THE EXISTENCE AND USEFULNESS OF EQUALITY CUTS IN THE MULTI-DEMAND MULTIDIMENSIONAL KNAPSACK PROBLEM

by

LEVI DELISSA

B.S., Kansas State University, 2014

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Industrial and Manufacturing Systems Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2014

Approved by:

Major Professor
Todd Easton

Abstract

Integer programming (IP) is a class of mathematical models useful for modeling and optimizing many theoretical and industrial problems. Unfortunately, IPs are $\mathcal{NP}$-complete, and many integer programs cannot currently be solved.

Valid inequalities and their respective cuts are commonly used to reduce the effort required to solve IPs. This thesis poses the questions, do valid equality cuts exist and can they be useful for solving IPs?

Several theoretical results related to valid equalities are presented in this thesis. It is shown that equality cuts exist if and only if the convex hull is not full dimensional. Furthermore, the addition of an equality cut can arbitrarily reduce the dimension of the linear relaxation.

In addition to the theory on equality cuts, the idea of infeasibility conditions are presented. Infeasibility conditions introduce a set of valid inequalities whose intersection is the empty set. infeasibility conditions can be used to rapidly terminate a branch and cut algorithm.

Applying the idea of equality cuts to the multi-demand multidimensional knapsack problem resulted in a new class of cutting planes named anticover cover equality (ACE) cuts. A simple algorithm, FACEBT, is presented for finding ACE cuts in a branching tree with complexity $O(m \cdot n \, log \, n)$.

A brief computational study shows that using ACE cuts exist frequently in the MDMKP instances studied. Every instance had at least one equality cut, while one instance had over 500,000. Additionally, computationally challenging instances saw an 11% improvement in computational effort. Therefore, equality cuts are a new topic of research in IP that is beneficial for solving some IP instances.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Integer Programming is a widely studied class of mathematical models. An Integer Program (IP) is defined as, maximize $\mathbf{c}^T\mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \in \mathbb{Z}_+^n$, where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, and $\mathbf{b} \in \mathbb{R}^m$. This research develops new techniques to improve the time required to solve integer programs by creating equality cuts and infeasibility conditions.

Integer programming has been used to model and solve many different problems in many different fields. Some of the fields where IPs have been applied are genetic research[5], portfolio management[3,17], transporting goods[8,18,19,21], and fighting cancer[11,12].

There are various classes of IPs and one of the most widely studied is called the Knapsack Problem (KP). KP models the classical analogy of a camper packing his/her knapsack before a trip. The camper can take items and each item has an associated benefit and weight. The camper wants to pack the knapsack in such a way that he/she has the maximum benefit, while still being able to carry the knapsack. The knapsack problem and variations of it, have numerous real world applications. Some examples include Merkle Hellman cryptography[14] and portfolio optimization[9].

A problem closely related to KP is the Demand Constraint Problem (DCP). DCP models a business deciding on a combination of marketing strategies. Each different strategy has a known cost and benefit. DCP seeks to minimize the cost spent while meeting certain market share requirements.

This thesis focuses on a particular class of IPs, called the Multi-Demand Multidimensional Knapsack Problem (MDMKP). The Demand Knapsack Problem contains both a knapsack and a demand constraint. Combining multiple demand constraints and multiple knapsack constraints results in the MDMKP. Formally, MDMKP has the form maximize $\mathbf{c}^T\mathbf{x}$ subject to $\mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{A}'\mathbf{x} \geq \mathbf{b}',$ and $\mathbf{x} \in \{0,1\}^n$, where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}_+^{m\times n}$, $\mathbf{A}' \in \mathbb{R}_+^{q\times n}$, $\mathbf{b} \in \mathbb{R}_+^m$, and $\mathbf{b}' \in \mathbb{R}_+^q$. The MDMKP has numerous real world applications. Some examples are project selection[20], capital budgeting[13], and cutting stock[4].

Unfortunately, IPs and MDMKPs are $\mathcal{NP}$-complete[7] and thus solving them can require exponential time. Some real world IPs still have no known solution. Additionally, even small IP instances are not guaranteed to be solvable within a reasonable time frame. This has motivated a significant amount of research into improving methods of solving IPs. The two most common solution methods are Branch and Bound[10] and cutting planes.

Branch and Bound, the primary method for solving IPs, works by solving the linear relaxation of each IP within the branching tree. The linear relaxation is the IP without the integer requirements. After solving a linear relaxation, branch and bound splits the problem of the parent node into two separate problems. This process is repeated until all the nodes have been fathomed. A node is fathomed if the solution at that node is integer, the linear relaxation at that node is infeasible, or the objective value at that node is worse than the objective value of the best integer solution found thus far. Once all nodes have been fathomed the best integer solution found is the optimal solution for the problem. If all the nodes have been fathomed and no integer solution was found, the problem is infeasible.

