alpha}, \hat{\beta}) \in \mathbb{R}^{n+1}$ can be generated from a given disjunction $(\hat{\pi}, \hat{\pi}_0)$ if the above system (2.14) is feasible for $\alpha = \hat{\alpha}$, $\beta = \hat{\beta}$. One can now find the "best" split-inequality that can be generated from a disjunction $(\hat{\pi}, \hat{\pi}_0)$ by optimizing over the polyhedral set described by (2.14).

Keeping formulation (2.14) in view, we now describe two criteria for selecting disjunctions for generating split inequalities: the criterion of *maximum violation* that has been used extensively in the past and the criterion of *maximum bound improvement* that we previously used for generating branching disjunctions.

2.4.1 Maximum Violation

One criterion commonly used for generating valid inequalities is that of *maximum violation*. Given $\hat{x} \in \mathbb{R}^n$ and a valid inequality (α, β), the violation of $\hat{x}$ is defined as $\beta - \alpha\hat{x}$. The vector $\hat{x}$ usually arises as the solution of the LP relaxation of the MIP. This criterion is based on the intuition that a maximally violated inequality would be "farthest" from the optimal solution of the LP relaxation and hence should tighten it by the maximum possible amount. Moreover, if for a given vector, the maximum violation after optimizing over (2.14) turns out to be non-positive, then this is a proof that the point cannot be separated using the class of valid inequalities under consideration. The criterion of maximum violation is hence useful both practically and theoretically and is thus used extensively in the context of many classes of valid inequalities by, amongst others, Balas and Saxena [2007] for split inequalities, Fischetti and Lodi [2005] for C-G inequalities, Balas et al. [1996] for lift-and-project inequalities and Gu et al. [1998] for lifted-cover inequalities.

In the context of split inequalities, a split inequality $(\alpha, \beta) \in \mathbb{R}^{n+1}$ is equivalent to the inequality $(\lambda\alpha, \lambda\beta)$ for any $\lambda > 0$ and hence a maximally violated inequality is well-defined only under a suitable normalization of α and β. We now define the problem of finding a maximally violated elementary split inequality.

Problem 8. *Given a mathematical program of the form (1.1), a point $\tilde{x} \in \mathbb{R}^n$ and a normalization scheme for defining valid inequalities, find an elementary split inequality that violates $\tilde{x}$ by the maximum.*

When a general disjunction $(\hat{\pi}, \hat{\pi}_0)$ is provided, one can solve the following Cut Generation Linear Program (CGLP) to find the most violated inequality that may be generated from that disjunction.

$$\begin{aligned} \max \beta - \alpha \hat{x} \\ u_L A - s_L \hat{\pi} - \alpha = 0 \\ u_L b - s_L \hat{\pi}_0 - \beta \geq 0 \\ u_R A + s_R \hat{\pi} - \alpha = 0 \\ u_R b + s_R(\hat{\pi}_0 + 1) - \beta \geq 0 \\ u_L, u_R \in \mathbb{R}^m_+ \\ s_L, s_R \in \mathbb{R}_+ \\ \alpha \in \mathbb{R}^n, \beta \in \mathbb{R}. \end{aligned} \tag{2.15}$$

If the above program has solution with positive solution value, then one can scale the variables arbitrarily to obtain an unbounded solution value. Normalization constraints as mentioned above are hence required to generate an inequality that is usable.

Balas and Saxena [2007] propose a parametric MIP formulation that may be solved to obtain a general disjunction that will yield a maximally violated inequality. Thus, they use the criterion of selecting a disjunction that generates a maximally violated inequality.

They formulate this problem as the program

$$
\begin{aligned}
&\max \beta - \alpha\hat{x} \\
&u_L A - s_L\hat{\pi} - \alpha = 0 \\
&u_L b - s_L\hat{\pi}_0 - \beta \geq 0 \\
&u_R A + s_R\hat{\pi} - \alpha = 0 \\
&u_R b + s_R(\hat{\pi}_0 + 1) - \beta \geq 0 \\
&u_L, u_R \in \mathbb{R}^m_+, \alpha \in \mathbb{R}^n \\
&s_L, s_R \in \mathbb{R}_+, \beta \in \mathbb{R} \\
&\pi \in \mathbb{Z}^d \times \{0\}^{n-d}, \pi_0 \in \mathbb{Z},
\end{aligned}
\tag{2.16}
$$

and introduce a normalization constraint $s_L + s_R = 1$. This reduces the above problem to a parametric MIP with parameter $s_L \in [0, 1]$ and with $s_R = 1 - s_L$.

2.4.2 Maximum Bound Improvement

We now describe the criterion of selecting disjunctions that will yield inequalities that maximally improve the lower bound. Such a criterion has been described for selecting disjunctions for branching and may thus be useful for generating valid inequalities as well. In order to address the complexity of the problem of selecting such disjunctions, we first need to point out the difference between a split inequality of rank one and an elementary split inequality.

The elementary split closure of a MIP is the region formed by the intersection of $\mathcal{P}$ and all elementary split inequalities of $\mathcal{P}$ with respect to $x_i, i = 1, 2, \ldots, d$. Cook et al. [1990] showed that this closure is also a polyhedron. An inequality is said to have split-rank one if it is valid for the elementary split closure of $\mathcal{P}$ but not for $\mathcal{P}$. Thus all elementary split inequalities have split-rank at most one. Also, any inequality that is a convex combination of two elementary split inequalities has split-rank at most one. However, since two different disjunctions may have been used to generate the two elementary split inequalities thus combined, the convex combination of these inequalities may not necessarily be an elementary split inequality, even though its rank is one. As an example, consider the following system

$$\begin{aligned} x_1 &\le 0.8 \\ x_2 &\le 0.8 \\ x &\in \mathbb{Z}^2. \end{aligned} \tag{2.17}$$

The inequality $x_1 + x_2 \le 0$ has split-rank one with respect to (2.17) because it can be obtained as a convex combination of the elementary split inequalities $x_1 \le 0$ and $x_2 \le 0$. The inequality $x_1 + x_2 \le 0$ separates the points $(\frac{1}{2}, 0), (0, \frac{1}{2}), (\frac{1}{2}, \frac{1}{2})$, which are in the associated polyhedron $\mathcal{P}$, from the feasible region of (2.17). If this were an elementary split inequality for $\mathcal{P}$ generated using a disjunction $\pi x \le \pi_0 \vee \pi x \ge \pi_0 + 1$, then the disjunction (π, π_0) should also separate the points $(\frac{1}{2}, 0), (0, \frac{1}{2}), (\frac{1}{2}, \frac{1}{2})$. So, (π, π_0) should, at least, satisfy the following three constraints:

$$\begin{aligned} \pi_0 < {} & \frac{1}{2}\pi_1 < \pi_0 + 1 \\ \pi_0 < {} & \frac{1}{2}\pi_2 < \pi_0 + 1 \\ \pi_0 < {} & \frac{1}{2}(\pi_1 + \pi_2) < \pi_0 + 1 \\ & \pi_0, \pi_1, \pi_2 \in \mathbb{Z}. \end{aligned} \tag{2.18}$$

Since the system (2.18) is infeasible, there is no such disjunction and hence $x_1 + x_2 \le 0$ is not an elementary split inequality even though it is a convex combination of two such inequalities.

The above distinction between a rank one split inequality and an elementary split inequality is important to distinguish between the following two problems.

Problem 9. *Given a mathematical program of the form (1.1) and $K \in \mathbb{R}$, does there exist a single elementary split inequality for (1.1) such that the LP relaxation bound achieved after adding it is at least z_l?*

Problem 10. *Given a mathematical program of the form (1.1) and $K \in \mathbb{R}$, does there exist a split inequality of rank one for (1.1) such that the LP relaxation bound achieved after adding it is at least K?*

We will show in next section that Problem 9 can be answered yes if and only if there exists a general disjunction $(\hat{\pi}, \hat{\pi}_0)$ such that $\min\{cx \mid Ax \geq b, \hat{\pi}x \leq \hat{\pi}_0\} \geq K$ and $\min\{cx \mid Ax \geq b, \hat{\pi}x \geq \hat{\pi}_0 + 1\} \geq K$. Thus it directly relates to our problem of selecting a suitable disjunction. Also, Problem 10 that has been studied previously, seems to have little relation with Problem 9.

2.5 Complexity of Generating Split Inequalities

Caprara and Letchford [2003] showed that the problem of determining whether a given vector $x \in \mathbb{R}^n$ may be separated from the elementary split closure of $\mathcal{P}$ with respect to the variables $x_i,\ i = 1, 2, \ldots, d$ is $\mathcal{NP}$-complete. Thus, Problem 8 is $\mathcal{NP}$-hard. In order to answer Problem 10, we have to optimize over the elementary split closure. It thus follows from the equivalence of separation and optimization, that Problem 10 is $\mathcal{NP}$-hard.

However, this does not imply anything about the complexity of Problem 9 or even of the problem of showing that a given inequality is an elementary split inequality. We have already seen that even if an inequality has split rank one, it may not be an elementary split inequality. We now show that the complexity of these problems follows directly from Proposition 2.3.1.

Problem 11. *Given a mathematical program of the form (1.1), is a given inequality* (α, β) *an elementary split inequality for* $\mathcal{P}$ *with respect to the variables* $x_i, i = 1, 2, \ldots, d$*?*

Proposition 2.5.1. *Problem 11 is* $\mathcal{NP}$*-complete*

Proof. Consider the special case when $\alpha = 0, \beta = 1$. Then $\alpha x \geq \beta$ (or $0 \geq 1$) is a split inequality for $\mathcal{P}$ if and only if there exists a disjunction $(\hat{\pi}, \hat{\pi}_0)$ such that the associated LPs (2.1) are infeasible. By Proposition 2.3.1, the problem of finding such $(\hat{\pi}, \hat{\pi}_0)$ is $\mathcal{NP}$-complete. □

In contrast, consider the case of Chvátal-Gomory (C-G) inequalities. Given a pure integer program (1.1) with $d = n$, a C-G inequality for (1.1) is an inequality of the form

$$\alpha x \geq \lceil \beta \rceil, \tag{2.19}$$

where $\alpha \in \mathbb{Z}^n$ and $\alpha x \geq \beta$ is a valid inequality for the feasible region $\mathcal{P}$ of the LP relaxation. Eisenbrand [1999] showed that the problem of separating a given vector $x \in \mathbb{R}^n$ from the elementary C-G closure of (1.1) is $\mathcal{NP}$-complete. Hence, the problem of deciding whether a given inequality has C-G rank one or not is also, like for the case of split inequalities, $\mathcal{NP}$-complete. However, unlike split inequalities, we can decide whether a given inequality (say $\alpha x \geq \delta$, where $\alpha \in \mathbb{Z}^n, \delta \in \mathbb{Z}$) is an elementary C-G inequality or not in polynomial time by solving the LP

$$\begin{aligned} &\min \ \alpha x \\ &\quad Ax \geq b. \end{aligned} \tag{2.20}$$

Then, $\alpha x \geq \delta$ is an elementary C-G inequality for (1.1) if and only if the optimal solution to (2.20) is strictly greater than $\delta - 1$.

The above results seem somewhat surprising. On the one hand, the problem of deciding whether the rank of a given inequality is one lies in the complexity class $\mathcal{NP}$-complete for the case of both C-G inequalities and split inequalities. On the other hand, the problem of deciding whether a given inequality is an elementary C-G inequality lies in complexity class $\mathcal{P}$, while the same problem for a split inequality lies in complexity class $\mathcal{NP}$-complete. We now provide the proof of complexity of Problem 9.

Proposition 2.5.2. *Problem 9 is $\mathcal{NP}$-complete.*

Proof. Problem 9 can be shown to be equivalent to Problem 3. Suppose there is a disjunction, say $(\hat{\pi}, \hat{\pi}_0)$, such that the LP objective function value associated with each subproblem is at least K, i.e. $(\hat{\pi}, \hat{\pi}_0)$ is a solution to Problem 3. Then, $cx \geq K$ is a valid inequality for both $\mathcal{P} \cap \{x \mid \hat{\pi}x \leq \hat{\pi}_0\}$ and $\mathcal{P} \cap \{x \mid \hat{\pi}x \geq \hat{\pi}_0 + 1\}$. Therefore, $cx \geq K$ is a valid elementary split inequality that when added to the LP relaxation makes the objective function value at least K.

Conversely, suppose there exists an elementary split inequality, say (α, β), derived from a disjunction $(\hat{\pi}, \hat{\pi}_0)$, such that the LP bound for the polytope $\mathcal{P}^1 = \mathcal{P} \cap \{x \mid \alpha x \geq \beta\}$ is at least K. Then $cx \geq K$ is a valid inequality for $\mathcal{P}^1$. This means that $cx \geq K$ is a valid inequality for the two subsets of $\mathcal{P}^1 : \mathcal{P} \cup \{x \mid \hat{\pi}x \leq \hat{\pi}_0\}, \mathcal{P} \cup \{x \mid \hat{\pi}x \geq \hat{\pi}_0 + 1\}$ and hence branching on the disjunction $(\hat{\pi}, \hat{\pi}_0)$ will push the LP bound to at least K.

□

The equivalence of Problem 9 and Problem 3 has been mentioned previously by, among others, Karamanov and Cornuéjols [2007]. However, we still provide the proof in order to complete a proof for Proposition 2.5.2. It is also well known that one can always find an elementary split inequality (for instance, a Gomory mixed integer inequality) to separate a given basic feasible solution of the LP relaxation in time polynomial in the size of the input (see, for instance, Cornuéjols [2008]). However, the above problem of maximizing the lower bound by adding a valid split inequality remains $\mathcal{NP}$-hard even if an optimal basic feasible point of $\mathcal{P}$ is provided as an input because the desired inequality must separate the set of all points that have lower objective values regardless of if they are basic feasible solutions.

2.6 Disjunctions for Separating Two Points

In Section 2.3 we showed that Problem 3 is $\mathcal{NP}$-complete. However, a similar problem of proving the same result when the given polytope $\mathcal{P}$ is just a point is trivial. In order to identify the "simplest" polytopes for which the problem of identifying the required general disjunction is theoretically difficult, we look at the problem that is a natural extension of the single-point case. Given a line segment between two points $x^1, x^2 \in \mathbb{Q}^n \backslash \mathbb{Z}^n$, we would like to know how difficult is it to show that it does not contain any integer points and what is the minimum number of disjunctions required to prove this. These questions are answered in the following discussion. The consequences of these results for the problem of separation of points that lie on an edge of the feasible region of the LP relaxation from the split closure is also discussed.

Problem 12. *Given two distinct points $x^1, x^2 \in \mathbb{Q}^n$, let $\mathcal{T}$ be the convex hull of x^1, x^2, i.e., $\mathcal{T} = \{x \in \mathbb{R}^n \mid x = \lambda x^1 + (1-\lambda)x^2, \lambda \in [0,1]\}$. Does $\mathcal{T}$ contain a point $z \in \mathbb{Z}^n$?*

Proposition 2.6.1. *Let $x^1, x^2, \mathcal{T}$ be defined as in Problem 12.*

1. *$\mathcal{T}$ does not contain integer points if and only if there exists $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$ such that $\mathcal{T} \subset \{x \in \mathbb{R}^n \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$.*

2. *Problem 12 can be solved in time polynomial in size of the inputs.*

Proof. The proposition trivially holds true if at least one of $x^1, x^2 \in \mathbb{Z}^n$. Assume that $x^1 \notin \mathbb{Z}^n, x^2 \notin \mathbb{Z}^n$. Let $\mathcal{L}$ be the set of points contained in the line that passes through x^1, x^2. Then $\mathcal{L} = \{x \in \mathbb{R}^n \mid x = x^1 + \gamma(x^2 - x^1), \gamma \in \mathbb{R}\}$. If $\mathcal{L}$ does not contain any $z \in \mathbb{Z}^n$, then neither does $\mathcal{T}$. Further, one can find in time polynomial in size of input, $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$ such that $\mathcal{L} \subset \{x \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$ [Nemhauser and Wolsey, 1988, pg. 191].

If $\mathcal{L}$ contains some integer points, then let $z^1 = x^1 + \gamma_1(x^1 - x^2)$, $z^2 = x^2 + \gamma_2(x^2 - x^1)$, with the condition that $\gamma_1, \gamma_2 \geq 0$ and that they are the smallest such values of γ_1 and γ_2. These conditions mean that z^1 (respectively, z^2) is an integer point on $\mathcal{L}$ that is closest to x^1 (x^2) in the direction away from x^2 (x^1). Now consider the system of Diophantine equations

$$\begin{aligned} \pi z^1 - \pi_0 &= 0 \\ \pi z^2 - \pi_0 &= 1 \\ \pi \in \mathbb{Z}^n, \pi_0 &\in \mathbb{Z}^{n+1}. \end{aligned} \tag{2.21}$$

We propose that there exists a point $z \in \mathcal{T} \cap \mathbb{Z}^n$ if and only if the system (2.21) is infeasible, i.e., if and only if z^1, z^2 lie in two different subsets associated with some valid disjunction (π, π_0). This is proved as follows. Clearly, if $z \in \mathcal{T} \cap \mathbb{Z}^n$, then there exists a $\beta \in (0, 1)$ such that $z = \beta z^1 + (1 - \beta) z^2$. Then, $\beta \pi z^1 - \beta \pi_0 + (1 - \beta)\pi z^2 - (1 - \beta)\pi_0 = \pi z - \pi_0 \in \mathbb{Z}$ but $0\beta + (1 - \beta) \notin \mathbb{Z}$. Hence, the system (2.21) is not feasible. On the other hand if system (2.21) is not feasible then there exist [Nemhauser and Wolsey, 1988, pg. 191] $\beta_1, \beta_2 \in \mathbb{R}$ such that

$$\beta_1 z^1 + \beta_2 z^2 \in \mathbb{Z}^n, \tag{2.22}$$

$$-\beta_1 - \beta_2 \in \mathbb{Z} \tag{2.23}$$

$$0\beta_1 + \beta_2 \notin \mathbb{Z} \tag{2.24}$$

From relation (2.22) and relation (2.23), we get $\beta_2(z^2 - z^1) \in \mathbb{Z}^n$. Let $\beta_2 = f_2 + \lfloor \beta_2 \rfloor$.

$f_2 \in (0,1)$ from (2.24). It now follows that

$$\begin{aligned} &\beta_2(z^2 - z^1) \in \mathbb{Z}^n \\ \Rightarrow &(f_2 + \lfloor \beta_2 \rfloor)(z^2 - z^1) \in \mathbb{Z}^n \\ \Rightarrow &f_2(z^2 - z^1) \in \mathbb{Z}^n \\ \Rightarrow &f_2(z^2 - z^1) + z^1 \in \mathbb{Z}^n \\ \Rightarrow &f_2 z^2 + (1 - f_2) z^1 \in \mathbb{Z}^n. \end{aligned}$$

The last relation implies that when the system (2.21) is infeasible, there exists a $z \in \mathcal{T} \cup \mathbb{Z}^n$. This completes the proof of part 1.

In order to prove part 2, when $\mathcal{L}$ has some integer points on it, we only need to show that one can find z^1, z^2 used in the proof above in time polynomial in the size of inputs. The problem of finding an integer point on $\mathcal{L}$ can be written as a system of Diophantine equations. Let z^0 be such a point on $\mathcal{L}$. All integer points on $\mathcal{L}$ then satisfy

$$z = z^0 + C_2 \zeta, \quad \zeta \in \mathbb{Z},$$

where, $C_2 \in \mathbb{Z}^n$. Both z^0 and C_2 can be found in time polynomial in size of the system of Diophantine equations, which are in turn polynomial in size of the input for Problem 12. Now one can do binary search over ζ to find the nearest integer point to x^1 away from x^2 on $\mathcal{L}$. Similarly, one can find z^2. This completes the proof for part 2 of the proposition. □

The proof above shows that given a line segment $\mathcal{T}$ between two points $x^1, x^2 \in \mathbb{Q}^n$, there can only be two cases: either (a) there is an integer point in $\mathcal{T}$ as in figure 2.1(a), or (b) $\mathcal{T}$ can be proven infeasible using a single disjunction as shown in figure 2.1(b). Further, it can be determined in time polynomial in size of inputs, which of these two cases holds. These results can be applied to the problem of finding split inequalities that separate points on an edge of the LP relaxation of a MIP from its elementary split closure.

Consider the problem

Problem 13. *Given a MIP with the feasible region of LP relaxation $\mathcal{P}$, two adjacent extreme points v^1, v^2 of $\mathcal{P}$ and a point $\hat{x} \in \mathbb{Q}^n$ on the edge joining v^1, v^2, find an elementary split inequality that separates $\hat{x}$ from the split closure of $\mathcal{P}$ or show that none exists.*

Proposition 2.6.2. *Problem (13) can be solved in time polynomial in size of input.*

Proof. We prove the result for the case when the given MIP is a pure integer program as it is easy to extend the result to the mixed case. If $\hat{x} \in \mathbb{Z}^n$ then the answer to Problem 13 is trivially no. Otherwise, consider the line segments $\mathcal{T}^1$ joining $\hat{x}$ and v^1 and $\mathcal{T}^2$ joining x and v^2. There are two cases

If both $\mathcal{T}^1$ and $\mathcal{T}^2$ contain integer points, then $\hat{x}$ is in the convex hull of those two points and hence can not be separated from the elementary split closure of $\mathcal{P}$.