Solving IPs using cutting planes uses the idea of a valid inequality. The goal of a valid inequality, $\boldsymbol{\alpha}^T\mathbf{x} \leq \beta$, is to remove portions of the linear relaxation space without eliminating a feasible integer point. The theoretical strength of a valid inequality can be measured by its induced face. Facet defining inequalities are the strongest theoretical valid inequalities.

Much research has been done on finding cutting planes for KP instances. A popular method to generate cuts is to use covers[1,22] on a knapsack constraint. Cover cuts can be facet defining, are easy to find and are commonly implemented in modern software.

1.1 Motivation

The following question motivated this research. Why are valid inequalities defined as $\boldsymbol{\alpha}^T\mathbf{x} \leq \beta$? Would it be possible to have a valid equality, $\boldsymbol{\alpha}^T\mathbf{x} = \beta$? This research answers this latter question in the affirmative and demonstrates how to find such equality cuts for some MDMKP instances.

1.2 Contributions

This thesis presents the idea of an equality set and corresponding valid equality. It is shown that equality sets only exist when the IPs convex hull is not full dimensional. Additionally, it is shown that although equality cuts can never be facet defining, the addition of an equality cut can arbitrarily reduce the dimension of an IPs linear relaxation.

The idea of equality cuts is applied to MDMKP by utilizing covers and introducing the concept of an associated anticover inequality. In certain instances, there exists an anticover cover equality set. Several examples of equality cuts are presented. These examples demonstrate the existence of equality sets and provide substantial discussion regarding their potential benefit.

A third contribution of this research is a formal definition of infeasibility conditions in general. If an infeasibility condition exists for an IP, then the IP is infeasible. Infeasibility conditions using anticovers and covers for the MDMKP are presented. An example demonstrates that although the linear relaxation may be full dimensional, an infeasibility condition can exist. Thus, such an IP would not need to be solved by branch and bound.

The final contribution of this work is a computational study of both equality cuts and infeasibility conditions for the MDMKP. Implementing anticover cover equality cuts and anticover cover infeasibility conditions resulted in an average of about a 7% improvment on small benchmark instances and an 11% gain on large benchmark instances.

1.3 Outline

Chapter 2 presents basic concepts of integer programming and polyhedral theory along with other background information requisite for understanding the research presented in this thesis. Some of the topics covered in this chapter include the Integer Programming, Polyhedral Theory, the Knapsack Problem, the Demand Problem, the Multi-Demand Multidimensional Knapsack Problem, Cutting Planes, Branch and Cut, and Covers.

Chapter 3 contains the advancements of this research. The concept of an equality set with corresponding valid equality is introduced along with results about their existence and theoretical usefulness. An simple algorithm for finding anticover cover equality (ACE) cuts on the MDMKP is presented with its running time. Finally the idea of infeasibility conditions by combining cuts is introduced.

The computational results of applying ACE on MDMKP can be found in Chapter 4. This chapter presents the results, an interpretation, and some discussions relating to the implementation.

Chapter 5 summarizes this thesis and discusses several ideas for further research on valid equalities. Additional topics are provided on both the theoretical and computational side.

Chapter 2

Background Information

This chapter provides a brief discussion of material prerequisite to understand this thesis. Topics such as integer programming, the knapsack problem, polyhedral theory, cutting planes and covers are discussed. To study this material in greater detail, refer to the classic text by Nemhauser and Wolsey[15].

2.1 Integer Programming

Integer programs (IP's) are a class of mathematical models useful for solving optimization problems. An IP is of the form:

$$\begin{array}{rrcl} \text{Maximize} & \mathbf{c}^T\mathbf{x} & & \\ \text{Subject to} & \mathbf{A}\mathbf{x} & \leq & \mathbf{b} \\ & \mathbf{x} & \in & \mathbb{Z}^n_+ \end{array}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $\mathbf{b} \in \mathbb{R}^m$.

An optimal solution to an IP takes the form z^* and $\mathbf{x}^*$, where $z^* = \mathbf{c}^T\mathbf{x}^*$. The set of feasible solutions is denoted as $P = \{\mathbf{x} \in \mathbb{Z}^n_+ : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$. Furthermore, the set of indices is

$N = \{1, ..., n\}$.

Integer programming has been used to model and solve many different problems in many different fields. Some of the fields where IPs have been applied are genetic research[5], portfolio management[3,17], transporting goods[8,18,19,21], and fighting cancer[11,12]. There are numerous other applications for integer programs not mentioned here. Additionally, there exist many problems where integer programming would be useful, but due to the IP's size, complexity and the lack of efficient methods for solving IPs, solutions are obtained using other techniques with no guarantee of optimality.

Integer programs have been shown to be $\mathcal{NP}$-hard[7]. Thus, no polynomial time algorithm has yet been discovered to solve IPs and such an algorithm is unlikely to exist. IPs are typically solved by utilizing the linear relaxation. The linear relaxation (LR) of an IP is the IP without the integer constraint. Thus the LR takes the form

$$\begin{array}{rrcl} \text{Maximize} & \mathbf{c}^T\mathbf{x} & & \\ \text{Subject to} & \mathbf{A}\mathbf{x} & \leq & \mathbf{b} \\ & \mathbf{x} & \in & \mathbb{R}^n_+ \end{array}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $\mathbf{b} \in \mathbb{R}^m$. The linear relaxation space is defined as $P^{LR} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$. The optimal solution to the LR is denoted z^{*LR} and $\mathbf{x}^{*LR}$, where $z^{*LR} = \mathbf{c}^T\mathbf{x}^{*LR}$.

Since no polynomial time algorithm has yet been discovered to solve IPs, much research effort has been given to improving current solution techniques. The most common such methods are branch and bound[10] and cutting planes.

Branch and bound begins by solving the linear relaxation. If $\mathbf{x}^{*LR} \in \mathbb{Z}^n_+$, then $\mathbf{x}^{*LR}, z^{*LR}$ is an optimal integer solution and the algorithm terminates. Otherwise, the algorithm selects some non-integer variable $x_i = q$ for branching. The original problem is considered the parent node and each branch creates two child nodes. One child node contains the original problem plus an additional constraint $x_i \leq \lfloor q \rfloor$; while the other child node is the original problem plus the constraint $x_i \geq \lceil q \rceil$. The algorithm repeats this process until all nodes have been fathomed.

A node is fathomed if the LR is infeasible, has an integer solution, or z^{*LR} at that node is worse than the best integer solution found thus far. Once all the nodes have been fathomed the best integer solution from any node is known to be the global optimal solution. If no integer solution exists the IP is infeasible.

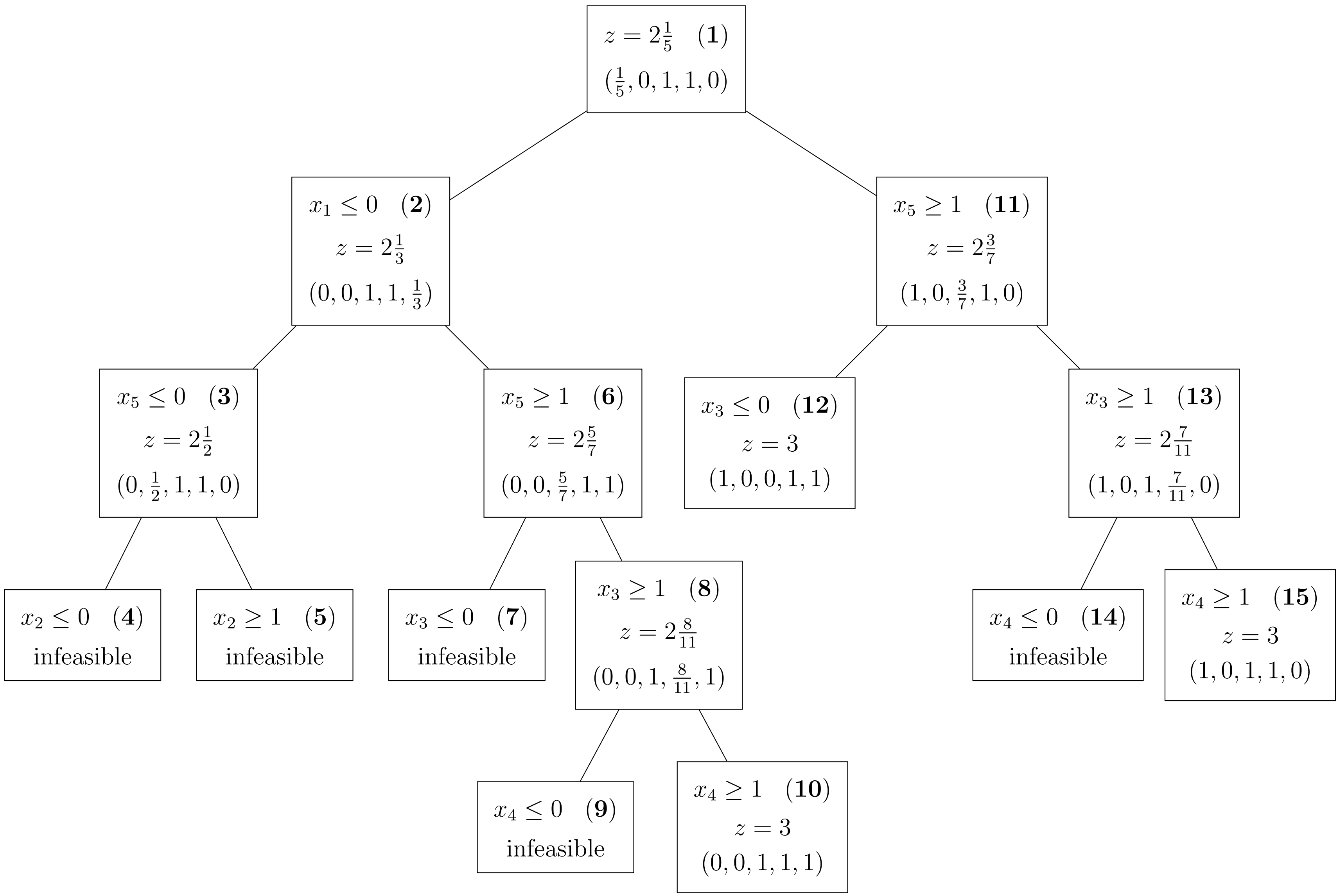
Consider the following branch and bound example.