If either of the two segments, say $\mathcal{T}^1$, does not contain an integer point, then using Proposition 2.6.1, $\hat{x}, v^1 \in \{x \in \mathbb{R}^n \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$ for some $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^{n+1}$. Consider the two polyhedra: $\mathcal{P}^1 = \mathcal{P} \cap \{x \in \mathbb{R}^n \mid \hat{\pi}x \leq \hat{\pi}_0\}$ and $\mathcal{P}^2 = \mathcal{P} \cap \{x \in \mathbb{R}^n \mid \hat{\pi}x \geq \hat{\pi}_0 + 1\}$. We claim that $\mathcal{T}^1 \cap conv(\mathcal{P}^1 \cup \mathcal{P}^2) = \phi$. Clearly $\mathcal{T}^1 \cap (\mathcal{P}^1 \cup \mathcal{P}^2) = \phi$ and $conv(\mathcal{P}^1 \cup \mathcal{P}^2) \subseteq \mathcal{P}$ is also a polyhedron. If our claim were false, then any point on $\mathcal{T}^1$ must be convex combination of two points in $\mathcal{P}^1 \cup \mathcal{P}^2$. This can not hold since $\mathcal{T}^1$ is part of an edge and its extreme point v^1 is not in $\mathcal{P}^1 \cup \mathcal{P}^2$. So our claim must be true. Using Proposition 2.6.1, one can find the required disjunction $(\hat{\pi}, \hat{\pi}_0)$ in time polynomial in size of input. Now we can use an approach similar to that used by Caprara and Letchford [2003] and by Balas and Saxena [2007] to find an elementary split inequality $(\hat{\alpha}, \hat{\beta})$ that separates $\hat{x}$ from the elementary split closure. Consider the two systems

$$\begin{aligned} z_L^* = \min\ & \hat{\alpha}x \\ \text{s.t.}\quad & Ax \geq b \\ & \hat{\pi}x \leq \hat{\pi}_0 \end{aligned} \qquad \text{and} \qquad \begin{aligned} z_R^* = \min\ & \hat{\alpha}x \\ \text{s.t.}\quad & Ax \geq b \\ & \hat{\pi}x \geq \hat{\pi}_0 + 1. \end{aligned} \tag{2.25}$$

$(\hat{\alpha}, \hat{\beta})$ is a required inequality if and only if $z_L^* \geq \hat{\beta}$ and $z_R^* \geq \hat{\beta}$. The dual of these programs can be written as

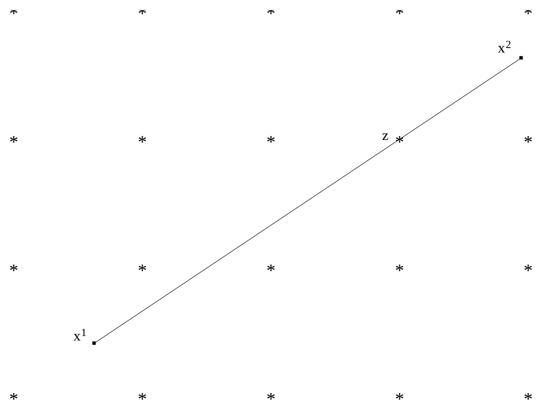
$$\begin{aligned} z_L^* = \max u_L b - s_L \hat{\pi}_0 \\ \text{s.t.} \quad u_L A - s_L \hat{\pi} = \hat{\alpha} \\ u_L, s_L \geq 0 \end{aligned} \qquad \text{and} \qquad \begin{aligned} z_R^* = \max u_R b + s_R(\hat{\pi}_0 + 1) \\ \text{s.t.} \quad u_R A + s_R \hat{\pi} = \hat{\alpha} \\ u_R, s_R \geq 0 \end{aligned} \tag{2.26}$$

Thus, $(\hat{\alpha}, \hat{\beta})$ is a required inequality if and only if the following system is feasible

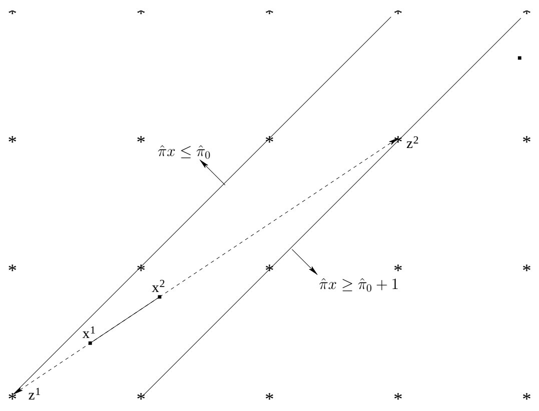
$$\begin{aligned} u_L b - s_L \hat{\pi}_0 \geq \hat{\beta} \\ u_R b + s_R(\hat{\pi}_0 + 1) \geq \hat{\beta} \\ u_L A - s_L \hat{\pi} = \hat{\alpha} \\ u_R A + s_R \hat{\pi} = \hat{\alpha} \\ u_L, s_L, u_R, s_R \geq 0 \end{aligned} \tag{2.27}$$

$\hat{\alpha}, \hat{\beta}$ can be scaled arbitrarily without affecting the inequality $\hat{\alpha} x \geq \hat{\beta}$. Thus we are free to impose a constraint that makes the violation of $\hat{x}, \beta - \hat{\alpha}\hat{x} = 1$. So $\hat{\alpha}, \hat{\beta}$ separates $\hat{x}$ from the elementary split closure if and only if $\alpha = \hat{\alpha}, \beta = \hat{\beta}$ in some feasible solution of the linear system

$$\begin{aligned} u_L b - s_L \hat{\pi}_0 \geq \beta \\ u_R b + s_R(\hat{\pi}_0 + 1) \geq \beta \\ u_L A - s_L \hat{\pi} = \alpha \\ u_R A + s_R \hat{\pi} = \alpha \\ \beta - \alpha \hat{x} = 1 \\ u_L, s_L, u_R, s_R \geq 0 \end{aligned} \tag{2.28}$$



(a) When $\mathcal{T}$ has an integer point, there are no possible $(\hat{\pi}, \hat{\pi}_0)$.


$\hat{\pi}x \leq \hat{\pi}_0$
z²
$\hat{\pi}x \geq \hat{\pi}_0 + 1$
x²
x¹
z¹

(b) When $\mathcal{T}$ has no integer points, one can always find $(\hat{\pi}, \hat{\pi}_0)$.

Figure 2.1: Illustration of two cases described in Proposition 2.6.1 that are possible for a line segment $\mathcal{T}$ between two points x^1, x^2.

Thus, the required inequality can be found in time polynomial in size of input. This completes our proof. □

Given a MIP of the form (1.1) and a point $\hat{x}$ that lies on an edge of the feasible region $\mathcal{P}$ of the LP relaxation, one can find in time polynomial in size of the input, the extreme points v^1, v^2 that are also the endpoints of an edge (see, for instance, Hadley [1961], pg. 80). Proposition 2.6.2 can thus be reduced to the following corollary.

Corollary 2.6.1. *Given a MIP in standard form (1.1), and a point $\hat{x}$ that lies on an edge of the feasible region of the LP relaxation, then one can either find in time polynomial in size of input, an elementary split inequality that violates $\hat{x}$ or show that no such inequalities exist.*

Corollary 2.6.2. *Given a MIP in standard form (1.1), if an edge of the feasible region $\mathcal{P}$ of the LP relaxation does not contain an integer feasible point, then the elementary split closure of the problem does not contain any points of that edge.*

2.7 Conclusions

In this chapter, we showed that the problem of selecting a general branching disjunction so that the LP relaxation associated with each subproblem becomes infeasible is $\mathcal{NP}$-complete. This leads to two important results—that the problem of selecting a general branching disjunction that maximizes the bound improvement of a given MIP is $\mathcal{NP}$-hard, and that the problem of deciding whether a given inequality is an elementary split inequality is $\mathcal{NP}$-complete. We also showed that the former problem remains $\mathcal{NP}$-hard even when several natural restrictions are imposed on the disjunctions or when all integer-constrained variables in the MIP are binary. We further showed that some of these problems can be solved in time polynomial in size of input if the polytope under consideration is just a line segment joining two points. Even though we answered some questions about computational complexity of finding desired disjunctions, there are several other related questions that are still open. These are described in Chapter 5.

Chapter 3

Computational Methods for Branching

3.1 Introduction

In Chapter 2, we formulated the problem of selecting a disjunction for branching as a MIP. In this chapter, we first describe the computational experiments performed to observe the effects of using the disjunctions obtained by solving this formulation using a standard MIP solver. Our experiments show that even though the size of the branch-and-bound tree is dramatically reduced when such disjunctions are used, the time taken to find these disjunctions using a stand-alone MIP solver is prohibitively high. In Section 3.2, we experiment with different settings to control the size of and the strategy for solving such MIPs and report on the effects of changing different parameters on the observed performance. In Section 3.3, we restrict ourselves to the study of branching on disjunctions with two variables only. We compare the performance of finding a two-variable disjunction by explicitly solving a MIP formulation against the case when it is obtained by enumeration. We then enhance the enumeration scheme such that the number of candidates that need to be enumerated is reduced by orders of magnitude. In Section 3.4, we adapt this enumeration technique for the case of strong branching on variables and demonstrate substantial improvement for this case as well. We first start with a small example that demonstrates the importance of the choice disjunctions for branching.

Consider the following problem:

$$\begin{aligned}
&\min 2x_1 + 2x_2 + 2x_3 - 3x_4 \\
&\text{s.t.} \\
&2x_1 + x_2 + x_3 - 2x_4 - 2r_1 = 0.1 \\
&x_1 + 2x_2 + x_3 - 2x_4 - 2r_2 = 0.1 \\
&x_1 + x_2 + 2x_3 - 2x_4 - 2r_3 = 0.1 \\
&2x_1 + 2x_2 + 2x_3 \geq 0.1 \\
&0 <= r_1 <= 0.90 \\
&0 <= r_2 <= 0.90 \\
&0 <= r_3 <= 0.90 \\
&x_1, x_2, x_3, x_4 \in \mathbb{Z}_+.
\end{aligned} \tag{3.1}$$

The above instance (3.1) is infeasible even when the constraint $2x_1 + 2x_2 + 2x_3 \geq 0.1$ is dropped. This constraint is added to confuse existing preprocessing routines and prevent them from detecting infeasibility easily. Although seemingly small and simple, the above instance is not able to be solved in reasonable time by most commercial and open source solvers available. For instance, ILOG CPLEX-10.2, with default settings, running on a machine with 2.4GHz Intel Xeon, 512KB cache and 4GB RAM, is unable to solve it even after running for 600 seconds and solving more than nine million subproblems. Similar behaviour is observed with default settings of Xpress-Optimizer-v18.10.00, SYMPHONY-5.1, CBC-2.1 and SCIP-1.0.

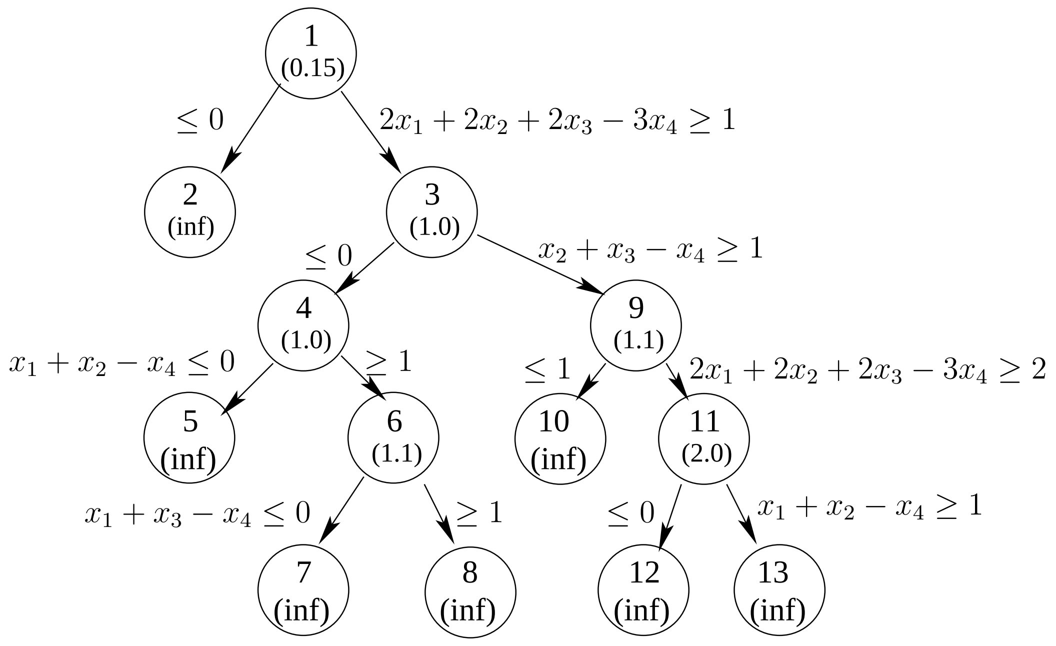


Figure 3.1: A branch and bound tree to prove the infeasibility of problem (3.1). A number in parentheses denotes the optimal value of the associated LP relaxation. (inf) denotes that the associated LP is infeasible.

The same instance can be shown to be infeasible using only thirteen subproblems in a branch-and-bound tree, without any use of advanced techniques of a generic branch-and-cut framework like generation of valid inequalities, preprocessing, probing and other heuristics. Such a branch-and-bound tree is described in Figure 3.1. The branching disjunctions are chosen to push the objective value of the LP relaxations as high as possible or to make them infeasible. This problem can be solved by the latest versions of both CPLEX (version-11.1) and SCIP (version-1.1) but one can still add constraints and modify the objective function to generate a behavior similar to one observed above. This example clearly shows that when useful disjunctions are used for either branching or for generating valid inequalities, the performance of branch-and-cut algorithm is improved dramatically.

3.2 Solving MIP Formulations

In order to test the effect of selecting branching disjunctions using the formulations presented in the previous chapter, we performed a series of experiments using ILOG CPLEX 10.2 with the default method for selecting branching disjunctions replaced by the ones previously described. Since our goal was only to discern the effectiveness of employing the disjunctions and not to test the efficiency of the method for determining them, our measure of effectiveness was reduction in total number of subproblems required to solve each instance. Thus, we ignored the time required to execute the algorithm, which was substantial in all cases. We used the criterion of maximizing the lower bound obtained after branching on the given disjunction. The formulation that was used for this purpose is derived from the formulation (2.4). A disjunction $(\hat{\pi}, \hat{\pi}_0)$ will result in a lower bound greater than a given value, say K if and only if, the following systems are both infeasible

$$\begin{aligned} Ax &\geq b \\ cx &\leq K \\ \hat{\pi}x &\leq \hat{\pi}_0 \end{aligned} \qquad \text{and} \qquad \begin{aligned} Ax &\geq b \\ cx &\leq K \\ \hat{\pi}x &\geq \hat{\pi}_0 + 1 \end{aligned} \tag{3.2}$$

It follows from the approach used in Section 2.2.1 that $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\} \times \mathbb{Z}^1$ is the required disjunction if and only if $\pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$ in a solution of the system

$$\begin{aligned} pA - s_L c - \pi &= 0 \\ qA - s_R c + \pi &= 0 \\ pb - s_L K - \pi_0 &\geq \delta \\ qb - s_R K + \pi_0 &\geq -1 + \delta \\ p, q, s_L, s_R &\geq 0 \\ (\pi, \pi_0) &\in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1, \end{aligned} \tag{3.3}$$

Table 3.1: List of 30 instances used in formulation-based experiments.

10teams	gt2	fiber	mod008	pp08aCUTS	ran13x13
aflow30a	harp2	flugpl	neos6	qnet1	rout
bell3a	khb05250	gen	nug08	qnet1_o	stein45
blend2	l152lav	gesa2	nw04	ran10x26	vpm1
egout	lseu	gesa2_o	p0548	ran12x21	vpm2

for some $\delta > 0$.

The test set for this experiment has only 30 representative instances from our original set. This set was chosen in order to complete experiments in reasonable time. Table 3.1 lists the names of these instances. The branching disjunctions were imposed using the callback functions provided with the CPLEX callable library. In all experiments, the best known objective function value was provided as an upper bound to the solver to ensure that the solution procedure was not affected by the order in which subproblems were solved or other extraneous factors related to improvement in the upper bound.

In the first experiment, a pure branch-and-bound procedure was used—other advanced techniques such as cutting planes, heuristics and probing were disabled. This allowed us to observe the effects of branching in isolation from the mitigating effects of applying these additional techniques. In the first experiment, a sequence of MIPs of the form (3.3) were solved to determine the disjunction yielding the maximum increase in lower bound. During initial testing, we concluded that optimizing over the entire set of general branching disjunctions was too time-consuming, as the MIPs (3.3) were sometimes extremely difficult to solve. We therefore imposed the following limitations for all tests.

1. π was restricted to the set $\{-M, -M+1, \ldots, M\}^n$. $M = 1$ was used in the first experiment and higher values were tried in other experiments.

2. Each π_i was replaced with two non negative variables substituting $\pi_i = \pi_i^+ - \pi_i^-, \pi_i^+, \pi_i^- \in [0, M]$. Such a transformation was used in order to make it easier for the solver to find heuristic solutions to the MIP formulation.

3. The constraint $\sum_{i=1}^{n} |\pi_i| \leq k$ was introduced to further restrict the search space. k was set to 2, 5, 10, 15 and 20 in different experiments.

4. A time limit of t seconds was imposed for solving any one MIP for selecting a branching disjunction. Additionally, a limit of $8t$ seconds was imposed on the time allowed to be spent in total on selecting any single branching disjunction. In the first experiment, t was set to 1000. Values of 50 and 100 were used in later experiments.

5. A total time limit of 20 hours was imposed for solving each instance. After 18 hours, only variable disjunctions were considered so that the problem could be solved to completion in the remaining two hours.

In case the search for a branching disjunction failed (because of time limits or because no solution was found), branching was carried out by considering variable disjunctions. Since it was not known how the selection rule of CPLEX works, the LP relaxations of the subproblem resulting from the imposition of each candidate variable disjunction were solved explicitly in order to determine the optimal variable disjunction according to the criteria of maximum increase in lower bound. In cases where it was found that there was no variable disjunction whose imposition resulted in an increase in the lower bound, the default variable branching scheme of CPLEX was invoked. The number of subproblems solved when branching on general disjunctions was compared against that when branching only on variable disjunctions.

The number of subproblems generated during solution of each instance in the first experiment is shown in Table 3.2. N_k denotes the number of subproblems created when the search was restricted by addition of the constraint $\sum_{i=1}^{n} |\pi_i| \leq k$. Thus, N_1 denotes the number of subproblems when branching was done using only variable disjunctions (by selecting a variable disjunction after solving the resulting LP relaxations explicitly, as described above). The value r_k is defined to be $\frac{N_1}{N_k}$. Even though the experiments for Table 3.2 were carried out with 91 instances, only results for the 30 selected for further investigation are reported, since other instances showed similar results.

For all remaining experiments, the performance profiles of Dolan and Moré [2002] were used to display compactly, the results comparing number of subproblems solved in various experiments. A point (α, β) in such a plot indicates that a fraction β of all instances required less than α times the number of subproblems required in the experiment achieving the lowest total overall. Figure 3.2 shows a performance profile for the data in Table 3.2.

In the next two experiments, the time limit t imposed on the solution of each MIP was reduced to 100 seconds and 50 seconds, respectively. This was done to determine whether good branching disjunctions could still be found in a shorter amount of time. Figures 3.3 and 3.4 show the performance profile when t was fixed and k was varied.

The experiments described so far show that branching on disjunctions that maximize the subsequent lower bound increase does in fact lead to a significant reduction in the number of subproblems required to be solved. In general, the number of subproblems is also reduced when the set of disjunctions considered is larger (i.e., the number of non-zeros allowed in the vector π is increased).

Figures 3.5(a)-3.5(e) show the effect of time spent in selecting a branching disjunction when k is fixed. In general, when k is small, additional time spent selecting a disjunction pays a bigger dividend than when k is large. Figure 3.5(d) shows that when $k = 15$ the number of subproblems solved does not vary much as t is increased. When k is set to 20, the performance with $t = 50$ is nearly equivalent to that with $t = 1000$. One possible explanation is that for large values of k, if a feasible solution to the branching disjunction selection problem is not found quickly, then it is unlikely that a solution will be found even after substantial additional search time. Thus, even though branching on disjunctions that increase the lower bound appears promising, the problem of selecting disjunctions becomes increasingly difficult with the number of nonzero coefficients that are allowed in the description. This seems to be the case for the instance vpm1 in particular (see Table 3.2) —when k is changed from 15 to 20, the number of subproblems goes up

Table 3.2: Number of nodes (N_i) in branch and bound tree and the ratio $r_i = \frac{N_1}{N_i}$ for selected instances when $t = 1000$ seconds. The criterion for selecting the branching disjunction is to maximize the lower bound.