Example 2.1.1. Given the following IP:

$$\begin{array}{rrcl}
\text{Minimize} & x_1 + x_2 + x_3 + x_4 + x_5 & & \\
\text{Subject to} & 3x_1 + 13x_2 + 7x_3 + 5x_4 + 11x_5 & \leq & 23 \\
& 5x_1 + 2x_2 + 7x_3 + 11x_4 + 3x_5 & \geq & 19 \\
& x_1, x_2, x_3, x_4, x_5 & \in & \{0, 1\}
\end{array}$$

In this example Branch and Bound uses a depth first left search strategy. Solving the linear relaxation at the root node (1) results in $z^* = 2\frac{1}{5}$ and $\mathbf{x}^* = (\frac{1}{5}, 0, 1, 1, 0)$. Since $\mathbf{x}^* \notin \mathbb{Z}^n$, branching begins. Node (2) is formed by adding the constraint $x_1 \leq 0$. Following the depth first left search strategy this process continues until node (4) which is infeasible. Thus the node is fathomed. The algorithm then returns to node (3) to find any unfathomed nodes. This logic is followed until all nodes have been fathomed. To see the full branching tree, which requires solving 15 linear relaxations, refer to Figure 2.1.

Figure 2.1: *A Branch and Bound Tree*



2.1.1 Knapsack Problems

A widely studied class of IPs is the Knapsack Problem (KP). The classical analogy depicts a camper packing his/her knapsack before a trip. There are n items that can be taken, every item j is associated with a benefit, c_j, and a nonnegative weight, a_j. The camper wants to pack the knapsack in such a way that he/she has the maximum benefit while still being able to carry the knapsack, a total weight less than or equal to b.

To model KP as an IP, let $x_j = 1$ if the camper selects item j; otherwise, $x_j = 0$. The IP formulation of the KP is,

$$\begin{array}{rrcl} \text{Maximize} & \mathbf{c}^T\mathbf{x} & & \\ \text{Subject to} & \mathbf{a^T x} & \leq & b \\ & \mathbf{x} & \in & \{0,1\}^n \end{array}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{a} \in \mathbb{R}^n_+$, and $b \in \mathbb{R}_+$.

A knapsack problem with more than one constraint is referred to as a multidimensional knapsack problem. Formally,

$$\begin{array}{rrcl} \text{Maximize} & \mathbf{c}^T\mathbf{x} & & \\ \text{Subject to} & \mathbf{Ax} & \leq & \mathbf{b} \\ & \mathbf{x} & \in & \{0,1\}^n \end{array}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}_+$, and $\mathbf{b} \in \mathbb{R}^m_+$.

Define the set of feasible solutions of a knapsack as $P_{KP} = \{x \in \{0,1\}^n : \mathbf{a^T x} \leq b\}$. Similarly, let $P_{MK} = \{\mathbf{x} \in \{0,1\}^n : \mathbf{Ax} \leq \mathbf{b}\}$ be the set of feasible solutions for the multidimensional knapsack problem.

An important generalization is that no item i has a weight a_i, such that $a_i > b$. In this case, the item can never be included so it is not allowed in the KP instance. This is important to show that KP is full dimensional. A proof of this is shown in the section on polyhedral theory.

The knapsack problem and variations of it, have numerous real world applications. Some examples include Merkle Hellman cryptography[14] and portfolio optimization[9].

2.1.2 Demand Constraint Problems

A class of problems closely associated with the knapsack problem is the Demand Constraint Problem (DCP). An analogy for the DCP depicts a business deciding on a combination of marketing strategies. Each different strategy j has a known cost c_j and an estimated increase in demand a'_j. The problem seeks to minimize the cost, while achieving a certain minimum level of product demand, b'.