Instance	N_1	N_2	r_2	N_5	r_5	N_{10}	r_{10}	N_{15}	r_{15}	N_{20}	r_{20}
10teams	115	106	1.08	28	4.11	18	6.39	12	9.58	12	9.58
aflow30a	36634	19408	1.89	19485	1.88	20388	1.8	24112	1.52	20271	1.81
bell3a	16387	14377	1.14	8771	1.87	588	27.87	259	63.27	259	63.27
blend2	304	251	1.21	231	1.32	188	1.62	165	1.84	209	1.45
egout	2246	1044	2.15	554	4.05	572	3.93	676	3.32	558	4.03
fiber	18412	7676	2.4	7612	2.42	3039	6.06	3358	5.48	3324	5.54
flugpl	394	176	2.24	6	65.67	10	39.4	6	65.67	6	65.67
gen	100	100	1	100	1	100	1	100	1	100	1
gesa2	33526	24433	1.37	21664	1.55	21849	1.53	21849	1.53	21778	1.54
gesa2_o	98550	24777	3.98	24435	4.03	24661	4	24661	4	24661	4
gt2	340	10	34	10	34	12	28.33	10	34	12	28.33
harp2	432010	157377	2.75	174656	2.47	183306	2.36	174454	2.48	179130	2.41
khb05250	738	606	1.22	594	1.24	588	1.26	614	1.2	618	1.19
l152lav	60	40	1.5	32	1.88	28	2.14	34	1.76	30	2
lseu	4058	2365	1.72	226	17.96	78	52.03	58	69.97	58	69.97
mod008	2840	1678	1.69	296	9.59	102	27.84	68	41.76	52	54.62
neos6	5989	2131	2.81	2131	2.81	2131	2.81	2131	2.81	2131	2.81
nug08	14	6	2.33	4	3.5	6	2.33	6	2.33	5	2.8
nw04	30	24	1.25	16	1.88	12	2.5	12	2.5	12	2.5
p0548	1050	500	2.1	466	2.25	566	1.86	565	1.86	565	1.86
pp08aCUTS	1301300	486340	2.68	147271	8.84	166943	7.79	168905	7.7	231527	5.62
qnet1	42	30	1.4	24	1.75	20	2.1	22	1.91	18	2.33
qnet1_o	154	126	1.22	94	1.64	77	2	80	1.93	92	1.67
ran10x26	68449	34693	1.97	23309	2.94	24716	2.77	23704	2.89	21520	3.18
ran12x21	494558	280551	1.76	219967	2.25	208948	2.37	225980	2.19	212910	2.32
ran13x13	124716	87495	1.43	74699	1.67	57825	2.16	66008	1.89	58789	2.12
rout	219322	79399	2.76	65201	3.36	61806	3.55	61226	3.58	57673	3.8
stein45	31086	31177	1	21238	1.46	20594	1.51	20601	1.51	20601	1.51
vpm1	263111	40952	6.42	145	1814.56	32	8222.22	20	13155.55	5929	44.38
vpm2	273994	145152	1.89	77504	3.54	67014	4.09	69515	3.94	73687	3.72

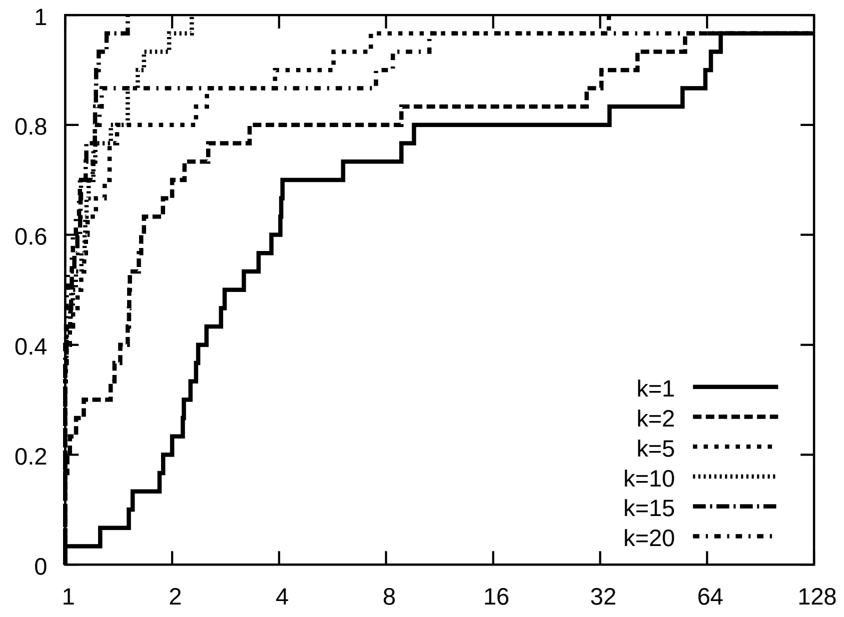


Figure 3.2: Performance profile for number of subproblems when $t = 1000s$ and k is varied.

from 20 to 5929, presumably because the branching disjunction selection problem becomes so difficult that only a few effective disjunctions are found within the time limit.

In the next experiment, cutting planes were enabled to see how the branching disjunction selection method would perform in a branch-and-cut algorithm. In general, introduction of cutting plane generation should be expected to reduce the total number of subproblems. The default settings of CPLEX were used for cut generation, with the exception that MIR, Gomory, and flow (cover and path) cuts were disabled because the presence of these cuts caused numerical difficulties while solving some of the associated branching disjunction selection problems. Figure 3.6(a) shows the effect of adding cuts when $t = 100$ seconds and k has values 1, 2, and 5. It shows that enabling cuts increases the performance of the solver significantly, even when branching on general disjunctions is used. Figure 3.6(b) shows how the performance varies when cuts are enabled and k is varied from 1 to 20. Figure 3.3 shows that, in the absence of cuts, around 80% of instances

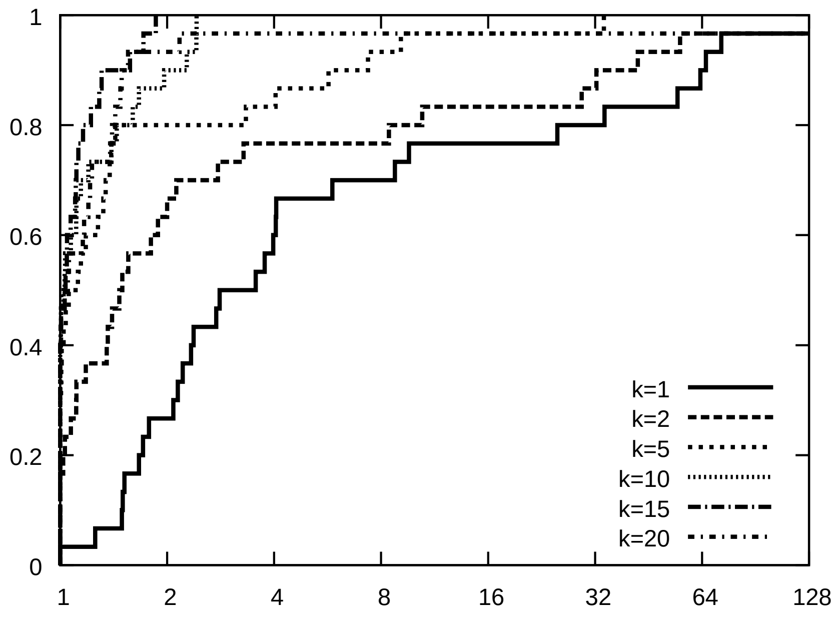


Figure 3.3: Performance profile for number of subproblems when $t = 100s$ and k is varied.

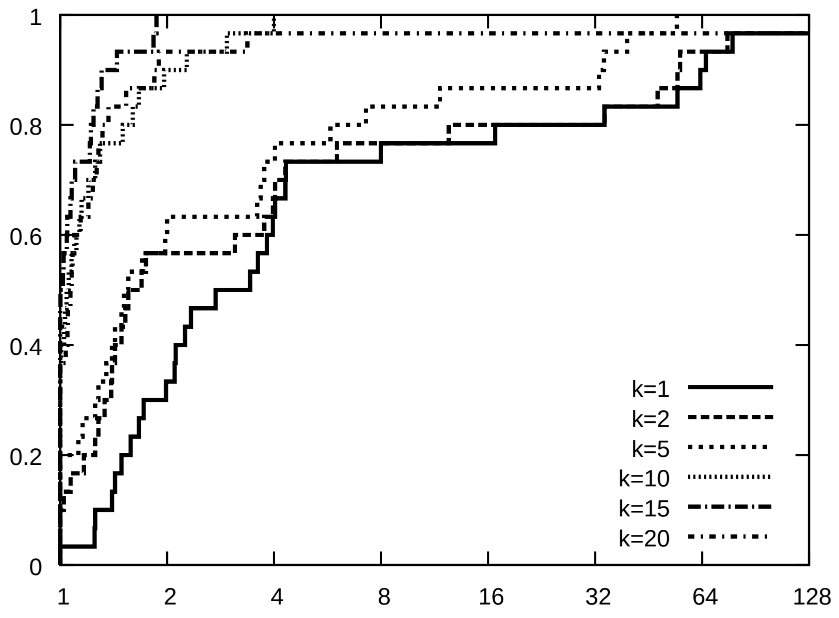
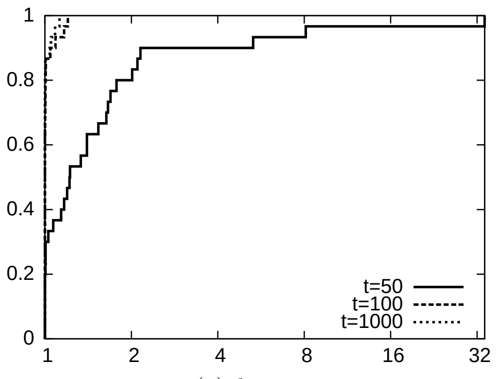
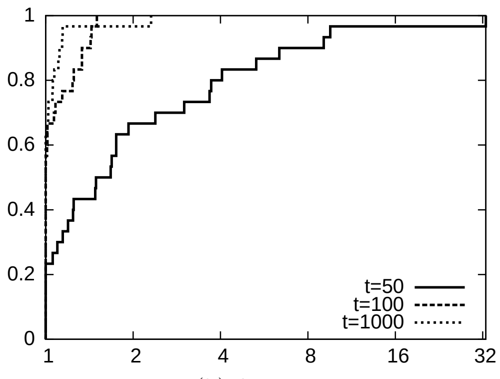


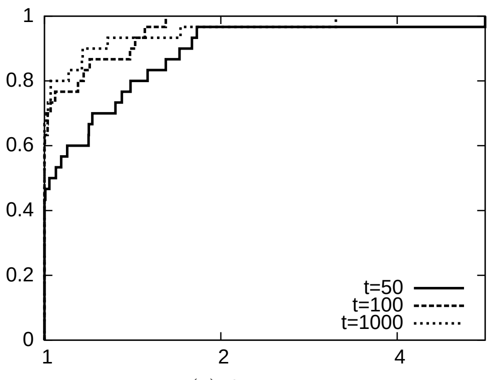
Figure 3.4: Performance profile for number of subproblems when $t = 50s$ and k is varied.



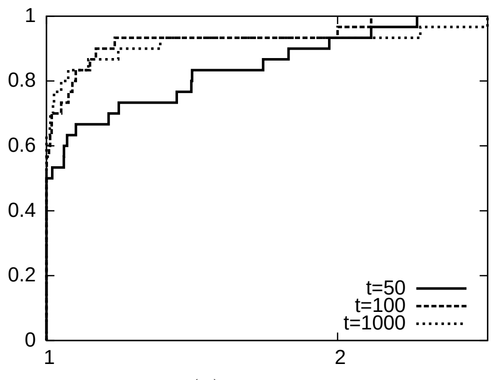
(a) $k = 2$



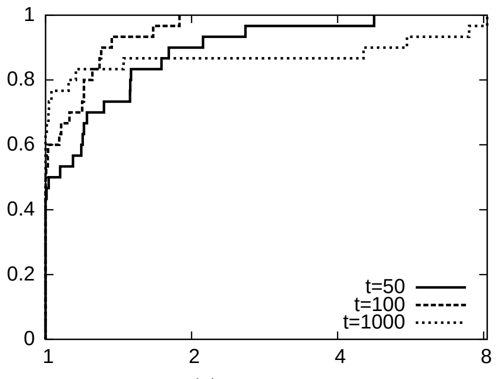
(b) $k = 5$



(c) $k = 10$



(d) $k = 15$



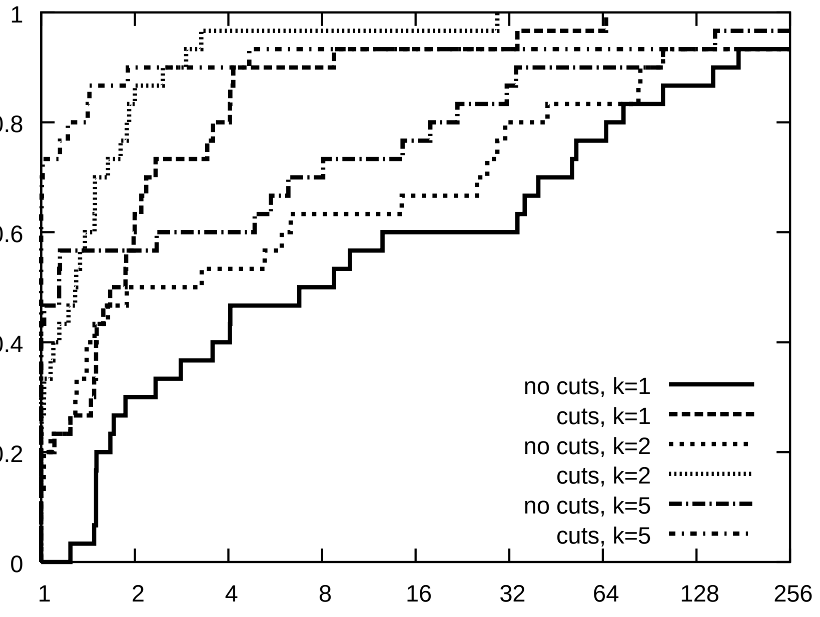
(e) $k = 20$

Figure 3.5: Performance profile for number of subproblems when k is fixed and t (in seconds) is varied.

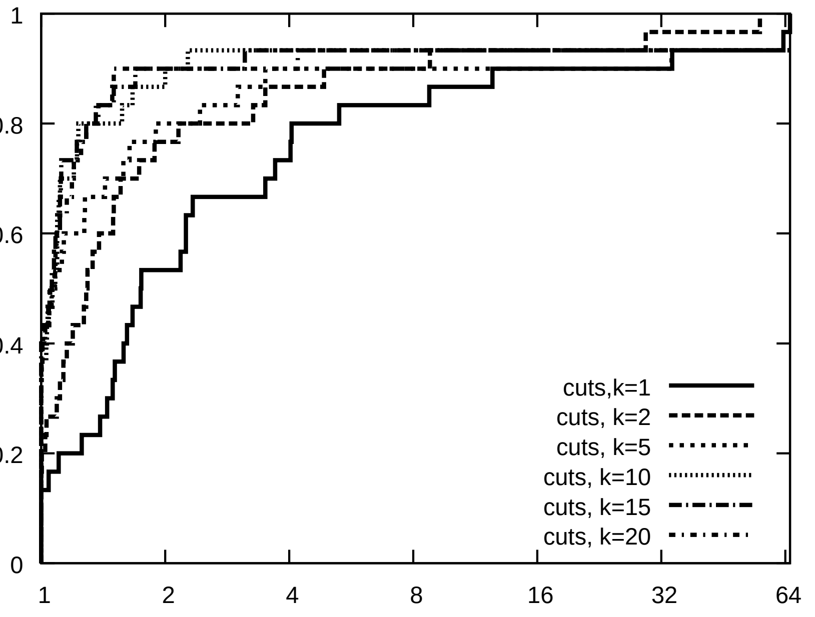
required at least half as many subproblems when branching on general disjunctions. When the cuts were enabled, this fraction dropped to 50%. So the effect of branching on general disjunctions is substantial even when the cuts are enabled, though it is not as dramatic.

To see the effect of increasing M, we performed one experiment with $M = 10, k = 15, t = 100$. Figure 3.7 shows a comparison of performance on this test against the others. The performance seems to be slightly worse than when $M = 1$. However, it could not be established whether this was due to larger coefficients in some of the disjunctions or because of the increased difficulty of the MIPs used to identify the disjunction. A similar experiment was carried with $M = 10, k = 15, t = 1000$ to see the effects for the case when more time was spent in finding disjunctions with $M > 1$. Figure 3.8 shows that there are no considerable effects from spending more time or changing M. These experiments seem to suggest that k is probably a more important parameter than either t or M.

Finally, we experimented with selecting a branching disjunction along a "thin" direction by solving the formulation (2.7). Additional constraints, as described for the criteria of maximizing lower bound above, were also added. Figure 3.9 compares the number of subproblems solved when branching on "thin" directions with other experiments. The performance is seen to be comparable to that of branching on variable-disjunctions. One plausible reason why branching along thin directions did not perform as well as other criteria might be that most of the integer constrained variables in the test set were binary variables. For such problems, the integer width of the polytope associated with the LP relaxation of a subproblem is at most one. Furthermore, there are typically a number of directions along which the width is one. So, for the case when the minimum width of the polytope is one, the formulation (2.6) selects any one of the many possible directions arbitrarily. One way to overcome this problem would be to resort to other criteria when the minimum width is found to be one. However, we have not yet pursued this line of research.



(a) $t = 100$ seconds



(b) $t = 100$ seconds

Figure 3.6: Performance profile for number of subproblems when cuts are added to the original problem. t is fixed and k is varied.

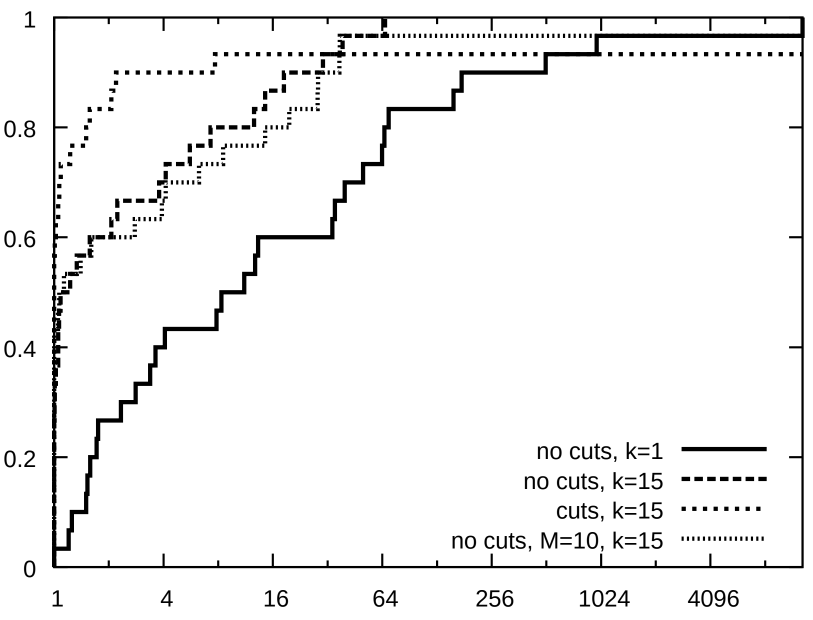


Figure 3.7: Performance profile to compare the effect of branching for maximum lower bound when M is increased to 10, $t = 100$ seconds.

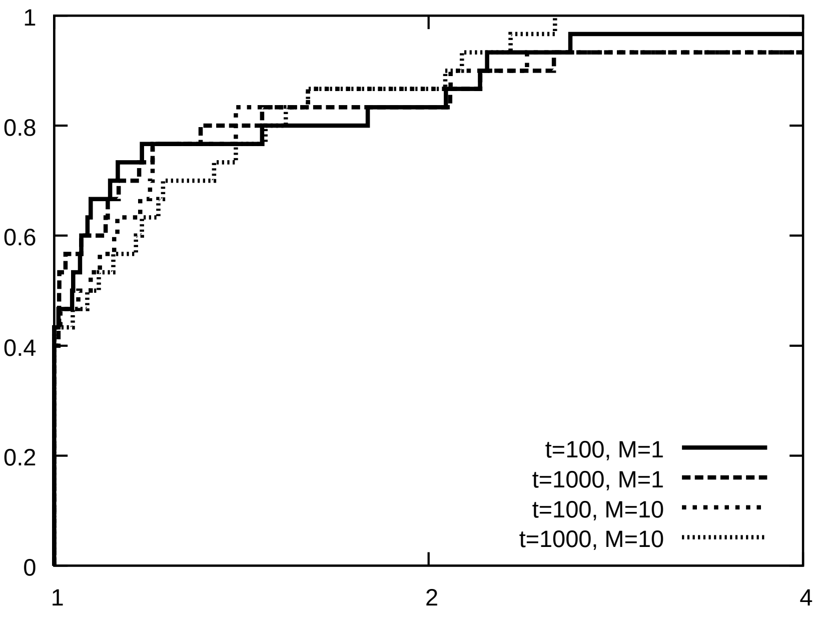


Figure 3.8: Performance profile to compare the effects of changing t when M in increased, $t = 100$ seconds.