To model this problem as an IP, let $x_j = 1$ if strategy j is used; otherwise, $x_j = 0$. The IP formulation is,

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}'^T\mathbf{x} \\ \text{Subject to} \quad & \mathbf{a'^T x} \geq b' \\ & \mathbf{x} \in \{0,1\}^n \end{aligned}$$

where $\mathbf{c}' \in \mathbb{R}^n$, $\mathbf{a}' \in \mathbb{R}^n_+$, and $b' \in \mathbb{R}_+$.

A demand problem with more than one constraint is referred to as the multidimensional demand constraint problem (MDCP). Formally,

$$\begin{aligned} \text{Minimize} \quad & \mathbf{c}'^T\mathbf{x} \\ \text{Subject to} \quad & \mathbf{A}'\mathbf{x} \geq \mathbf{b}' \\ & \mathbf{x} \in \{0,1\}^n \end{aligned}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m\times n}_+$, and $\mathbf{b} \in \mathbb{R}^m_+$.

Define the set of feasible solutions of a DCP as $P_{DCP} = \{\mathbf{x} \in \{0,1\}^n : \mathbf{a^T x} \leq b\}$.

Similarly let $P_{MDCP} = \{\mathbf{x} \in \{0,1\}^n : \mathbf{Ax} \geq \mathbf{b}\}$, be the set of feasible solutions for the multidimensional demand constraint problem.

Observe that any DCP can be transformed into a KP by substituting $(1 - x_i')$ for x_i. This substitution results in an objective function of Maximize $\mathbf{c}'^T\mathbf{x}' - \mathbf{c}'^T\mathbf{1}$. The demand constraint becomes $\mathbf{a}'^T\mathbf{x}' \leq \mathbf{a}'^T\mathbf{1} - b'$. Thus, the DCP is equivalent to the KP. A demand constraint formed using this substitution is called an associated demand constraint for the knapsack instance. Likewise, a knapsack constraint can be formed from a demand constraint using this substitution is called the associated knapsack constraint.

Due to this substitution, much less work has focused on DCP. Fortunately, results from KP can be trivially extended to DCP instances. Thus, DCP has the same real world applications as the KP and similar theoretical results. The converse is also true.

2.1.3 Demand Knapsack Problems

An extension of the KP and DCP is the Demand Knapsack Problem (DKP). A DKP is formed by adding a demand constraint to a knapsack problem. Formally,

$$\begin{array}{rrcl} \text{Maximize} & \mathbf{c}^t\mathbf{x} & & \\ \text{Subject to} & \mathbf{a^T x} & \leq & b \\ & \mathbf{a'^T x} & \geq & b' \\ & \mathbf{x} & \in & \{0,1\}^n \end{array}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{a}, \mathbf{a}' \in \mathbb{R}^n_+$, and $b, b' \in \mathbb{R}_+$.

Observe that the demand constraint in a DKP cannot be transformed into knapsack constraint. This is because the variable substitution of $(1 - x_j')$ for x_j merely flips which constraint is the demand constraint and which constraint is the knapsack constraint. Thus, the DKP problem is different from either the KP or DCP. Consequently, results for either

KP or DCP cannot automatically be applied to DKP instances.

A DKP with more than one knapsack or demand constraint is referred to as the Multi-Demand Multidimensional Knapsack Problem (MDMKP). The MDMKP is considered throughout this thesis. MDMKP has the following IP,

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^t\mathbf{x} \\ \text{Subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{A}'\mathbf{x} \geq \mathbf{b}' \\ & \mathbf{x} \in \{0,1\}^n \end{aligned}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}_+^{m\times n}$, $\mathbf{A}' \in \mathbb{R}_+^{q\times n}$, $\mathbf{b} \in \mathbb{R}_+^m$, and $\mathbf{b}' \in \mathbb{R}_+^q$.

Define the set of feasible solutions as $P_{DKP} = \{\mathbf{x} \in \{0,1\}^n : \mathbf{a^T x} \leq b, \mathbf{a'^T x} \geq b'\}$. Similarly let $P_{MDMKP} = \{\mathbf{x} \in \{0,1\}^n : \mathbf{Ax} \leq \mathbf{b}, \mathbf{A'x} \geq \mathbf{b}'\}$. This thesis focuses on P_{MDMKP}.

The MDMKP has numerous real world applications. Some examples include topics related to project selection[20], capital budgeting[13], and cutting stock[4].

2.2 Polyhedral Theory

Polyhedral theory is an area of research in mathematical programming and is used to study the feasible space for integer and linear programs. Relevant definitions from polyhedral theory are discussed in this section.

A linear relaxation of an IP is a linear program. The feasible space P^{LP} for a linear program is defined by a finite set of linear inequalities, written $P^{LP} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} \leq \mathbf{b}\}$, where $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$. The solution space for an inequality is $\{\mathbf{x} \in \mathbb{R}^n : \sum_{j=1}^{n} a_{ij}x_j \leq b_i\}$, and is called a halfspace. The set of all points P formed by the intersection of a finite number of halfspaces is a polyhedron. By definition, the feasible space of a linear program is a polyhedron. A polyhedron P with some bound k such that $|\mathbf{x}| \leq k$ for all $\mathbf{x} \in P$ is called a polytope.

A set $S \subseteq \mathbb{R}^n$ is convex if and only if $\lambda\mathbf{x} + (1-\lambda)\mathbf{x}' \in S$ for all $\mathbf{x}, \mathbf{x}' \in S$ and $\lambda \in [0,1]$. The convex hull of S, written $S^{CH} = conv(S)$, is the intersection of all convex sets S' such that $S \subseteq S'$. It can easily be seen that S^{CH} is convex. This thesis focuses on $P^{CH}_{MDMKP} = conv(\{\mathbf{x} \in \{0,1\}^n : \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{A}'\mathbf{x} \geq \mathbf{b}'\})$. As such, results are also provided relative to P^{CH}_{KP}, P^{CH}_{DC} and P^{CH}_{DKP}.

The feasible space for a linear program is a convex set and has dimension defined as the number of linearly independent vectors in P^{LP}. The feasible region of an IP, P, is only a collection of points so no vectors are feasible. Consequently, the dimension of P^{CH} is determined using affine independence. The intuition behind this is to take any one of the affinely independent feasible points in P and make linearly independent vectors from this point to the other affinely independent points that are also in P. Thus, these vectors must be contained in P^{CH}.

A collection of points $\mathbf{p}_1, \mathbf{p}_2, ...\mathbf{p}_r \in \mathbb{R}^n$ is affinely independent if $\sum\limits_{j=1}^{r} \lambda_j p_j = 0, \sum\limits_{j=1}^{r} \lambda_j = 0$ is uniquely solved by the trivial solution $\lambda_j = 0$ for all $j = 1, ..., r$. The dimension of the convex hull of an IP is defined as the number of affinely independent points minus one. To identify points in $\mathbb{R}^n$, define $\mathbf{e}_i$ to be the i^{th} identity point, the point that is translated one unit from the origin in the i^{th} dimension. Also let $\mathbf{0}$ and $\mathbf{1}$ be the vector of 0s and 1s, respectively.

A polyhedron $P^{LP} \subseteq \mathbb{R}^n$ is said to be full dimensional if $dim(P^{LP}) = n$. If P^{LP} is not full dimensional, then $dim(P^{LP}) < n$, and there is at least one inequality $\boldsymbol{\alpha}_j\mathbf{x} \leq \beta_j$ that is satisfied at equality by all points in P.

The following arguments demonstrate the concepts of the dimension of P^{CH}_{KP} and P^{CH}_{DC}. These arguments require some standard assumptions that are used throughout this thesis.

Recall that Section 2.1.1 argued that each a_i of a knapsack instance is less than or equal to b or x_i can be eliminated. In addition to this assumption, assume that the indices of all KP instances are sorted in descending order according to their $\mathbf{a}$ coefficients. With these assumptions, the dimension of KP is trivially bounded from above by n, due to the n variables. To show that the $\dim(P_{KP}^{CH})$ is $|N|$, there must be $|N|+1$ affinely independent points. Consider the points $\mathbf{0}, \mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_{|N|}} \in P_{KP}^{CH}$. These points are all affinely independent making the $dim(P_{KP}^{CH}) = |N|$.

For DC instances, assume that the indices are sorted in descending order according to the $\mathbf{a}'$ coefficients. If $\sum_{i=2}^{n} a_i' < b'$, then x_1 must equal to one for every feasible solution and it can be eliminated from the problem with the proper adjustments to b' and c. Thus, assume that $\sum_{i=2}^{n} a_i' \geq b'$. With this assumption, $\dim(P_{DC}^{CH}) = |N|$. Consider the points $\mathbf{1}$ and $\mathbf{1} - e_i$ for each $i \in N$. These points are affinely independent and clearly in P_{DC}, so $\dim(P_{DC}^{CH}) = |N|$.

Let $M = \{1, ..., m\}$ be the set of inequalities defining P^{LP} with the inequality set $M^{\leq} = \{j \in M : \boldsymbol{\alpha}_j \mathbf{x} < \beta_j$ for some $\mathbf{x} \in P^{LP}\}$ sometimes denoted as $(\mathbf{A}^{\leq}, \mathbf{b}^{\leq})$, and the equality set $M \setminus M^{\leq} = M^{=} = \{j \in M : \boldsymbol{\alpha}_j \mathbf{x} = \beta_j \ \ \forall \ \ \mathbf{x} \in P^{LP}\}$ sometimes denoted $(\mathbf{A}^{=}, \mathbf{b}^{=})$. A common result is that $dim(P^{LP}) + rank(\mathbf{A}^{=}, \mathbf{b}^{=}) = n$.