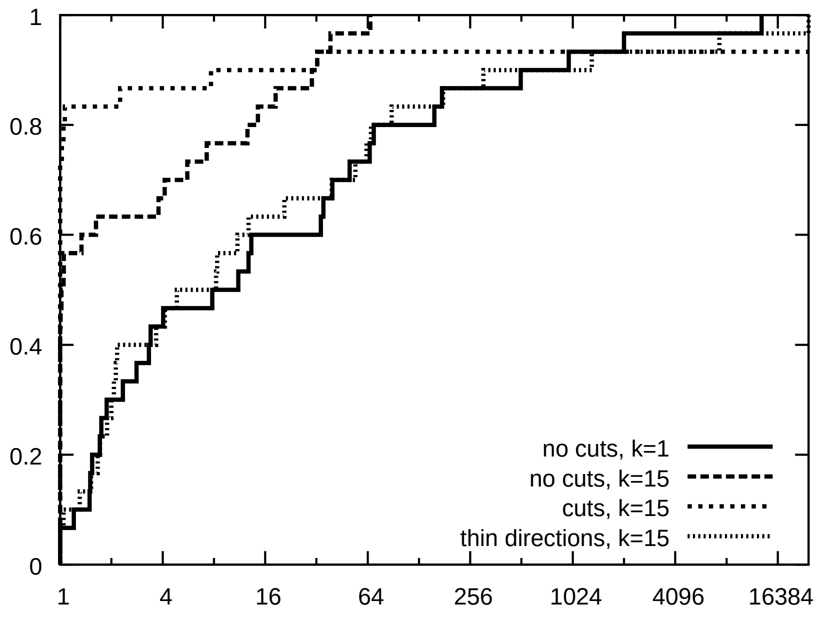


Figure 3.9: Performance profile to compare the effect of branching on "thin" directions against other criteria, $t = 100$ seconds.

3.3 Using Disjunctions with Only Two Variables

The experimental results of Section 3.2 clearly show that branching on appropriate general disjunctions can greatly reduce the size of the branch-and-bound tree when compared to strong branching on variables. The experiments also highlight a need to develop fast heuristics to identify good branching disjunctions. Two reasons why strong branching on variables seems attractive for selecting a variable disjunction for branching are:

1. There are at most d candidates that need to be evaluated, and
2. The dual-simplex method provides a fast way of re-solving the LP relaxation after a bound has been changed. The number of iterations used to solve a problem from scratch is typically an order of magnitude higher than the number required to re-solve from an advanced basis.

Keeping in mind these two points, a natural direction for developing heuristics for our problem is to consider the idea of performing strong branching on a set slightly broader than the set of variable disjunctions only but still a limited set of all general disjunctions. We consider the simplest case: strong branching on disjunctions with only two variables. Further, we consider only disjunctions with co-efficients in $\{-1, 0, 1\}$, motivated in part by the observation that using coefficients of larger absolute values did not have an impact on the performance in our previous experiments and also in part by the fact that such disjunctions are sufficient for the case of mixed binary programs, a property that is shown by most instances of our test set.

The criterion we use to select a branching disjunction is slightly different from the one used in the experiments of Section 3.2. In the following experiments, we use the score function of Linderoth and Savelsbergh [1999] to select the best candidate. More precisely, if z_L^i, z_R^i are objective function values obtained after branching on disjunction i, then the score of candidate i is evaluated as

$$s_i = \alpha \min\{z_L^i, z_R^i\} + (1 - \alpha) \max\{z_L^i, z_R^i\},$$

where α was fixed to 0.8. Such a function was not used in previous experiment, because it is trivial to find poor general disjunctions such that $\max\{z_L^i, z_R^i\}$ is unbounded, thus making the score function ineffective. Since this is not the case here, we can use the score function again. When we come across a disjunction, one side of which is infeasible, we store it as a valid inequality, ignoring its score and passing it to both subproblems along with the branching disjunction. The description of the steps in Algorithm 3.1 makes this clear. Upon completion, the algorithm yields a branching disjunction (π, π_0) and a collection of valid inequalities C. This collection of valid inequalities is added to both the subproblems obtained after branching on the disjunction (π, π_0). In the worst case, one needs to solve $2d^2$ LPs when this routine is called. This number tends to become unacceptably high even for moderately sized problems and when the number of integer-constrained variables with fractional values is much less than d. For instance, after solving the LP relaxation for the instance "10teams" with $d = 1800$ and with 148 of the integer variables having fractional values in the LP solution, this algorithm required to evaluate 971,450 LPs and used 20,000 seconds just for the root node.

In order to compare the solution values obtained after branching on such disjunctions, we use the following notation. The optimal values obtained in the two subproblems obtained after branching on the variable disjunction $x_i \leq \pi_0 \vee x_i \geq \pi_0 + 1$ are denoted as z_L^i, z_R^i respectively. Those obtained after branching with the disjunction $x_i + x_j \leq \pi_0 \vee x_i + x_j \geq \pi_0 + 1$ are denoted as $z_L^{(i,j)}, z_R^{(i,j)}$. Similarly, $z_L^{(i,-j)}, z_R^{(i,-j)}$ denote the optimal values obtained after branching with the disjunction $x_i - x_j \leq \pi_0 \vee x_i - x_j \geq \pi_0 + 1$.

Observe that number of LPs required to find the best disjunction as described above can be reduced if one can estimate correctly the score of a particular disjunction from the solutions obtained while performing strong branching on other disjunctions. To see this, consider the following simple example. Suppose $\tilde{x}_1 = 0.1, \tilde{x}_2 = 0.5$ in an optimal solution of the LP relaxation. Therefore, $x_1 - x_2 \leq -1 \vee x_1 - x_2 \geq 0$ is a valid disjunction. Also, $x_1 \leq 0 \vee x_1 \geq 1$ and $x_2 \leq 0 \vee x_2 \geq 1$ are candidates for branching. Then clearly, $z_R^{(1,-2)} \leq \min\{z_L^1, z_R^2\}$ because $\{x \in [0,1]^2 \mid x_1 \geq 1\} \subseteq \{x \in [0,1]^2 \mid x_1 - x_2 \geq 0\}$ and $\{x \in [0,1]^2 \mid x_2 \leq 0\} \subseteq \{x \in [0,1]^2 \mid x_1 - x_2 \geq 0\}$. Additionally, if $x_1 = x_2 = 0$ appeared as a solution for one of the earlier candidates, then the optimal value obtained after imposing this disjunction will provide an upper bound on $z_L^{(1,-2)}$. These upper bounds on $z_L^{(1,-2)}, z_R^{(1,-2)}$, when available, can be used to obtain an upper bound on the score of this disjunction. If this score happens to be less than the score of the best disjunction found so far, then this disjunction can be discarded without carrying out any further computation.

The improved version of Algorithm 3.1 is presented as Algorithm 3.2. We have deliberately skipped some fine details in the description of Algorithm 3.2 that can be used to further eliminate some candidates. For instance, if setting $x_i \geq 1$ makes the LP infeasible for some i, then all disjunctions of the form $x_i + x_j \leq 1 \vee x_i + x_j \geq 2, j = 2, 3, \ldots, d$ may be ignored when $x_j \in \{0, 1\}$.

A comparison of number of LPs that were solved in the root node only using Algorithm 3.1 and the time taken for the 30 instances listed in Table 3.2 are tabulated against time taken by Algorithm 3.2 in Table A.2. The performance profiles comparing the number of LPs solved and the time taken are shown in Figure 3.10 and Figure 3.11. We observe orders of magnitude improvement for most instances. We also observe that the percentage reduction seen in running time is less than that seen in the number of LPs solved, though the decrease in running time is still one to two orders of magnitude. This can probably be explained by the time used in additional book-keeping required in saving the solutions and checking if the score can be estimated using previously obtained solutions. In one instance (bell3a), we observed that the number of LPs solved was more for Algorithm 3.2 than that used by Algorithm 3.1. This is theoretically not possible and we surmise that this behavior is due to numerical difficulties associated with solving the LPs.

Figure 3.12 compares the performance of strong branching on disjunctions using Algorithm 3.2 against the time taken to solve the instances by solving the formulation in Section 3.2 with $k = 2$ and the three different values of parameter t. Even though a time limit of 20 hours was used for the formulation based branching scheme, it was only two hours for Algorithm 3.2. Five instances were not solved by Algorithm 3.2 in two hours and hence we see its profile staying horizontal at around 83%. A comparison of the number of nodes taken by the same set of experiments is shown in Figure 3.14. These figures show that the benefits of using dual-simplex method to quickly re-solve the problem from the current solution combined with the elimination of poor choices of candidates far outweigh the problem of explicitly evaluating a large number of possible candidates.

Algorithm 3.1: Algorithm *evaluate-all* to find the best disjunction with at most two coefficients by enumerating all solutions.

```
Input: A subproblem $P^k$, optimal solution $\tilde{x}^k$ of its LP relaxation, integer
       constrained variables $1, 2, \dots, d$.
Output: A disjunction $(\pi, \pi_0)$
$C = \phi$, $maxscore = -\infty$;
foreach $i$ in $1, 2, \dots, d$ do
    foreach $j$ in $i, i+1, i+2, \dots, d$ do
        foreach $l$ in $\{0, 1\}$ do
            if $(i == j$ && $l == 1)$ then CONTINUE;
            else if $(i == j$ && $l == 0)$ then
                $activity = \tilde{x}_i^k$;
                $\hat{\pi} = e_i, \hat{\pi}_0 = \lfloor activity \rfloor$;
            else if $(l == 0)$ then
                $activity = \tilde{x}_i^k + \tilde{x}_j^k$;
                $\hat{\pi} = e_i + e_j, \hat{\pi}_0 = \lfloor activity \rfloor$;
            else if $(l == 1)$ then
                $activity = \tilde{x}_i^k - \tilde{x}_j^k$;
                $\hat{\pi} = e_i - e_j, \hat{\pi}_0 = \lfloor activity \rfloor$;
            end
            if $activity - \lfloor activity \rfloor > 0$ then
                $(z_L, z_R) = strongbranch(P^k, \hat{\pi}, \hat{\pi}_0)$ ;
                if $\min\{z_L, z_R\} >= \infty$ then
                    $\pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$ ;
                    STOP.
                end
                if $z_L >= \infty$ then Add inequality $(\hat{\pi}, \hat{\pi}_0 + 1)$ to $C$;
                if $z_R >= \infty$ then Add inequality $(-\hat{\pi}, -\hat{\pi}_0)$ to $C$;
                $score = \alpha \min\{z_L, z_R\} + (1 - \alpha) \max\{z_L, z_R\}$ ;
                if $(score < \infty$ && $score > maxscore)$ then
                    $score = maxscore$ ;
                    $\pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$ ;
                end
            end
        end
    end
end
```

Algorithm 3.2: A refined version of Algorithm 3.1, called *with-elimination*, to find the best disjunction with at most two coefficients by enumerating all solutions.

```
Input: A subproblem $P^k$, optimal solution $\tilde{x}^k$ of its LP relaxation, integer
       constrained variables $1, 2, \ldots, d$.
Output: A disjunction $(\pi, \pi_0)$
$C = \phi$, $maxscore = -\infty$;
foreach $i$ in $1, 2, \ldots, d$ do
    foreach $j$ in $i, i+1, i+2, \ldots, d$ do
        foreach $l$ in $\{0, 1\}$ do
            if $(i == j$ && $l == 1)$ then CONTINUE;
            else if $(i == j$ && $l == 0)$ then
                $activity = \tilde{x}^k_i$;
                $\hat{\pi} = e_i, \hat{\pi}_0 = \lfloor activity \rfloor$;
            else if $(l == 0)$ then
                $activity = \tilde{x}^k_i + \tilde{x}^k_j$;
                $\hat{\pi} = e_i + e_j, \hat{\pi}_0 = \lfloor activity \rfloor$;
            else if $(l == 1)$ then
                $activity = \tilde{x}^k_i - \tilde{x}^k_j$;
                $\hat{\pi} = e_i - e_j, \hat{\pi}_0 = \lfloor activity \rfloor$;
            end
            if $activity - \lfloor activity \rfloor > 0$ then
                Find $\hat{x}_L \in L$ with minimum $cx$ s.t. $\hat{x}_L$ satisfies $(-\hat{\pi}, -\hat{\pi}_0)$;
                Find $\hat{x}_R \in L$ with minimum $cx$ s.t. $\hat{x}_R$ satisfies $(-\hat{\pi}, -\hat{\pi}_0)$;
                if $\hat{x}_L, \hat{x}_R$ exist then $estscore = \alpha \min\{z^i_L, z^i_R\} + (1-\alpha)\max\{z^i_L, z^i_R\}$
                ;
                else $estscore = \infty$;
                if $(estscore > maxscore)$ then
                    $(z_L, z_R, L) = strongbranch2(P^k, \hat{\pi}, \hat{\pi}_0, L)$ ;
                    if $\min\{z_L, z_R\} >= \infty$ then
                        $\pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$, STOP ;
                    if $z_L >= \infty$ then Add inequality $(\hat{\pi}, \hat{\pi}_0 + 1)$ to $C$ and $P^k$;
                    if $z_R >= \infty$ then Add inequality $(-\hat{\pi}, -\hat{\pi}_0)$ to $C$ and $P^k$;
                    $score = \alpha \min\{z_L, z_R\} + (1-\alpha)\max\{z_L, z_R\}$ ;
                    if $(score < \infty$ && $score > maxscore)$ then
                        $score = maxscore$ ;
                        $\pi = \hat{\pi}, \pi_0 = \hat{\pi}_0$ ;
                    end
                end
            end
        end
    end
end
```

Algorithm 3.3: Algorithm *strongbranch*$(Q, \hat{\pi}, \hat{\pi}_0)$ describing the strong branching subroutine used in Algorithm 3.1

```
Input: An LP $Q$, a disjunction $(\hat{\pi}, \hat{\pi}_0)$.
Output: The objective function values $z_L, z_R$ obtained after branching on $(\hat{\pi}, \hat{\pi}_0)$.
Add the inequality $(-\hat{\pi}, -\hat{\pi}_0)$ to $Q$ and solve. Let the solution value be $z_L$;
Remove the inequality from $Q$;
if $Q$ is infeasible then  $z_L = \infty$;
Add the inequality $(\hat{\pi}, \hat{\pi}_0 + 1)$ to $Q$ and solve. Let the solution value be $z_R$;
Remove the inequality from $Q$;
if $Q$ is infeasible then  $z_R = \infty$;
```

Algorithm 3.4: Algorithm *strongbranch*$2(Q, \hat{\pi}, \hat{\pi}_0, L)$ describing the strong branching subroutine used in Algorithm 3.2

```
Input: An LP $Q$, a disjunction $(\hat{\pi}, \hat{\pi}_0)$, a list $L$ of feasible points.
Output: The objective function values $z_L, z_R$ obtained after branching on $(\hat{\pi}, \hat{\pi}_0)$
        and a list $L$ of feasible points.
Add the inequality $(-\hat{\pi}, -\hat{\pi}_0)$ to $Q$ and solve. Let the solution value be $z_L$;
Remove the inequality from $Q$;
if $Q$ is infeasible then  $z_L = \infty$;
else Append the current solution to $L$ Add the inequality $(\hat{\pi}, \hat{\pi}_0 + 1)$ to $Q$ and
solve. Let the solution value be $z_R$;
Remove the inequality from $Q$;
if $Q$ is infeasible then  $z_R = \infty$;
else Append the current solution to $L$
```

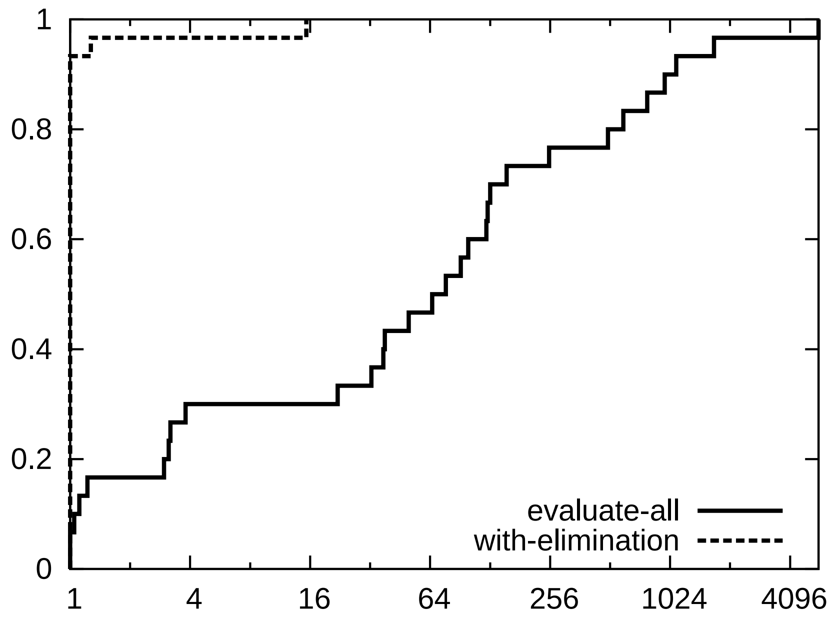


Figure 3.10: Performance profile comparing the performance of Algorithm 3.1 (*evaluate-all*) and Algorithm 3.2 (*with-elimination*) in terms of number of LPs solved in the root node.

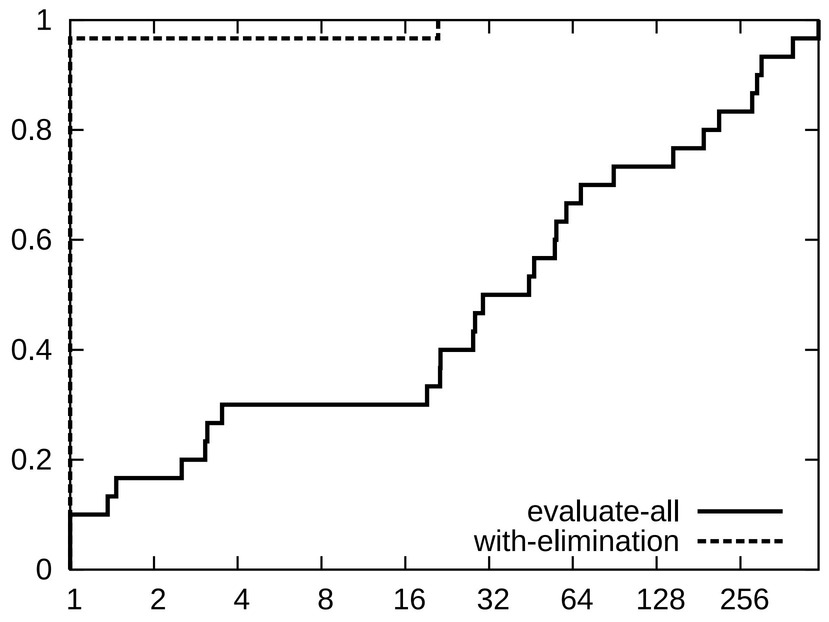


Figure 3.11: Performance profile comparing the performance of Algorithm 3.1 (*evaluate-all*) and Algorithm 3.2 (*with-elimination*) in terms of time taken in the root node.

3.4 Improving Branching on Variables

The dramatic improvement in the time taken to select disjunctions with two non zero coefficients by using the elimination procedure of Algorithm 3.2 motivated us to try using a similar technique for improving the performance of strong branching on single variables as well. Most solvers use strong branching, at least until the pseudo-costs of variables are assumed to be reliably known. Our enhancements to the strong branching procedure can potentially eliminate poor candidates without having to spend time when not necessary. The procedure is exactly the same as described in the previous section with the exception that only single variable disjunctions are considered in place of disjunctions with two variables. In order to test this procedure, we performed several experiments on all 177 instances listed in Tables A.1.

Most commercial solvers like CPLEX have built-in functions for strong branching, and

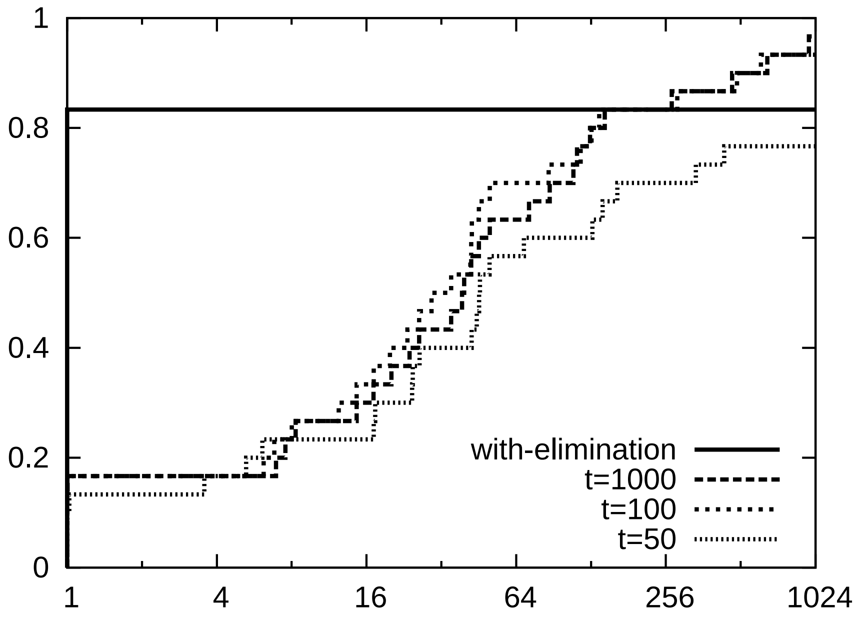


Figure 3.12: Performance profile comparing the time taken by Algorithm 3.2 (*with-elimination*) against formulation based branching with parameter $t = 1000, 100$ and $50s$.

we are comparing the performance of our procedure against this function in our tests. This is in spite of the fact that we do not know what kind of algorithm is used inside this function. We studied the source code of an open source LP solver, CLP, but did not find any strategy similar to that described in Section 3.3. Instead, we found that it implements several techniques such as saving the LU factorization, disabling scaling, turning off some checks, etc. in order to speed up the dual-simplex iterations by utilizing the knowledge that only variable bounds are being changed in each iteration.