A fundamental result is that P^{CH} is a polyhedron and can be exactly defined by a finite number of inequalities. Furthermore, all of the extreme points of P^{CH} are integer. Thus, if a set of equalities is known that exactly defines P^{CH}, an optimal integer solution can be found in P^{CH} by solving an LP. This is the motivation for cutting planes.

2.2.1 Cutting Planes and Faces

If $P^{LR} = P^{CH}$, then a solution to the IP can be found in polynomial time. Cutting planes are used to remove points in $P^{LR} \setminus P^{CH}$. An inequality $\sum_{j \in N} \alpha_j x_j \leq \beta$ is said to be a valid inequality for P^{CH} if, and only if $\sum_{j \in N} \alpha_j x_j' \leq \beta$ is satisfied for every $\mathbf{x}' \in P$.

The set of all points in a polyhedron that exactly satisfy a valid inequality is defined as a face $F = \{\mathbf{x} \in P^{CH} : \sum_{j \in N} \alpha_j x_j = \beta\}$. If $F \subseteq P^{CH}$ and $F \neq \emptyset$, then F is said to support P^{CH}. A face is said to be proper if F supports P^{CH} and $F \neq P^{CH}$.

A facet defining inequality of P^{CH} defines a face of P^{CH} with dimension exactly one less than the dimension of P^{CH}. A face defined by a facet defining inequality is called a facet. All of the facets of P^{CH} are necessary and sufficient to define P^{CH}. The following example helps to explain these concepts.

Example 2.2.1. Given the integer program:

$$\begin{array}{rrcl} \text{Maximize} & x_1 + 2x_2 & & \\ \text{Subject to} & 3x_1 + 2x_2 & \leq & 12 \\ & 3x_1 + 4x_2 & \leq & 15 \\ & x_1, x_2 & \in & \mathbb{Z}_+ \end{array}$$

The first constraint, $3x_1 + 2x_2 \leq 12$, passes through points $(0, 6)$, C, and D. The second constraint, $3x_1 + 4x_2 \leq 15$, passes through the points A, B, C, and $(5, 0)$. The linear relaxation space of the IP is defined by these two constraints, the x_1 axis, and x_2 axis. The large circles represent the feasible integer points, P, of the problem as shown in Figure 2.2.

Clearly there is space within the linear relaxation that is outside the feasible integer points. The aim of a cutting plane is to remove this non-integer solution space without eliminating any of the feasible integer points. The dashed line represents the inequality $x_1 + x_2 \leq 4$ and passes through the points $(0, 4)$, B, and D. This cutting plane eliminates the region BCD of the linear relaxation space without cutting off any feasible integer points. Thus, it is a valid inequality and a cutting plane.

By finding the dimension of P^{CH} and the dimension of the cut's faces, this cutting plane can be classified as facet defining. The dimension of the polyhedron, P^{CH}, is 2, because it contains three affinely independent points $(0, 0)$, $(0, 1)$, and $(1, 0)$. Next the dimension of the cut's face is found by listing affinely independent points that meet the inequality at