In the following experiments, we simply compare the strategy of explicitly calculating the score of each candidate variable (like in Algorithm 3.1) with that of calculating scores for those candidates that cannot be eliminated on the basis of feasible solutions obtained from other candidates (as in Algorithm 3.2). In the first experiment, we performed simple branch-and-bound, without using any valid inequalities, primal heuristics or preprocessing techniques. We also provided as input the best known solution value, given so as to

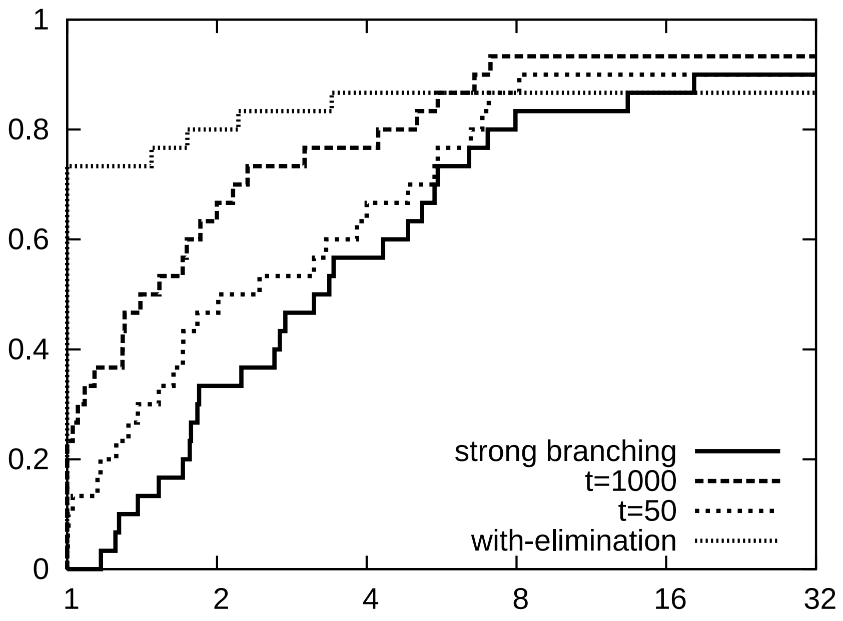


Figure 3.13: Performance profile comparing the nodes used by Algorithm 3.2 (*with-elimination*) against formulation based branching with parameter $t = 1000s$, $50s$ and also strong branching on variables.

mitigate the effect of the order in which each subproblem was solved. The time taken using the two strategies is profiled in Figure 3.14 and is compared against the inbuilt strong branching function of CPLEX. The figure shows that our enhancements do in fact speed up strong branching substantially when compared against the explicit evaluation of all candidates. The enhanced performance, however, still lags behind that of the built-in function, presumably because of the above mentioned techniques implemented in the LP solver. We now compare the performance of our procedures in order to observe how the difference in number of calls to dual-simplex procedure in strong branching translates in to the amount of time used. In order to observe this, we shortlisted 55 problems of medium difficulty. These are all those instances out of the initial set of 177 instances that satisfy two criteria:

1. the instance was solved to completion by either of our procedures or by CPLEX (with strong branching) within two hours and,
2. it took more than 10 seconds to solve the instance by any of the three procedures.

These instances are listed in Table 3.3. Figure 3.15 shows that evaluating all candidates requires calling the dual-simplex procedure 50% more times than that required by the elimination procedure in more than half of the instances. Figure 3.16, however, shows that the savings in time are not as big as the savings in the number of calls, probably on account of extra time spent in bookkeeping.

We now report the observations for a more realistic experiment: when primal heuristics and generation of valid inequalities are enabled and no upper bound is provided as an input for cut off. The time taken when the above three branching strategies are used is profiled in Figure 3.17. We observe that the elimination strategy is more useful under these settings than it was in pure branch-and-bound as described above. We also shortlisted 74 instances, listed in Table 3.4, using the above two criteria again. Figure 3.18 and Figure 3.19 show that the benefits of using an elimination strategy are more pronounced

Table 3.3: List of 55 instances shortlisted as reasonably sized problems for branch-and-bound.

10teams	cap6000	mas76	neos648910	qiu
30_05_100	core2536-691	misc07	neos7	ran10x26
30_95_98	dano3_3	mitre	neos8	ran12x21
aflow30a	dano3_4	mod011	neos823206	ran13x13
air04	dano3_5	modglob	nug08	rout
air05	fiber	mzzv42z	nw04	seymour1
bell5	gesa2	neos10	pk1	stein45
biella1	gesa2_o	neos11	pp08aCUTS	swath2
bienst1	harp2	neos13	prod1	trento1
bienst2	lrn	neos4	prod2	vpm1
blp-ir98	mas74	neos5	qap10	vpm2

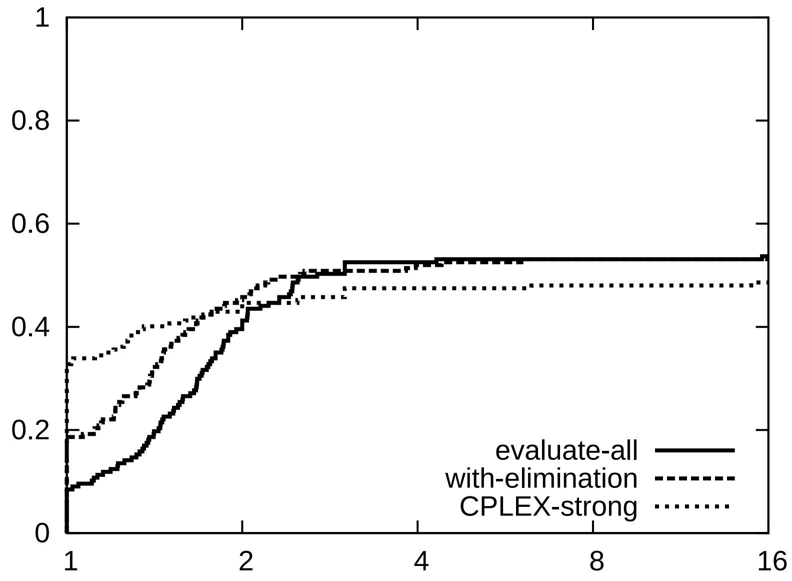


Figure 3.14: Performance profile of time taken in branch-and-bound when the strong branching variable is chosen using the single-variable variant of Algorithm 3.2 (*with-elimination*), that of Algorithm 3.1 (*evaluate-all*) and inbuilt function of CPLEX.

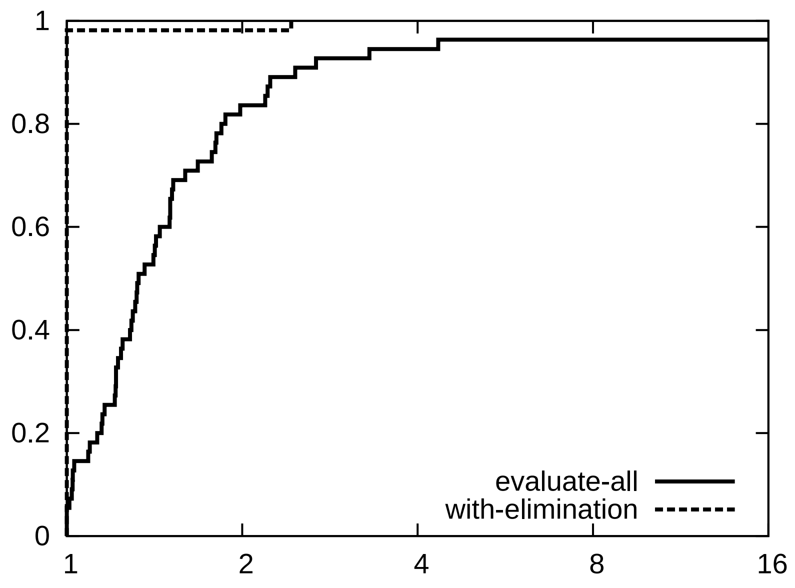


Figure 3.15: Performance profile of number of calls to dual-simplex in branching for instances in Table 3.3 for the same experiment as in Figure 3.14.

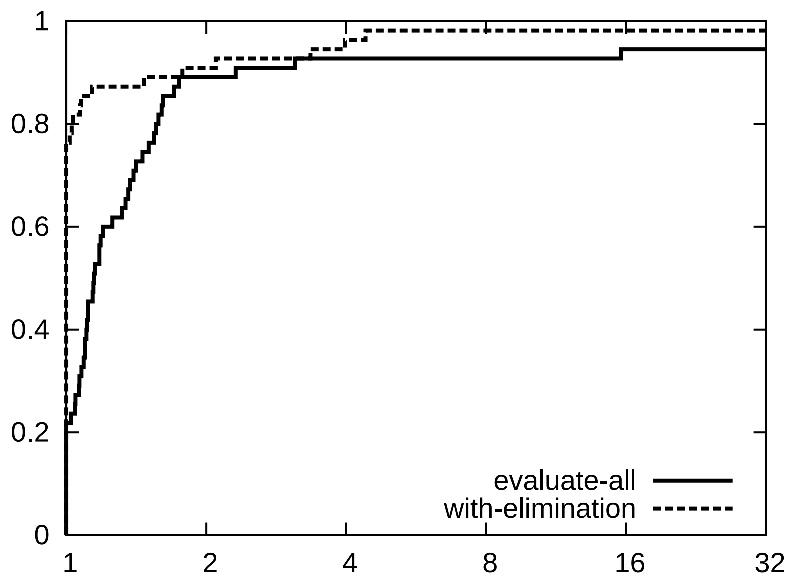


Figure 3.16: Performance profile of the same experiment as Figure 3.14 when the instances are limited to those listed in Table 3.3.

Table 3.4: List of 74 instances shortlisted as reasonably sized problems for branch-and-cut.

10teams	bienst2	m20-75-5	neos4	qap10
30_05_100	binkar10_1	markshare1_1	neos5	qiu
30_95_100	blend2	markshare2_1	neos6	qnet1
30_95_98	dano3_3	mas74	neos648910	qnet1_o
acc3	dano3_4	mas76	neos7	railway_8_1_0
acc4	dano3_5	misc07	neos8	ran10x26
aflow30a	fixnet6	mitre	noswot	ran12x21
air03	gesa2_o	mod011	nug08	ran13x13
air04	gt2	modglob	nw04	rout
air05	harp2	neos10	p2756	seymour1
arki001	l152lav	neos11	pk1	stein45
bc1	m20-75-1	neos13	pp08a	swath2
bell3a	m20-75-2	neos20	pp08aCUTS	swath3
bell5	m20-75-3	neos21	prod1	vpm2
bienst1	m20-75-4	neos23	prod2	

in these realistic settings than when doing pure branch-and-bound with an known upper bound. The profile in Figure 3.20 shows that our elimination strategy performs better than the strong branching function of CPLEX for these 74 instances. The differences in the profiles of Figure 3.17 and Figure 3.20 suggest that the elimination strategy becomes more effective as the difficulty of the problem increases.

3.5 Conclusions

In this chapter, we considered computational employment of general disjunctions of the form "$\pi x \leq \pi_0 \vee \pi x \geq \pi_0 + 1$" in branch-and-bound and branch-and-cut algorithms. We formulated the problem of selecting the optimal such disjunction using two different criteria and reported on the effect of using the associated optimization models to select branching disjunctions within the branch-and-bound framework of the commercial solver CPLEX. The naive approach to formulating and solving the branching disjunction selection problem described herein yielded mixed results. The optimization problems that arose turned out to be extremely difficult to solve using off-the-shelf software. On the bright side, we observed

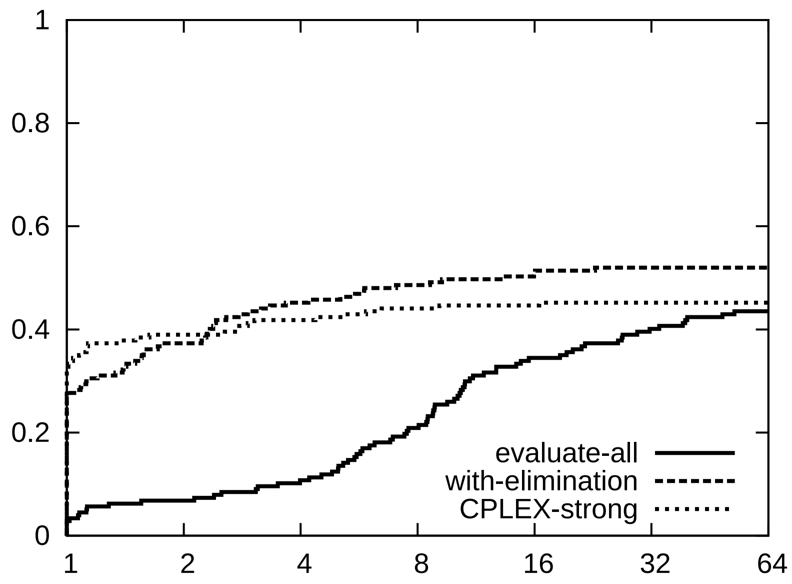


Figure 3.17: Performance profile of time taken in branch-and-cut when the strong branching variable is chosen using the single-variable variant of Algorithm 3.2 (*with-elimination*), that of Algorithm 3.1 (*evaluate-all*) and strong branching function of CPLEX

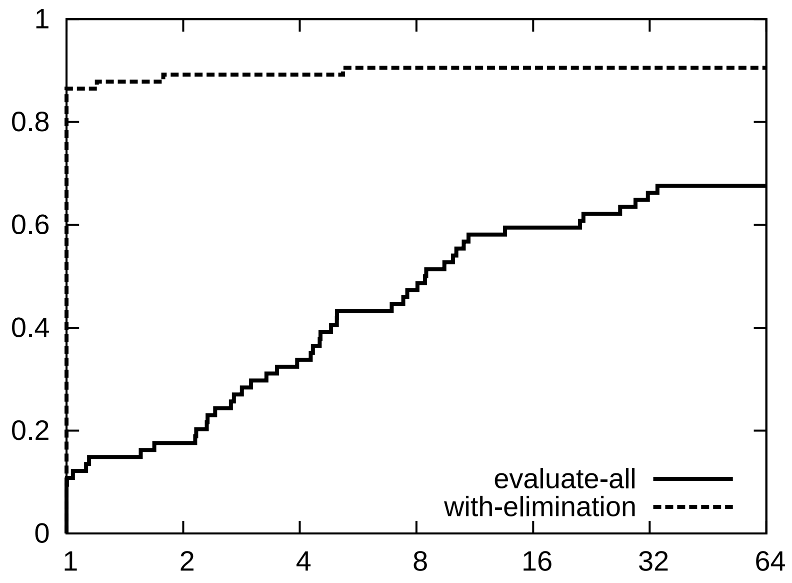


Figure 3.18: Performance profile of time taken in branch-and-cut for same experiment as in Figure 3.17 for instances listed in Table 3.4

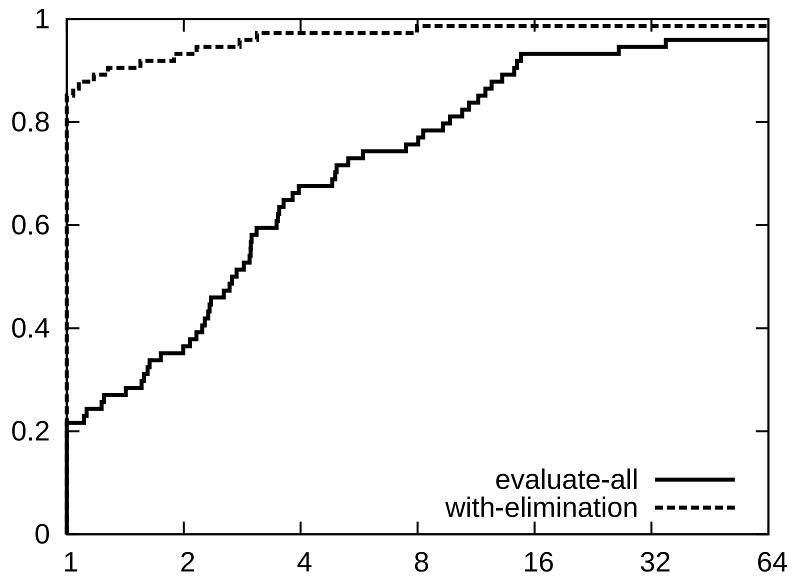


Figure 3.19: Performance profile of number of calls to dual-simplex procedure in branching in same experiment as Figure 3.18 for instances listed in Table 3.4

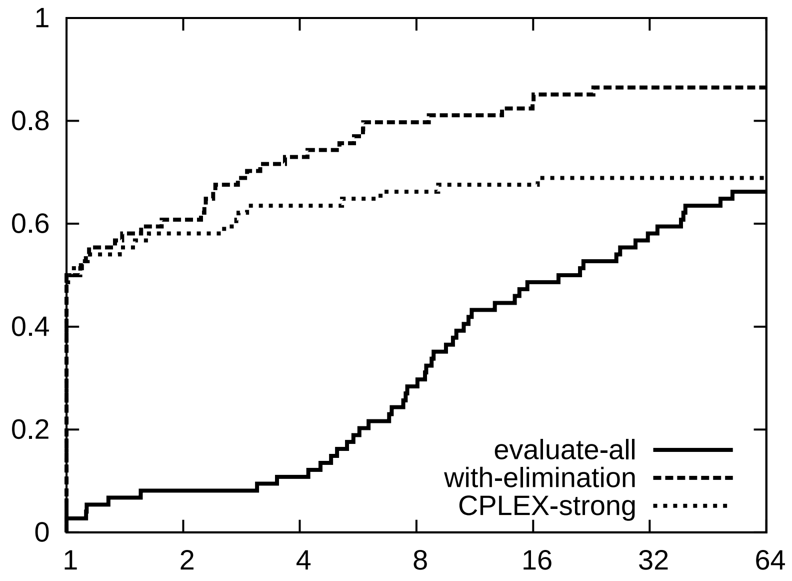


Figure 3.20: Performance profile of time taken in branch-and-cut for instances listed in Table 3.4(with CPLEX)

consistent substantial reductions in the number of subproblems required to be solved when using general disjunctions for branching. Motivated by these results, we considered using general disjunctions with only two variables. We observed that even though the number of such candidates could be unacceptably high for evaluating each of them explicitly with dual-simplex procedure, still one can reduce this number by orders of magnitude by a simple elimination strategy. Our experiments showed that this strategy outperforms the strategy of selecting such candidates by solving the associated MIP formulation, thus suggesting that solving the formulation may not be the best way for finding disjunctions with only a few coefficients. However, further research into ways of solving the formulation may lead to more competitive methods. We also used the elimination procedure for variable disjunctions and showed that the strong branching function can be significantly improved by this method.

Chapter 4

Computational Methods for Valid Inequalities

4.1 Introduction

Valid inequalities that can be used in a cutting-plane algorithm can be generated in many ways. Some classes of valid inequalities are derived from the special structure of the instance, e.g., flow and cover inequalities, implication inequalities, knapsack inequalities, clique inequalities etc., while others are more general and can be derived from the simplex tableau or some other transformations of the constraint matrix A e.g., C-G inequalities, MIR inequalities, GMI inequalities, Lift and Project inequalities etc. So, given a MIP instance, one can generate a variety of valid inequalities for the cutting-plane algorithm. A good choice of valid inequality can reduce the number of iterations of the algorithm significantly. However, a poor choice of an inequality can lead to many more iterations and also make the LP relaxation difficult to solve. The choice of valid inequalities is thus critical for the branch-and-cut algorithm.

As described in Section 1.4.2, all the above mentioned classes of inequalities are special cases of split inequalities that can be derived from general disjunctions. The disjunctions that we derived in Chapter 2 and Chapter 3 for maximally improving the bound are then natural candidates for deriving split inequalities as well. The experiments we perform in this chapter look at precisely such inequalities.

Motivated by the computational results from Chapter 3 showing the effectiveness of the criterion of maximizing the lower bound for selecting a disjunction for branching, we propose to employ the same criterion for selecting disjunctions for generating valid inequalities as well. In Section 4.2, we compare the effects of selecting C-G inequalities on the basis of the two above mentioned criteria. We observe that the problem of maximizing the bound improvement after adding a C-G inequality can be formulated as a MIP, not much different from one used in Chapter 3. In Section 4.3, we perform a similar experiment for split inequalities. Our experiments indicate that a greater improvement in bound can be achieved by selecting much fewer inequalities using our criteria.

In Section 4.4 we consider the problem of deciding whether the disjunctions that we derived are more suitable for branching or for generating valid inequalities. We perform a simple experiment to observe the differences in performance of these two approaches. Even though the experiment does not completely settle the question, it does give some valuable insight in to the problem and also gives a motivation to study a larger problem of deciding whether one should use valid inequalities or branch in order to most efficiently solve a given problem (or even a subproblem in the branch-and-cut algorithm). We finally present conclusions in Section 4.5.

4.2 Selecting C-G Inequalities

In this section, we compare the two criteria for selecting Chvátal-Gomory cuts. We first formulate the problem of selecting a C-G inequality on the basis of these criteria. Recall from Section 1.4.2 that an inequality $(\hat{\alpha}, \hat{\beta}) \in \mathbb{Z}^{n+1}$ is an elementary C-G inequality for a MIP in standard form (1.1) if $\min_{x \in \mathbb{R}^n} \{\hat{\alpha} x \mid Ax \geq b\} > \hat{\beta} - 1$. Consider the following two

LPs

$$\begin{aligned} Ax &\geq b \\ \hat{\alpha}x &\leq \hat{\beta} - 1, \end{aligned} \quad \text{and} \quad \begin{aligned} Ax &\geq b \\ cx &\leq K \\ \hat{\alpha}x &\geq \hat{\beta}, \end{aligned} \tag{4.1}$$

where $K \in \mathbb{R}^n$ is given. Clearly, both the above LPs are infeasible if and only if $(\hat{\alpha}, \hat{\beta})$ is a valid C-G inequality that when added to the MIP pushes the optimal value of the LP relaxation to at least K. Using the same approach used in Section 2.2.1, it can be shown that we can improve the lower bound of the LP relaxation to at least K if and only if the program

$$\begin{aligned} pA - \alpha &= 0 \\ pb - \beta &\geq -1 + \delta \\ qA - s_R c + \alpha &= 0 \\ pb - s_R K + \beta &> \delta \\ p, q, s_R &\geq 0 \\ (\alpha, \beta) &\in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1, \end{aligned} \tag{4.2}$$

has a feasible solution with $\alpha = \hat{\alpha}, \beta = \hat{\beta}$ for some $\delta > 0$. The maximum such K can be obtained by solving program (4.2) for different values K using a standard MIP solver. The same approach can be used to find a *maximum violation* C-G inequality by solving the MIP,

$$
\begin{aligned}
\max \beta - \alpha x^* & \\
pA - \alpha &= 0 \\
pb - \beta &\geq -1 + \delta \\
p &\geq 0 \\
(\alpha, \beta) &\in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1,
\end{aligned}
\tag{4.3}
$$

for some $\delta > 0$. Formulation (4.3) has been used by Fischetti and Lodi [2005] for determining the elementary closure of C-G inequalities and also mentioned by Eisenbrand [1999].

In order to compare the effectiveness of the criteria of *maximum violation* and *maximum bound improvement*, we carried out computational experiments using a set-up described in Section 1.5. For the criterion of *maximum violation*, we solve, in each iteration, the MIP (4.3) and the C-G inequality corresponding to the best solution obtained in a time limit of 1000 seconds. For selecting a C-G inequality resulting in *maximum bound improvement*, we solve, at each iteration, several MIPs of the form 4.2, to find the maximum value for K. Each of these MIPs is solved with a time limit of 1000 seconds. A C-G inequality (α, β) obtained for the highest such K is then added to the LP relaxation and the process is repeated. For both cases, we continue iterating until we are unable to find a required valid C-G inequality or we reach the maximum time limit of four hours. We also added the constraint $-1 \leq \alpha_i \leq 1, i = 1, 2, \ldots, d$ to further restrict the problem so as to achieve results in a reasonable time.

In order to measure the effectiveness of the two criteria, we compare them on the basis of the percentage of the gap, between the objective function value of the LP relaxation and the best known solution closed by each. The measure can be expressed formally as

$$\phi = 100 \left(1 - \frac{z - z_{LP}}{z_{best} - z_{LP}} \right),$$

where z is the lower bound obtained after the C-G cuts have been added iteratively, z_{LP} is the lower bound obtained after solving the LP relaxation of the original instance and z_{best} is the value of the best known solution for that instance.

Figure 4.1 plots the gap closed by using each of the two criteria and the number of cuts that were generated. The number of iterations used by each procedure is the same as the number of cuts since we add only one inequality in each iteration. We observe that the criterion of *maximum bound improvement* closed the gap by more than 64% for eight instances but the criterion of *maximum violation* did the same for only three instances. The tallies for 16% gap stand at 19 and 10 instances, respectively. We also observe that 10 or more inequalities were generated for only four instances when the criterion of *maximum bound improvement* was used, while the same number for the other criterion was 77. Thus, even though generation of inequalities that maximize violation is easier (as reflected from the number of inequalities generated) than the generation of those that *maximize bound improvement*, they appear to be significantly less effective in changing the bound. Table A.3 tabulates the number of cuts and the gap closed for all 177 instances.

4.3 Selecting Split Inequalities

We now describe a similar experiment for selecting split inequalities. Recall from Section 1.4.2 that an inequality $(\alpha, \beta) \in \mathbb{R}^{n+1}$ is a split inequality for a MIP of the form (1.1) if (α, β) is a valid inequality for both the disjunctive subsets $\{x \in \mathbb{R}^n \mid Ax \geq b, \hat{\pi}x \leq \hat{\pi}_0\}$ and $\{x \in \mathbb{R}^n \mid Ax \geq b, \hat{\pi}x \geq \hat{\pi}_0 + 1\}$ for some general disjunction $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}$. We have already described in Section 2.4 that if the bound obtained after branching on a general disjunction $(\hat{\pi}, \hat{\pi}_0)$ improves to say K, then the same disjunction can be used to

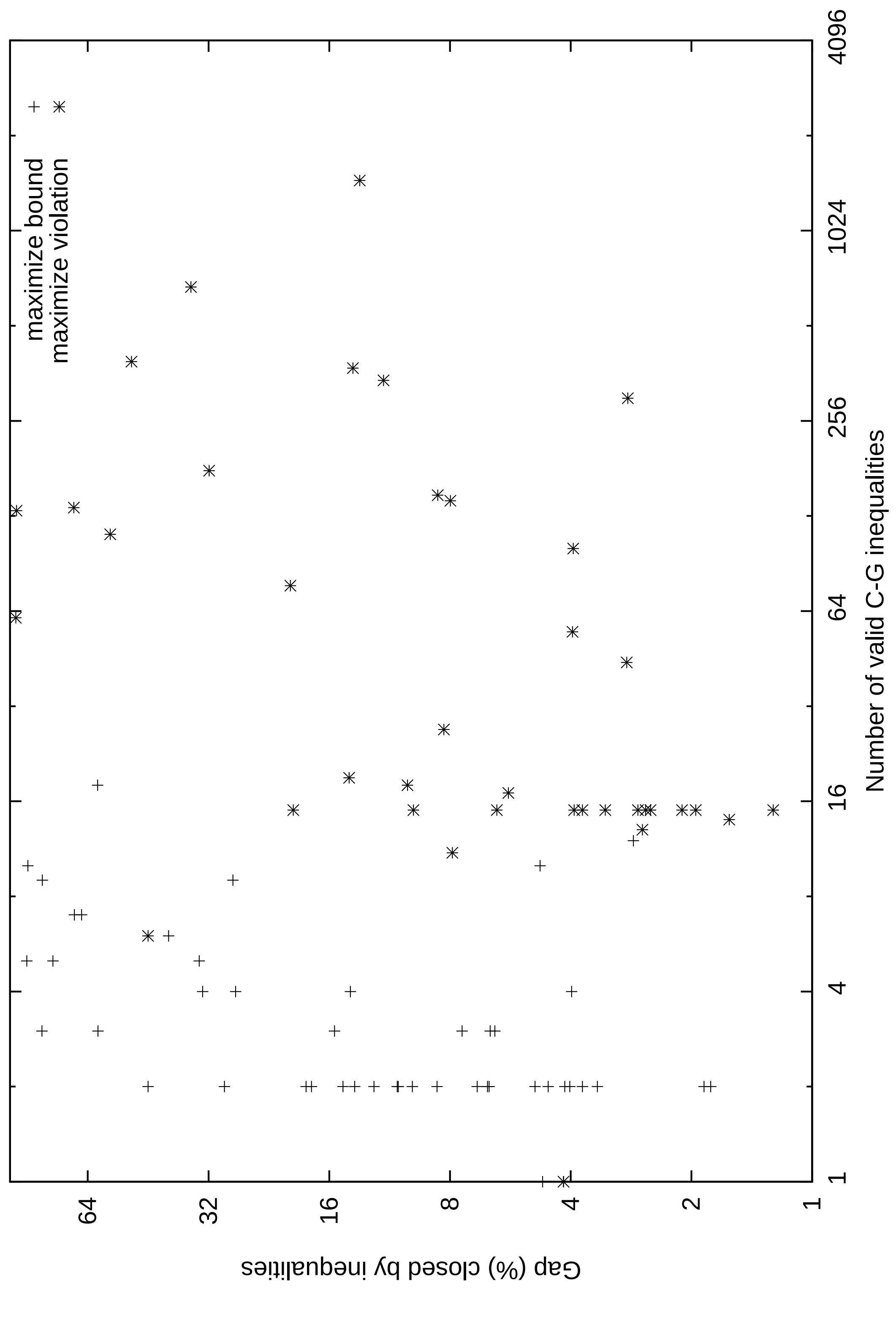


Figure 4.1: Plot of number of C-G inequalities generated and the gap closed by them.

generate split inequalities which when added to the LP relaxation will improve the bound to K. The MIP formulation (3.3) can be solved for some $\delta > 0$ in order to find such a disjunction. We rewrite the formulation here

$$\begin{aligned} pA - s_L c - \pi &= 0 \\ qA - s_R c + \pi &= 0 \\ pb - s_L K - \pi_0 &\geq \delta \\ qb - s_R K + \pi_0 &\geq -1 + \delta \\ p, q, s_L, s_R &\geq 0 \\ (\pi, \pi_0) &\in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}^1. \end{aligned} \tag{4.4}$$

In order to select a disjunction that generates a split inequality with maximum violation, we resort to the technique of Balas and Saxena [2007]. Suppose $(\hat{\alpha}, \hat{\beta}) \in \mathbb{R}^{n+1}$ is an elementary split inequality for a given MIP of the form (1.1). Let $(\hat{\pi}, \hat{\pi}_0) \in \mathbb{Z}^d \times \{0\}^{n-d} \times \mathbb{Z}$ be a general disjunction that can be used to derive $(\hat{\alpha}, \hat{\beta})$. Then, by definition, $(\hat{\alpha}, \hat{\beta})$ is a valid inequality for the feasible regions of both the following LPs:

$$\begin{aligned} Ax &\geq b \\ \hat{\pi} x &\leq \hat{\pi}_0 \end{aligned} \quad \text{and} \quad \begin{aligned} Ax &\geq b \\ \hat{\pi} x &\geq \hat{\pi}_0 + 1 \end{aligned} \tag{4.5}$$

The inequality $(\hat{\alpha}, \hat{\beta})$ is valid for the above two systems if and only if there exist $p, q \in \mathbb{R}^m_+$ and $s_L, s_R \in \mathbb{R}_+$ such that

$$\begin{aligned} pA - s_L \hat{\pi} &= \hat{\alpha} \\ pb - s_L \hat{\pi}_0 &\geq \hat{\beta} \end{aligned} \quad \text{and} \quad \begin{aligned} qA + s_R \hat{\pi} &= \hat{\alpha} \\ qb + s_R(\hat{\pi}_0 + 1) &\geq \hat{\beta}. \end{aligned} \tag{4.6}$$

Thus, in order to find a split inequality that violates a given point, say $\hat{x} \in \mathbb{R}^n$, one can solve the following program

$$\begin{aligned} \min \alpha\hat{x} - \beta \\ s.t. pA - s_L\pi = \alpha \\ pb - s_L\pi_0 \geq \beta \\ qA + s_R\pi = \alpha \\ qb + s_R(\pi_0 + 1) \geq \beta \\ p, q \in \mathbb{R}^m_+, \alpha \in \mathbb{R}^n \\ s_L, s_R, \beta \in \mathbb{R}_+ \\ \pi \in \mathbb{Z}^d \times \{0\}^{n-d}, \pi_0 \in \mathbb{Z}. \end{aligned} \tag{4.7}$$

Observe that the program (4.7) has nonlinear terms $s_L\pi, s_L\pi_0$, etc. Also observe that if there exists a solution to the program such that the objective function is negative, then one can scale the variables $\alpha, \beta, p, q, s_L, s_R$ arbitrarily to make the objective function $\alpha\hat{x} - \beta$ arbitrarily low. Balas and Saxena [2007] propose normalizing the variables by using an additional constraint $s_L + s_R = 1$. Under such a normalization, one can reformulate the program (4.7) as a parametric MIP with a single parameter θ

$$\begin{aligned}
\min \alpha\hat{x} - \beta \\
s.t. pA - \theta\pi = \alpha \\
pb - \theta\pi_0 \geq \beta \\
qA + (1-\theta)\pi = \alpha \\
qb - (1-\theta)(\pi_0 + 1) \geq \beta \\
p, q \in \mathbb{R}^m_+, \alpha \in \mathbb{R}^n, \beta \in \mathbb{R} \\
\pi \in \mathbb{Z}^d \times \{0\}^{n-d}, \pi_0 \in \mathbb{Z}. \\
\theta \in [0,1]
\end{aligned} \tag{4.8}$$

We can now solve a series of MIPs of the form (4.8), fixing θ to different values in each iteration in order to solve approximately the program (4.7).

A C-G inequality is itself a valid disjunction whose one disjunctive subset is infeasible. However, this is not true for the case of split inequalities and thus, one can generate many different split inequalities using a given general disjunction $(\hat{\pi}, \hat{\pi}_0)$. In our experiments, we use an iterative procedure to generate split inequalities from disjunctions based on either criteria. This procedure uses a cut-generating LP (CGLP) similar to one mentioned in

Section 2.6. Given a general disjunction (π, π_0), and a point $\hat{x} \in \mathbb{R}^n$, we solve the LP

$$\begin{aligned} \min \alpha\hat{x} - \beta & \\ pA - s_L\hat{\pi} = \alpha & \\ qA + s_R\hat{\pi} = \alpha & \\ pb - s_L\hat{\pi}_0 \geq \beta & \\ qb + s_R(\hat{\pi}_0 + 1) \geq \beta & \\ p, q, s_L, s_R \geq 0 & \\ \alpha \in \mathbb{R}^n, \beta \in \mathbb{R} & \end{aligned} \tag{4.9}$$

to obtain a split inequality (α, β). The normalization constraint $\sum_{i=1}^m p_i + \sum_{i=1}^m q_i + s_L + s_R = 1$ is used to avoid arbitrary high values of variables in the case a desired inequality exists. If this LP returns a feasible solution with a non-negative solution value, we add the inequality (α, β) to the LP relaxation of the MIP and obtain a new solution $\hat{x}$ to the tighter LP. We continue to iteratively solve CGLPs of the form (4.9) until one becomes infeasible or we obtain a solution that lies in one of the disjunctive subsets associated with the disjunction (π, π_0). In the latter case, we identify another disjunction using the desired criterion and start again. The iterative procedure for solving MIPs to obtain a disjunction and then generating split inequalities from that disjunction is depicted in Figure 4.2.

Table A.4 tabulates the number of inequalities generated and the gap closed by the procedures described above. The scatter plot in Figure 4.3 provides a graphical description of the same. We observe that the disjunctions generated using the criterion of maximum bound improvement usually provide split inequalities that can close much more of the gap than that can be closed by the criterion of maximum violation. Further, the difference between the two criteria is much more pronounced than the difference in the case of C-G inequalities.

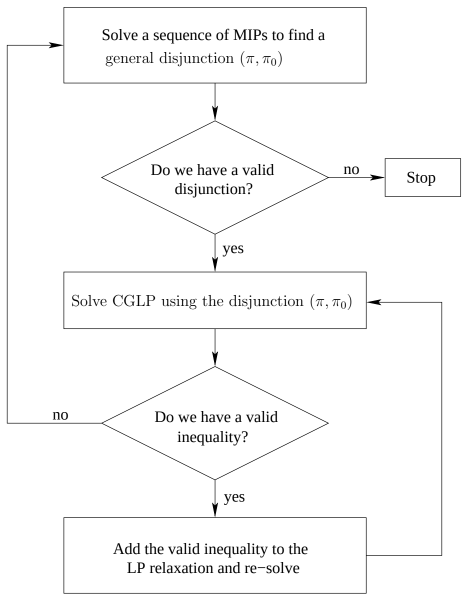

Solve a sequence of MIPs to find a general disjunction (π, π_0)
Do we have a valid disjunction?
no
Stop
yes
Solve CGLP using the disjunction (π, π_0)
Do we have a valid inequality?
no
yes
Add the valid inequality to the LP relaxation and re−solve

Figure 4.2: A flowchart depicting the procedure used in the experiments for generating split inequalities.

4.4 Cut or Branch

Experiments performed in Section 3.2, Section 4.1 and Section 4.2 indicate that using certain disjunctions can reduce significantly the number of iterations of both the branch-and-bound and the cutting-plane algorithm. In this section, we perform experiments to observe whether the disjunctions derived previously are more effective when used for branching or for generating valid inequalities. This experiment is thus a step towards understanding the more general question of deciding whether a given arbitrary disjunction should be used for branching or for generating valid inequalities which in turn is a step towards understanding at what stage should one switch from a cutting-plane algorithm to the branch-and-bound algorithm in a branch-and-cut framework.

Given a general disjunction $(\pi, \pi_0) \in \mathbb{Z}^d \times \mathbb{R}^{n-d} \times \mathbb{Z}$, there are some obvious trade-offs in the decision to branch or to generate valid inequalities. Branching on a poor disjunction will create two subproblems that may be as difficult as the problem under

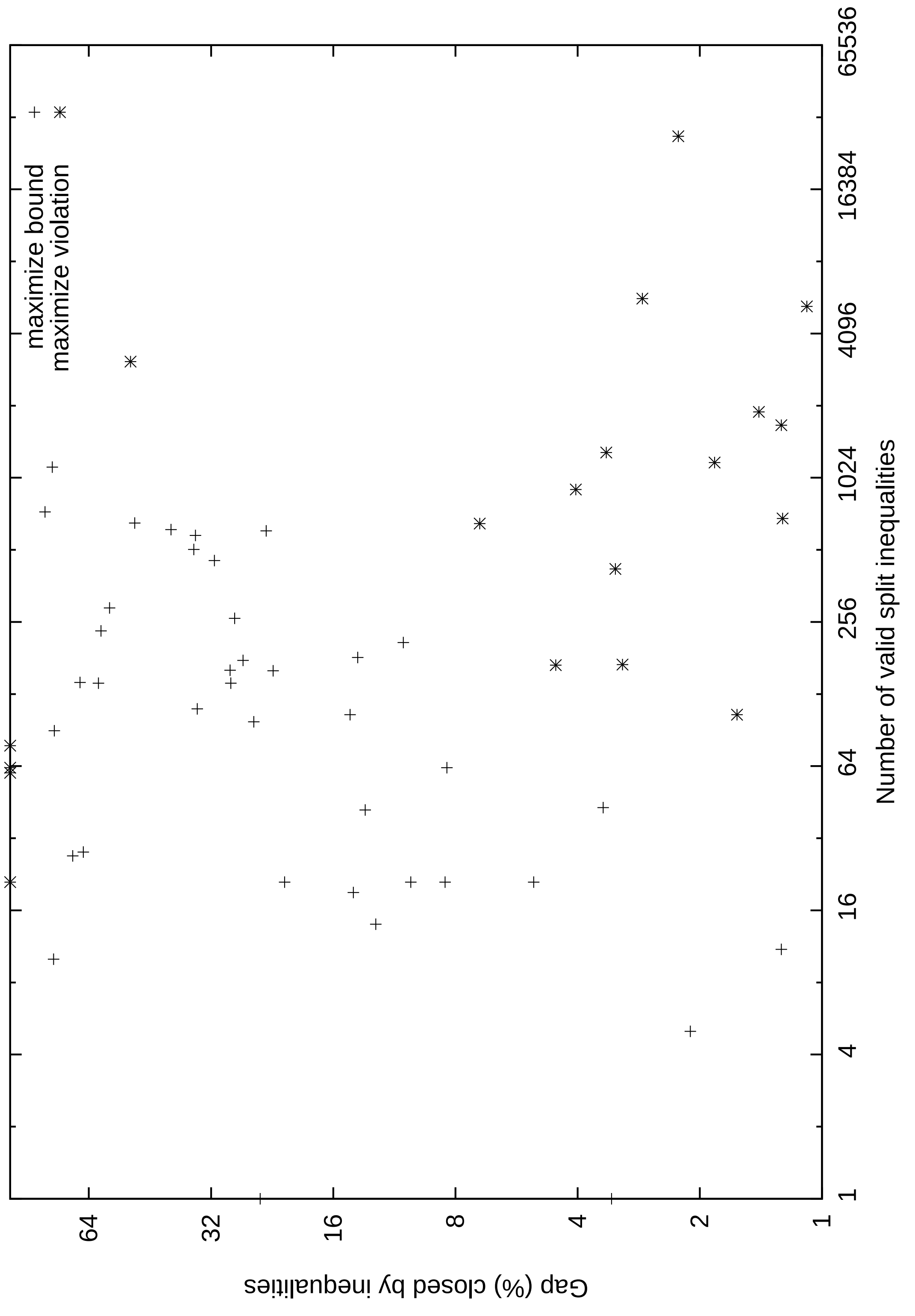


Figure 4.3: Plot of number of split inequalities generated and the gap closed by them.

consideration and thus double the time to solve. On the other hand, generating too many valid inequalities from a good disjunction may make the LP relaxation difficult to solve and generating too few inequalities may not lead to full benefits from imposing the disjunction. Another difference between branching and generating valid inequalities from a given disjunction is that of repetition. Once a disjunction has been used for branching, it will never be used again in any of the subproblems. However, the same disjunction may be used again to generate more inequalities after valid inequalities obtained from other disjunctions have been added. It is not apparent how such disjunctions should be used.

The decision to branch or to generate valid inequalities may also affect the choice of disjunctions in the subsequent iterations and this makes the decision problem even more complex. The myopic criterion of maximum bound improvement is theoretically not applicable to this problem, because one can get the same improvement with branching and generating valid inequalities from a given disjunction. The improvement in the bound, the number of iterations and the running time can be different when several disjunctions are used iteratively and it is precisely this difference that we wish to study. Toward this goal, we performed a simple experiment to study the difference in the performance of the CPLEX solver when disjunctions obtained by previously mentioned methods are for the two techniques.

In our experiment, we solved the instances by using the branch-and-bound algorithm of CPLEX with two different strategies. In the first method, we identified general disjunctions that improve the lower bound by the maximum using the methods of Section 3.2 and then used those for branching. We also saved the disjunctions for the second method. We tried to generate these disjunctions for all subproblems associated with the first k levels of the branch-and-bound tree only. All other branching decisions were done using the solver's default method. In the second method, we used the general disjunctions generated in the first to generate valid inequalities in the root node instead of using them to branch. We continued to add valid inequalities derived from these disjunctions until we could not add

Table 4.1: Time (in seconds) and number of nodes used when the same disjunctions are used for branching (bnb) and for generating valid inequalities in the root node(bnc).

Instance	# disj.	# cuts	time (disj)	time (bnb)	time (bnc)	# nodes (bnb)	# nodes (bnc)
10teams	1	5	3602.05	12.30	2260.91	1187	126837
bc1	9	18	3823.03	129.21	7180.90†	2001	89177 †
bell3a	6	130	2.92	3.16	4.93	19736	16395
bienst1	13	250	8923.97	70.77	256.24	22318	12663
bienst2	15	257	13942.31	411.39	4223.55	93670	148905
dcmulti	12	555	2490.92	0.41	32.44	1014	960
egout	5	5	2.86	0.03	0.04	299	287
gesa2	1	276	9085.47	7.51	387.37	13108	190285
gesa2_o	4	349	4666.20	14.62	4161.61	23590	308324
lseu	17	157	1160.31	0.32	1.64	3175	6174
misc03	2	1	3.20	0.36	0.41	684	853
neos11	17	149	13534.66	1162.16	3170.10	15484	10506
neos20	13	11	324.03	211.74	748.33	72346	184286
neos7	5	1	290.18	20935.84	10798.49†	5161360	2372189†
neos8	4	1	2070.84	81.81	63.30	6	46
nug08	3	7	2906.38	6.60	39.61	4	554
qnet1	7	155	2306.29	1.86	12.31	363	157
qnet1_o	12	289	5200.83	1.53	42.32	638	489
rout	6	339	4649.99	1155.17	7197.67†	1788298	601802†
seymour1	5	271	15258.85	861.94	1745.10	10271	12003
vpm1	5	7	193.42	0.03	0.02	79	31
vpm2	9	1329	1896.94	64.32	3600.80	226790	315397

†Result obtained after stopping the solver because of time limit

any more or we hit a time limit (1000s in our experiment), at which point we continued with the default branch-and-bound strategy. We compare the time spent and the number of subproblems solved in each case. We do not include the time used in identifying the disjunctions since the same disjunctions are used in both methods. In both methods, we let the solver do branch-and-bound for 7200s in addition to the time spent in identifying the disjunctions and the valid inequalities.

Table 4.1 shows the comparison between the two methods in terms of the time spent and the number of subproblems solved in the branch-and-bound stage. Even though the experiment was performed on all 177 instances listed in Table A.1, we report only 22 instances because on all other instances, either no disjunctions were found in the time limit or both the methods reached the time limit in branch and bound. For the problems listed in Table 4.1, the second method of generating valid inequalities in the root node seems to take much more time than simply using the disjunctions for branching. This is especially true for the case of 10teams, where the time spent and the number of subproblems solved when using valid inequalities differs by two orders of magnitude. We also observe that even though generation of valid inequalities by solving the CGLP consistently leads to longer running times, the number of subproblems solved is reduced for some instances (bell3a, bienst1, dcmulti, egout, neos11, qnet1, qnet1_o, vpm1).

While the experiment provides strong evidence that the general disjunctions that we obtain are more suitable for branching than for generating valid inequalities, the reasons are not particularly clear. In particular, we do not know whether this is because of the inequalities generated using the CGLP or was it because of the way the experiment is set up (we do not know of any other documented experiments of this kind).

4.5 Conclusions

The computational experiments performed in this chapter provide strong evidence that generating valid inequalities from disjunctions that improve the bound is more effective than the existing techniques of generating inequalities using the criterion of maximum violation. The time taken in identifying such disjunctions using a MIP solver is, like that observed in Chapter 3, exorbitant and fast heuristics are required to mitigate this problem. Our experiments also suggest that these disjunctions are more beneficial for branching than for generating valid inequalities, even though the reasons for these observations are not fully clear. The bigger question of whether branching or generation of valid inequalities is more useful for a given problem or even a given subproblem obtained in a branch-and-cut algorithm arises naturally in this context. This is discussed in more detail in the next chapter on future work.

Chapter 5

Conclusions and Future Research

In this thesis, we investigated several problems concerning identification of good disjunctions from the rich class of general disjunctions. We applied the criterion of maximum bound improvement that has previously been used for selecting variable disjunctions to the problem of selecting general disjunctions. The problem of selecting the best general disjunction for this criterion turns out to be $\mathcal{NP}$-hard. This result in turn leads to more results on the computational complexity of selecting and identifying elementary split inequalities. An analogous question regarding the computational complexity of problems when applied to the special case of C-G inequalities are still open.

Problem 14. *Given a mathematical program of the form (1.1) and $K \in \mathbb{R}$, does there exist a single elementary C-G inequality for (1.1) such that the LP relaxation bound achieved after adding it is at least K?*

In Chapter 2, we also developed a polynomial time algorithm to find a split inequality that can separate a point on an edge of the feasible region of the LP relaxation of a MIP. The computational complexity of the problem for finding a C-G inequality with the same quality is also open. We formally define the problem as follows.

Problem 15. *Given a MIP with the feasible region of LP relaxation $\mathcal{P}$, two adjacent* *extreme points v^1, v^2 of $\mathcal{P}$ and a point $\hat{x} \in \mathbb{Q}^n$ on the edge joining v^1, v^2, find an elementary C-G inequality that separates $\hat{x}$ from the C-G closure of $\mathcal{P}$ or show that none exists.*

A natural extension of Proposition 2.6.1 is to find conditions under which a general disjunction could be used to separate more more than two points simultaneously. Two such problems are as follows.

Problem 16. *Given three points $x^1, x^2, x^3 \in \mathbb{Q}^n$ does there exist a disjunction $(\hat{\pi}, \hat{\pi}_0)$ such that $x^1, x^2, x^3 \in \{x \in \mathbb{R}^n \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$.*

Problem 17. *Given k points $x^1, x^2, \ldots, x^k \in \mathbb{Q}^n$ does there exist a disjunction $(\hat{\pi}, \hat{\pi}_0)$ such that $x^1, x^2, x^3, \ldots, x^k \in \{x \in \mathbb{R}^n \mid \hat{\pi}_0 < \hat{\pi}x < \hat{\pi}_0 + 1\}$.*

In our computational experiments, we studied the impact on the branch-and-cut algorithm of using those general disjunctions that maximize the improvement in lower bound. The use of such disjunctions for branching seems to reduce dramatically the number of iterations when compared to the existing techniques like strong-branching using variable disjunctions and branching on thinnest directions. Analogous results were observed when the disjunctions that maximize bound improvement are used to generate split inequalities and C-G inequalities. Much fewer number of valid inequalities are required to close the gap between upper and lower bounds of the MIP when using our criterion as compared to the existing criterion of maximum violation. That such disjunctions can improve the performance of branch-and-cut algorithms drastically is clear from our experiments. However, we still do not know of any good heuristics that may be employed to identify these disjunctions. We believe that developing heuristic methods for this problem is a rich and fruitful line of research.

Another interesting direction for future research towards finding good general disjunctions quickly is to consider only restricted subsets, much like variable disjunctions, but more flexible. The experiments using disjunctions with two variables were a step in this direction. Our elimination procedure improves the time taken to enumerate all such disjunctions by one to two orders of magnitude. We think that with a few more such simple refinements and enhancements one can significantly improve the performance of the available state-of-the-art MIP solvers.

A natural question that arises in any implementation of branch-and-cut algorithm is that of deciding whether to branch or generate valid inequalities. The most popular technique is that of keep generating valid inequalities until some stopping criteria are met at which point, the solvers resort to branch-and-bound. The commonly used conditions for stopping are no further improvement in bound, the number of simplex iterations used to solve the LP, distance of the new optimal solution from that of the previous iteration etc. However, such criteria can often severely affect the performance by either branching too early or too late. Moreover, valid inequalities may again become useful in later iterations of branch-and-bound. This especially is important when the size of the branch-and-bound tree gets large as is often the case for difficult instances. In this thesis, we took a small step towards answering this question by comparing branching against generation of valid inequalities from the same disjunctions. While the experiment gave some good insight on this problem, it will seemingly take substantial theoretical and experimental research to settle this question.

Bibliography

K. Aardal and F. Eisenbrand. Integer programming, lattices and results in fixed dimenstion. Technical report, Probability, Networks and Algorithms, 2004.

T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operation Research Letters*, 33:42–54, 2005.

T. Achterberg, T. Koch, and A. Martin. Miplib 2003. *Operations Research Letters*, 34(4): 1–12, 2006.

K. Andersen, G. Cornuéjols, and Y. Li. Reduce-and-split cuts: Improving the performance of mixed integer gomory cuts. *Management Science*, 51(11):1720–1732, 2005.

K. Andersen, Q. Louveaux, R. Weismantel, and L. Wolsey. Inequalities from two rows of a simplex tableau. In M. Fischetti and D. Williamson, editors, *IPCO 2007, LNCS 4513*, pages 1–15, 2007.

E. Balas. Intersection cuts – a new type of cutting planes for integer programming. *Operations Research*, 19(1):19–39, 1971.

E. Balas. Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics*, 89:3–44, 1998.

E. Balas and A. Saxena. Optimizing over the split closure. *Mathematical Programming, Series A*, 2007. To appear.

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

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

W. Banaszczyk, A. E. Litvak, A. Pajor, and S. J. Szarek. The flatness theorem for nonsymmetric convex bodies via the local theory of banach spaces. *Mathematics of Operations Research*, 24(3):728–750, 1999.

E. M. L. Beale and J. A. Tomlin. Special facilities in general mathematical programming system for non-convex problems using ordered sets of variables. In J. Lawrence, editor, *Proceedings of the Fifth International Conference on Operations Research*, pages 447–454, 1970.

M. Benichou, J. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and O. Vincent. Experiments in mixed–integer linear programming. *Mathematical Programming*, 1:76–94, 1971.

D. Bertsimas and J. N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, Massachusetts, 1997.

R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. Savelsbergh. An updated mixed integer programming library: Miplib 3. Technical Report TR98-03, Rice University, 1998.

R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. Mip: Theory and practice - closing the gap, 2000. In M. J. D. Powell and S. Scholtes, editors, System Modelling and Optimization: Methods, Theory, and Applications, pages 19–49. Kluwer Academic Publishers.

A. Caprara and A. N. Letchford. On the separation of split cuts and related inequalities. *Mathematical Programming, Series B*, 94:279–294, 2003.

CBC-2.1. Available from https://projects.coin-or.org/cbc.

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

W. Cook, H. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization.* John Wiley, 1998.

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

G. Cornuéjols and F. Margot. On the facets of mixed integer prgrams with two integer variables and two constraints. *Mathematical Programming*, 2009. Series A.

G. Cornuéjols, L. Liberti, and G. Nannicini. Improved strategies for branching on general disjunctions. Working Paper, 2008.

I. Derpich and J. R. Vera. Improving the efficiency of branch and bound algorithm for integer programming based on "flatness" information. *European Journal of Operational Research*, 174:92–101, 2006.

E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

N. J. Dribeek. An algorithm for the solution of mixed integer programming problems. *Management Science*, pages 576–587, 1966.

J. Eckstein. Parallel branch-and-bound algorithms for general mixed integer programming on the cm-5. *SIAM journal on optimization*, 4(4):794–814, 1994.

F. Eisenbrand. Note on the membership problem for the elementary closure of a polyhedron. *Combinatorica*, 19(2):297–300, 1999.

M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–47, 2003. Series B.

M. Fischetti and A. Lodi. *Integer Programming and Combinatorial Optimization*, chapter Optimizing over the First Chvátal Closure, pages 12–22. Springer Berlin / Heidelberg, 2005.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co., 1979.

R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of American Mathematical Society*, 64:275–278, 1958.

Z. Gu, G. Nemhauser, and M. Savelsbergh. Lifted cover inequalities for 0-1 integer programs: Computation. *INFORMS Journal on Computing*, 10:427–437, 1998.

G. Hadley. *Linear Programming*. Addison-Wesley Publishing Company, 1961.

J. H.W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

ILOG CPLEX-10.2. Available from http://www.ilog.com/products/cplex.

R. Jeroslow. Trivial integer programs unsolvable by branch and bound. *Mathematical Programming*, 6:105–109, 1974.

M. Karamanov and G. Cornuéjols. Branching on general disjunctions. Working Paper, 2007.

B. Krishnamoorthy. Is Thinner Better. Working Paper, 2008.

B. Krishnamoorthy and G. Pataki. Column basis reduction and decomposable knapsack problems, 2006. Submitted, available at http://www.optimization-online.org/DB_HTML/2007/06/1701.html.

A. Land and A. Doig. An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, 28(3):497–520, 1960.

P. Liberatore. On the complexity of choosing the branching literal in DPLL. *Artificial intelligence*, 116(1-2):315–326, January 2000.

J. Linderoth and M. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11:173–187, 1999.

H. Mittelmann. Mixed integer LP problems, 2008. Available at http://plato.la.asu.edu/ftp/milp/.

G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc., 1988.

G. L. Nemhauser and L. A. Wolsey. A recursive procedure to generate all cuts for 0-1 mixed integer programs. *Mathematical Programming*, 46:379–390, 1990.

J. H. Owen and S. Mehrotra. Experimental results on using general disjunctions in branch-and-bound for general-integer linear programs. *Computational optimization and applications*, 20(2), November 2001.

J. Patel and J. W. Chinneck. Active constraint variable ordering for faster feasibility of mixed integer linear programs. *Mathematical Programming*, 110(3):445–474, September 2007.

T. J. Schaefer. The complexity of statisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226, 1978.

SCIP-1.0. Available from http://scip.zib.de/.

A. Sebő. An introduction to empty lattice simplices. In *Proceedings of the 7th International IPCO Conference on Integer Programming and Combinatorial Optimization*, LNCS, pages 400–414, 1999.

SYMPHONY-5.1. Available from http://www.branchandcut.org/SYMPHONY.

Xpress-Optimizer-v18.10.00. Available from http://www.dashoptimization.com.

Appendix A

Instances

Table A.1: Number of constraints, variables, integer variables, binary variables and non-zero coefficients in the 177 instances used in experiments.

Instance	Constraints	Variables	Integer	Binary	Non-Zeros
10teams	230	2025	1800	1800	12150
30_05_100	12050	10772	10771	10771	45889
30_95_100	12526	10976	10975	10975	46640
30_95_98	12471	10990	10989	10989	46387
a1c1s1	3312	3648	192	192	10178
A2C1S1	3312	3648	192	192	10178
acc0	1737	1620	1620	1620	7290
acc1	2286	1620	1620	1620	12978
acc2	2520	1620	1620	1620	15327
acc3	3249	1620	1620	1620	16785
acc4	3285	1620	1620	1620	17073
acc5	3052	1339	1339	1339	16134
aflow30a	479	842	421	421	2091
aflow40b	1442	2728	1364	1364	6783
air03	124	10757	10757	10757	91028
air04	823	8904	8904	8904	72965
air05	426	7195	7195	7195	52121
arki001	1048	1388	538	415	20439
atlanta-ip	21732	48738	46773	46667	257532
B1C1S1	3904	3872	288	288	11408
B2C1S1	3904	3872	288	288	11408

Continued on next page

Instance	Constraints	Variables	Integer	Binary	Non-Zeros
bc1	1913	1751	252	252	276842
bell3a	123	133	71	39	347
bell5	91	104	58	30	266
berlin_5_8_0	1532	1083	794	794	4507
bg512142	1307	792	240	240	3953
biella1	1203	7328	6110	6110	71489
bienst1	576	505	28	28	2184
bienst2	576	505	35	35	2184
binkar10_1	1026	2298	170	170	4496
blend2	274	353	264	231	1409
blp-ar98	1128	16021	15806	15806	200601
blp-ic97	923	9845	9753	9753	118149
blp-ic98	717	13640	13550	13550	191947
blp-ir98	486	6097	6031	6031	79152
cap6000	2176	6000	6000	6000	48243
CMS750_4	16381	11697	7196	7196	44903
core2536-691	2539	15293	15284	15284	177739
core2586-950	2589	13226	13215	13215	104678
core4284-1064	4287	21714	21705	21705	245121
core4872-1529	4875	24656	24645	24645	218762
dano3_3	3202	13873	69	69	79655
dano3_4	3202	13873	92	92	79655
dano3_5	3202	13873	115	115	79655
dano3mip	3202	13873	552	552	79655
danoint	664	521	56	56	3232
dc1c	1649	10039	8380	8380	121158
dc1l	1653	37297	35638	35638	448754
dcmulti	290	548	75	75	1315
dg012142	6310	2080	640	640	14795
disctom	399	10000	10000	10000	30000
dolom1	1803	11612	9720	9720	190413
dsbmip	1182	1886	192	160	7366
ds	656	67732	67732	67732	1024059
egout	98	141	55	55	282
enigma	21	100	100	100	289
fast0507	507	63009	63009	63009	409349
fiber	363	1298	1254	1254	2944
fixnet6	478	878	378	378	1756
flugpl	18	18	11	0	46
gen	780	870	150	144	2592
gesa2	1392	1224	408	240	5064
					Continued on next page

Instance	Constraints	Variables	Integer	Binary	Non-Zeros
gesa2_o	1248	1224	720	384	3672
gesa3	1368	1152	384	216	4944
gesa3_o	1224	1152	672	336	3624
glass4	396	322	302	302	1815
gt2	29	188	188	24	376
harp2	112	2993	2993	2993	5840
khb05250	101	1350	24	24	2700
l152lav	97	1989	1989	1989	9922
liu	2178	1156	1089	1089	10626
lrn	8491	7253	2455	2455	46123
lseu	28	89	89	89	309
m20-75-1	445	520	500	75	28270
m20-75-2	445	520	500	75	28270
m20-75-3	445	520	500	75	28270
m20-75-4	445	520	500	75	28270
m20-75-5	445	520	500	75	28270
manna81	6480	3321	3321	18	12960
markshare1_1	6	62	45	45	312
markshare1	6	62	50	50	312
markshare2_1	7	74	54	54	434
markshare2	7	74	60	60	434
mas74	13	151	150	150	1706
mas76	12	151	150	150	1640
misc03	96	160	159	159	2053
misc06	820	1808	112	112	5859
misc07	212	260	259	259	8619
mitre	2054	10724	10724	10724	39704
mkc	3411	5325	5323	5323	17038
mod008	6	319	319	319	1243
mod010	146	2655	2655	2655	11203
mod011	4480	10958	96	96	22254
modglob	291	422	98	98	968
momentum1	42680	5174	2349	2349	103198
momentum2	24237	3732	1809	1808	349695
momentum3	56822	13532	6599	6598	949495
msc98-ip	15850	21143	20290	20237	92918
mzzv11	9499	10240	10240	9989	134603
mzzv42z	10460	11717	11717	11482	151261
neos10	46793	23489	23489	23484	251197
neos11	2706	1220	900	900	9360
neos12	8317	3983	3136	3136	25781

Continued on next page

Instance	Constraints	Variables	Integer	Binary	Non-Zeros
neos13	20852	1827	1815	1815	253842
neos1	5020	2112	2112	2112	21312
neos20	2446	1165	967	937	7428
neos21	1085	614	613	613	12503
neos22	5208	3240	454	454	12312
neos23	1568	477	232	232	4284
neos2	1103	2101	1040	1040	7326
neos3	1442	2747	1360	1360	9580
neos4	38577	22884	17172	17172	99930
neos5	63	63	53	53	2016
neos648910	1491	814	748	748	4121
neos6	1036	8786	8340	8340	251946
neos7	1994	1556	454	434	5304
neos818918	2450	2750	50	50	9730
neos823206	709	1830	1720	1720	6362
neos8	46324	23228	23228	23224	313180
neos9	31600	81408	2099	2099	244224
net12	14021	14115	1603	1603	80384
noswot	182	128	100	75	735
NSR8K	6284	38356	32040	32040	371608
nsrand_ipx	735	6621	6620	6620	223261
nsrand-ipx	735	6621	6620	6620	223261
nug08	912	1632	1632	1632	7296
nw04	36	87482	87482	87482	636666
opt1217	64	769	768	768	1542
p0033	16	33	33	33	98
p0201	133	201	201	201	1923
p0282	241	282	282	282	1966
p0548	176	548	548	548	1711
p2756	755	2756	2756	2756	8937
pk1	45	86	55	55	915
pp08aCUTS	246	240	64	64	839
pp08a	136	240	64	64	480
prod1	208	250	149	149	5350
prod2	211	301	200	200	10501
protfold	2112	1835	1835	1835	23491
qap10	1820	4150	4150	4150	18200
qiu	1192	840	48	48	3432
qnet1	503	1541	1417	1288	4622
qnet1_o	456	1541	1417	1288	4214
rail507	509	63019	63009	63009	468878

Continued on next page

Instance	Constraints	Variables	Integer	Binary	Non-Zeros
railway_8_1_0	2527	1796	1177	1177	7098
ran10x26	296	520	260	260	1040
ran12x21	285	504	252	252	1008
ran13x13	195	338	169	169	676
rd-rplusc-21	125899	622	457	457	852384
rentacar	6803	9557	55	55	41842
rgn	24	180	100	100	460
roll3000	2295	1166	738	246	29386
rout	291	556	315	300	2431
set1ch	492	712	240	240	1412
seymour1	4944	1372	451	451	33549
seymour	4944	1372	1372	1372	33549
siena1	2220	13741	11775	11775	258915
sp97ar	1761	14101	14101	14101	290968
sp97ic	1033	12497	12497	12497	316629
sp98ar	1435	15085	15085	15085	426148
sp98ic	825	10894	10894	10894	316317
stein27	118	27	27	27	378
stein45	331	45	45	45	1034
stp3d	159488	204880	204880	204880	662128
swath2	884	6805	2406	2406	34965
swath3	884	6805	2706	2706	34965
swath	884	6805	6724	6724	34965
t1717	551	73885	73885	73885	325689
timtab1	171	397	171	64	829
timtab2	294	675	294	113	1482
tr12-30	750	1080	360	360	2508
trento1	1265	7687	6415	6415	93571
UMTS	4465	2947	2874	2802	23016
usAbbrv.8.25_70	3291	2312	1681	1681	9628
van	27331	12481	192	192	487296
vpm1	234	378	168	168	749
vpm2	234	378	168	168	917

Table A.2: Number of LPs solved and time taken when using Algorithm 3.1 (1) and Algorithm 3.2 (2) in the root node.

Instance	#LPs-1	#LPs-2	time-1 (s)	time-2 (s)
10teams	971450	1237	19783.60	64.90
aflow30a	50254	324	87.10	0.97
bell3a	64	981	0.02	0.42
blend2	4572	46	4.21	0.15
egout	1172	396	0.28	0.09
fiber	182468	306	152.83	0.52
flugpl	20	18	0.00	0.00
gen	9646	193	14.69	0.33
gesa2	32824	1494	73.92	3.86
gesa2_o	56928	1503	106.81	3.51
gt2	22	18	0.01	0.01
harp2	333636	347	308.46	1.63
khb05250	1102	353	0.68	0.27
l152lav	415000	244	1187.71	2.43
lseu	3668	56	0.57	0.02
mod008	6330	25	1.47	0.01
neos6	4107750	722	>25000	116.42
nug08	840	1066	193.88	132.38
nw04	527586	1056	24939.63	88.35
p0548	60968	1635	32.97	1.54
pp08aCUTS	7540	1986	9.81	3.21
qnet1	162756	148	355.70	0.90
qnet1_o	22	21	0.15	0.11
ran10x26	12184	95	7.53	0.11
ran12x21	16550	133	10.05	0.18
ran13x13	11498	150	5.11	0.11
rout	36322	296	86.23	1.42
stein45	2660	836	8.22	2.34
vpm1	10764	118	3.31	0.06
vpm2	17542	540	8.96	0.42

Table A.3: Number of valid C-G inequalities generated using two criteria: maximum violation (denoted as *vio*) and maximum improvement in bound (denoted as *imp*) and the gap closed by them.

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
10teams	0.00	0.00	0	0
30_05_100	0.00	0.00	0	0
30_95_100	100.00	100.00	0	0
30_95_98	0.00	0.00	0	0
A2C1S1	7.98	4.14	143	2
B1C1S1	13.97	10.83	376	2
B2C1S1	11.72	10.76	344	2
CMS750_4	0.00	0.00	0	0
NSR8K	0.00	0.00	0	0
UMTS	0.00	0.00	0	0
a1c1s1	8.58	4.02	149	2
acc0	100.00	100.00	15	0
acc1	100.00	100.00	15	0
acc2	100.00	100.00	15	0
acc3	100.00	100.00	1	0
acc4	100.00	100.00	0	0
acc5	100.00	100.00	0	0
aflow30a	5.72	13.82	17	2
aflow40b	1.25	3.43	15	2
air03	0.00	0.00	1	0
air04	0.00	0.00	0	0
air05	0.00	0.00	0	0
arki001	0.00	0.00	15	0
atlanta-ip	0.00	0.00	0	0
bc1	96.77	60.37	61	3
bell3a	35.38	27.38	679	4
bell5	0.26	83.25	7	3
berlin_5_8_0	0.00	0.00	12	0
bg512142	0.00	0.00	0	0
biella1	0.00	0.00	0	0
bienst1	0.00	0.00	0	0
bienst2	0.00	0.00	0	0
binkar10_1	1.95	0.00	15	0
blend2	2.60	0.00	15	1
blp-ar98	0.00	0.00	1	0
blp-ic97	0.00	0.00	0	0

Continued on next page

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
blp-ic98	0.00	0.00	3	0
blp-ir98	0.47	0.00	13	0
cap6000	0.00	0.00	0	0
core2536-691	0.00	0.00	0	0
core2586-950	0.00	0.00	0	0
core4284-1064	0.00	0.00	0	0
core4872-1529	0.00	0.00	0	0
dano3_3	0.01	0.01	0	0
dano3_4	0.00	0.00	0	0
dano3_5	0.00	0.00	0	0
dano3mip	0.00	0.00	0	0
danoint	0.00	0.00	0	0
dc1c	0.00	0.00	0	0
dc1l	0.00	0.00	0	0
dcmulti	19.68	33.10	15	4
dg012142	0.00	0.00	0	0
disctom	100.00	100.00	0	0
dolom1	0.00	0.00	0	0
ds	0.00	0.00	0	0
dsbmip	100.0	100.0	0	0
egout	49.79	60.48	394	18
enigma	100.00	100.00	22	0
fast0507	0.00	0.00	0	0
fiber	0.32	9.93	15	2
fixnet6	13.43	40.25	1476	6
flugpl	45.27	45.27	6	2
gen	3.92	14.78	15	2
gesa2	0.18	29.19	25	2
gesa2_o	0.32	1.79	39	2
gesa3	3.74	6.39	15	2
gesa3_o	0.00	3.74	15	2
glass4	0.00	0.00	0	0
gt2	2.90	90.87	44	5
harp2	0.00	0.00	0	0
khb05250	0.00	4.70	0	1
l152lav	0.00	0.00	0	0
liu	0.00	0.00	0	0
lrn	0.00	0.00	0	0
lseu	69.37	66.31	136	7
m20-75-1	0.24	0.53	162	2

Continued on next page

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
m20-75-2	0.35	0.47	199	2
m20-75-3	0.33	0.37	239	2
m20-75-4	0.24	0.44	142	2
m20-75-5	0.02	0.39	16	2
manna81	0.00	0.00	0	0
markshare1	0.00	0.00	0	0
markshare1_1	100.00	100.00	0	0
markshare2	0.00	0.00	0	0
markshare2_1	100.00	100.00	0	0
mas74	0.00	0.00	0	0
mas76	0.00	0.00	0	0
misc03	0.00	8.62	63	2
misc06	0.00	0.00	0	0
misc07	0.00	0.72	92	1
mitre	0.00	0.00	0	0
mkc	0.00	1.86	15	2
mod008	96.36	90.31	133	10
mod010	0.00	0.00	3	0
mod011	0.00	0.00	-1	0
modglob	0.00	0.00	0	0
momentum1	0.00	6.84	11	2
momentum2	0.00	0.00	0	0
momentum3	0.00	0.00	0	0
msc98-ip	0.00	0.00	0	0
mzzv11	0.00	0.00	0	0
mzzv42z	0.00	0.00	0	0
neos1	0.00	0.00	21	0
neos10	0.00	0.00	14	0
neos11	0.00	0.00	15	0
neos12	0.00	0.00	1	0
neos13	0.00	0.00	0	0
neos2	0.00	17.72	15	2
neos20	0.00	0.00	21	0
neos21	6.11	7.46	15	3
neos22	0.00	0.00	0	0
neos23	0.00	0.00	0	0
neos3	0.00	18.28	15	2
neos4	0.00	-0.00	0	1
neos5	4.17	4.17	1	1
neos6	100.00	100.00	1	0
Continued on next page				

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
neos648910	0.00	0.00	33	0
neos7	0.00	0.00	0	0
neos8	0.00	0.00	1	0
neos818918	0.00	0.00	28	0
neos823206	0.00	15.52	17	3
neos9	0.00	0.00	0	0
net12	0.00	0.00	3	0
noswot	0.00	0.00	29	0
nsrand-ipx	2.53	0.30	15	1
nsrand_ipx	2.11	0.95	15	1
nug08	0.00	0.00	0	0
nw04	0.00	0.00	0	0
opt1217	0.00	0.00	0	0
p0033	56.27	69.12	112	7
p0201	10.21	12.37	18	2
p0282	0.35	83.10	15	9
p0548	0.00	2.79	156	12
p2756	0.00	0.00	92	0
pk1	0.00	0.00	0	0
pp08a	3.96	3.98	55	4
pp08aCUTS	0.56	0.54	28	2
prod1	0.00	4.55	110	2
prod2	0.00	0.00	32	0
protfold	0.00	0.00	2	0
qap10	0.00	0.00	0	0
qiu	14.28	0.00	19	0
qnet1	1.61	0.85	14	1
qnet1_o	0.81	0.67	11	1
rail507	0.00	0.00	0	0
railway_8_1_0	0.00	0.00	1	0
ran10x26	3.28	6.18	15	3
ran12x21	2.72	6.35	15	3
ran13x13	8.28	14.17	27	4
rd-rplusc-21	0.00	0.00	20	0
rentacar	0.00	0.00	-1	0
rgn	0.00	0.00	0	0
roll3000	0.00	0.00	2	0
rout	0.00	4.91	16	2
set1ch	2.88	27.81	302	9
seymour	0.00	0.00	12	0

Continued on next page

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
seymour1	0.00	0.00	15	0
siena1	0.00	0.00	0	0
sp97ar	0.00	0.00	1	0
sp97ic	0.00	0.00	0	0
sp98ar	0.00	0.00	4	0
sp98ic	0.00	0.00	0	0
stein27	20.00	0.00	77	0
stein45	0.00	0.00	16	0
stp3d	0.00	0.00	0	0
swath	0.00	0.00	15	0
swath2	7.89	0.00	11	0
swath3	9.87	0.00	15	0
t1717	0.00	0.00	0	0
timtab1	0.00	7.66	54	0
timtab2	0.00	6.45	31	2
tr12-30	3.94	4.77	101	10
trento1	0.00	0.00	0	0
usAbbrv.8.25_70	0.00	0.00	26	0
van	2.65	0.00	13	0
vpm1	0.00	78.18	25	5
vpm2	31.88	33.75	178	5

Table A.4: Number of valid split inequalities generated using two criteria: maximum violation (denoted as *vio*) and maximum improvement in bound (denoted as *imp*) and the gap closed by them.

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
10teams	0.00	0.00	1	0
30_05_100	0.00	0.00	0	0
30_95_100	100.00	100.00	0	0
30_95_98	0.00	0.00	0	0
A2C1S1	0.04	0.05	1	0
B1C1S1	0.02	0.00	21	0
B2C1S1	0.00	0.00	64	0
CMS750_4	0.00	0.00	0	0
NSR8K	0.00	0.00	0	0
UMTS	0.00	0.00	0	0
a1c1s1	0.08	0.00	1	0
acc0	100.00	100.00	21	0
acc1	100.00	100.00	0	0
acc2	100.00	100.00	0	0
acc3	100.00	100.00	0	0
acc4	100.00	100.00	0	0
acc5	100.00	100.00	0	0
aflow30a	3.40	22.51	1305	160
aflow40b	0.22	0.00	22	0
air03	0.00	0.00	0	0
air04	0.00	0.00	0	0
air05	0.00	0.00	0	0
arki001	0.00	0.00	0	0
atlanta-ip	0.00	0.00	0	0
bc1	0.08	2.30	2	0
bell3a	6.97	49.32	659	663
bell5	0.29	0.00	708	0
berlin_5_8_0	0.00	0.00	208	0
bg512142	0.01	0.00	497	0
biella1	0.00	0.00	0	0
bienst1	0.00	0.00	1806	0
bienst2	0.00	0.00	1360	0
binkar10_1	0.17	0.00	34	0
blend2	0.00	0.00	194	0
blp-ar98	0.00	0.00	0	0
blp-ic97	0.00	0.00	0	0

Continued on next page

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
blp-ic98	0.00	0.00	0	0
blp-ir98	0.00	0.00	0	0
cap6000	0.00	0.00	0	0
core2536-691	0.00	0.00	0	0
core2586-950	0.00	0.00	0	0
core4284-1064	0.00	0.00	0	0
core4872-1529	0.00	0.00	0	0
dano3_3	0.01	0.01	0	0
dano3_4	0.00	0.00	0	0
dano3_5	0.00	0.00	0	0
dano3mip	0.00	0.00	0	0
danoint	0.18	0.00	861	0
dc1c	0.00	0.00	0	0
dc1l	0.00	0.00	0	0
dcmulti	1.43	12.57	1929	14
dg012142	0.00	0.00	0	0
disctom	100.00	100.00	0	0
dolom1	0.00	0.00	0	0
ds	0.00	0.00	0	0
dsbmip	100.10	100.10	-1	-1
egout	2.26	-1	27314	-1
enigma	100.00	100.00	60	0
fast0507	0.00	0.00	0	0
fiber	0.00	28.60	243	142
fixnet6	0.33	34.59	1519	111
flugpl	50.51	0.00	3129	0
gen	0.11	0.00	213	0
gesa2	0.04	8.40	617	63
gesa2_o	0.05	10.30	354	21
gesa3	0.54	0.00	427	0
gesa3_o	0.00	0.00	131	0
glass4	0.00	0.00	14	2
gt2	0.44	77.85	1917	90
harp2	0.00	0.00	0	0
khb05250	4.53	0.00	169	0
l152lav	0.17	0.00	7	0
liu	0.00	24.21	86	1
lrn	0.00	0.00	0	0
lseu	0.00	66.06	548	28
m20-75-1	0.02	0.02	265	4

Continued on next page

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
m20-75-2	0.05	0.01	347	0
m20-75-3	0.03	0.00	346	0
m20-75-4	0.04	2.11	322	5
m20-75-5	0.03	1.26	290	11
manna81	0.00	0.00	1	1420
markshare1	0.00	0.00	69	0
markshare1_1	100.00	100.00	63	0
markshare2	0.00	0.00	92	0
markshare2_1	100.00	100.00	78	0
mas74	0.00	0.00	0	0
mas76	0.00	0.00	0	0
misc03	0.00	10.75	77	210
misc06	0.00	0.00	26	0
misc07	0.00	0.71	83	21
mitre	0.00	0.00	0	0
mkc	0.00	0.00	0	0
mod008	4.04	78.77	914	1134
mod010	0.00	0.00	0	0
mod011	0.00	0.00	-1	-1
modglob	1.26	0.00	1696	0
momentum1	0.00	0.00	0	0
momentum2	0.00	0.00	0	0
momentum3	0.00	0.00	0	0
msc98-ip	0.00	0.00	0	0
mzzv11	0.00	0.00	0	0
mzzv42z	0.00	0.00	0	0
neos1	0.00	0.00	2	0
neos10	0.00	0.00	0	0
neos11	0.00	0.00	21	0
neos12	0.00	0.00	0	0
neos13	0.00	0.00	0	0
neos2	0.00	8.48	55	21
neos20	0.00	0.00	21	0
neos21	0.00	31.40	935	462
neos22	0.00	0.00	252	0
neos23	0.00	0.00	3130	0
neos3	0.00	5.13	13	21
neos4	4.47	4.47	0	0
neos5	3.23	13.93	426	182
neos6	100.00	100.00	0	0

Continued on next page

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
neos648910	0.00	0.00	270	0
neos7	0.00	0.00	0	0
neos8	0.00	0.00	0	0
neos818918	0.00	0.00	22	8
neos823206	0.00	0.00	16	0
neos9	0.00	0.00	0	0
net12	0.00	1.21	0	0
noswot	0.00	0.00	3999	0
nsrand-ipx	0.00	0.00	0	0
nsrand_ipx	0.00	0.00	0	0
nug08	0.00	0.00	0	0
nw04	0.00	0.00	0	0
opt1217	0.00	0.19	258	84
p0033	0.00	70.16	173	27
p0201	3.10	67.31	170	143
p0282	0.01	82.06	780	737
p0548	0.00	0.00	9	0
p2756	0.00	0.00	3	12
pk1	0.00	0.00	656	0
pp08a	2.77	28.73	5734	161
pp08aCUTS	1.09	13.34	5317	42
prod1	0.00	40.14	926	622
prod2	0.00	34.97	773	588
protfold	0.00	3.30	0	1
qap10	0.00	0.00	0	0
qiu	1.62	35.29	105	514
qnet1	0.02	21.08	13	21
qnet1_o	0.03	60.63	10	142
rail507	0.00	0.00	0	0
railway_8_1_0	0.00	0.00	52	21
ran10x26	1.25	25.12	692	98
ran12x21	0.97	14.28	1118	19
ran13x13	1.84	27.99	1185	265
rd-rplusc-21	0.00	0.00	0	0
rentacar	0.00	0.00	-1	-1
rgn	0.00	23.40	120	615
roll3000	0.00	0.00	0	0
rout	0.00	14.54	1015	105
set1ch	0.07	3.46	1575	43
seymour	0.00	0.00	88	0

Continued on next page

Instance	% gap closed		# generated	
	(vio)	(imp)	(vio)	(imp)
seymour1	0.00	0.00	158	0
siena1	0.00	0.00	0	0
sp97ar	0.00	0.00	0	0
sp97ic	0.00	0.00	0	0
sp98ar	0.00	0.00	0	0
sp98ic	0.00	0.00	0	0
stein27	0.00	0.00	705	0
stein45	0.00	0.00	666	0
stp3d	0.00	0.00	0	0
swath	0.00	0.00	0	0
swath2	0.00	0.00	0	0
swath3	0.00	0.00	0	0
t1717	0.00	0.00	0	0
timtab1	0.00	59.72	206	235
timtab2	0.00	26.68	234	177
tr12-30	0.13	0.10	324	2
trento1	0.00	0.00	0	0
usAbbrv.8.25_70	0.00	0.00	13	21
van	0.00	0.00	0	0
vpm1	0.00	78.18	251	10
vpm2	0.00	56.93	573	293

Biography

Name:	Ashutosh Mahajan.
Place of birth:	Kangra, India.
Date of birth:	June 08, 1981.
Parents:	Mrs. Venu Mahajan and Mr. Arun Mahajan.

Education

- Ph.D., Industrial Engineering, Lehigh University, Bethlehem, PA, August 2003 – May 2009.
- B.Tech. (Hons.), Production and Industrial Engineering, Indian Institute of Technology, Delhi, India, August 1999 – May 2003.

Professional Experience

- Graduate Research Assistant, Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, June 2008 – May 2009.
- Graduate Research Intern, Operations Research R & D, SAS Institute, Cary, NC, May 2007 – August 2007.
- High Performance Computing (HPC) Graduate Assistant, Library and Technology Services, Lehigh University, Bethlehem, PA, July 2006 – May 2008.

- Graduate Research Assistant, SAS Institute, Cary, NC and Industrial and Systems Engineering, Lehigh University, August 2005 – July 2006.
- Systems Administrator, COR@L Lab, Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, August 2004 – August 2008.
- Teaching Assistant Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, August 2003 – May 2004.
- Summer Intern, Heavy Vehicles Factory, Avadi, Chennai (India), May – July 2002.

Publications

- A. Mahajan and T.K. Ralphs, Experiments with Branching on General Hyperplanes, Proceedings of the 11th Informs Computing Society Conference, 2009.
- A. Mahajan and T.K. Ralphs, On the Complexity of Selecting Branching Disjunctions in Integer Programming, submitted, 2008.

Talks and Presentations

- A. Mahajan and T.K. Ralphs, On selecting general branching hyperplanes for mixed integer programs, INFORMS Annual Conference, Washington DC, October, 2008.
- A. Mahajan, M. Guzelsoy and T.K. Ralphs, SYMPHONY: A Mixed Integer Programming Solver, INFORMS Annual Conference, Washington DC, October, 2008.
- T. Ralphs, M. Guzelsoy, S. Oshkai and A. Mahajan, Warm Starting for Mixed Integer Linear Programs, INFORMS Annual Conference, Seattle, WA, November, 2007.

Memberships and Professional Activities

- Reviewer, Mathematical Programming Computation.
- Associate Member, Common Infrastructure for Operations Research (COIN-OR).

- Member, Beta Pi Chapter of Phi Beta Delta (Honor Society for International Scholars).

- Member, Institute for Operations Research and the Management Sciences (INFORMS) and INFORMS Computing Society (ICS).