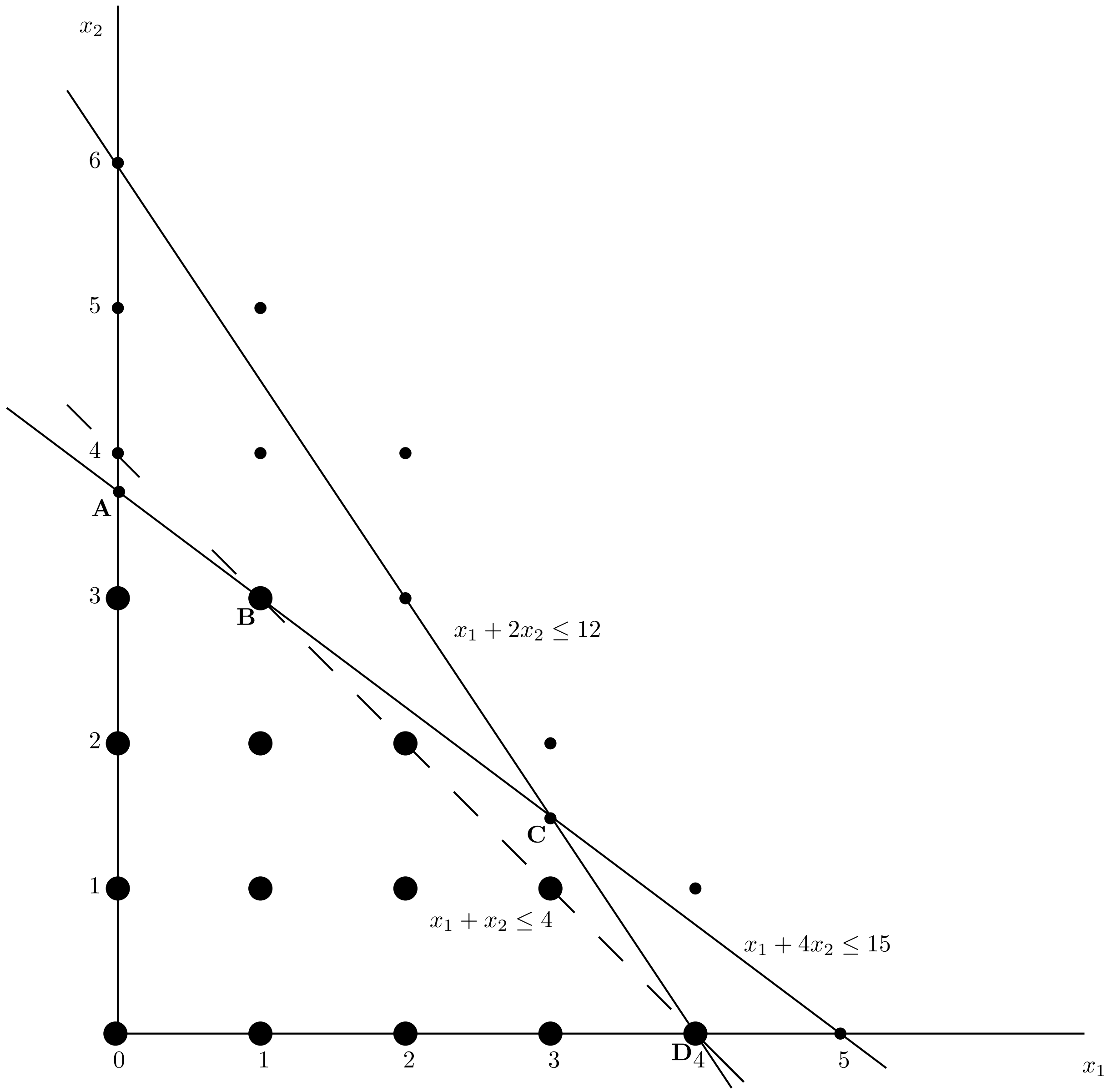


Figure 2.2: *Cutting Planes Example*

equality. In this case, $(1, 3)$ and $(0, 4)$ can be used as these points. Furthermore, it is evident that the face does not define P^{CH}. Consequently, $x_1 + x_2 \leq 4$ is a facet defining inequality.

For this example, the other facet defining inequalities are $x_1 \geq 0$, $x_2 \geq 0$, and $x_2 \leq 3$. If these inequalities are added, then P^{CH} is defined. Notice that all corner points are integer.

This example shows the simplicity of finding the valid inequalities in a two dimensional integer programming problem. As the number of variables increases, so does the complexity of finding valid inequalities and facet defining inequalities.

Combining the concepts of cutting planes within a branch and bound algorithm results in branch and cut[16]. Branch and cut follows the same idea as branch and bound. The difference is that at any unfathomed node, cutting planes may be included instead of a branch. If cutting planes are added, then only one child is created.

There are typically two classes of cutting planes, local and global cuts, added in branch and cut. Local cutting planes are applied to any node that is a descendant of this node. A global cut applies to all of the nodes of the tree. This research implements a branch and cut algorithm when solving MDMKPs using local cuts.

2.2.2 Covers

One of the most commonly implemented and studied cutting planes is derived from a cover on a knapsack constraint. A cover represents a collection of items that is too heavy for the hiker to carry. Thus, a cover represents an infeasible point.

Formally, given a knapsack constraint, a cover is a set $C \subseteq N$ such that $\sum_{i \in C} a_j > b$. Thus, setting $x_i = 1$ for all $i \in C$ is infeasible. A cover C has a corresponding valid inequality of the form, $\sum_{i \in C} x_i \leq |C| - 1$.

There are various classes of covers. A cover is said to be minimal if $\sum_{i \in C \setminus \{j\}} a_i \leq b$ for each $j \in C$. A cover is supporting if $\sum_{i \in C \setminus \{i_1\}} a_i \leq b$, where i_1 is the first element in C. Covers are also commonly called dependent sets and edges in a conflict hypergraph.

A cover inequality is typically strengthened by viewing its extension. An extended cover $E(C)$ is created by adding elements to C with coefficients larger than every other coefficient associated with an index in the cover. Thus, $E(C) = C \cup \{i \in N \setminus C : a_i \geq max_{j \in C}\{a_j\}\}$. Clearly, an extended cover induces a valid inequality of P_{KP}^{CH} of the form $\sum_{i \in E(C)} x_i \leq |C| - 1$. The following example depicts these ideas.

Example 2.2.2. Given the knapsack constraint $3x_1 + 5x_2 + 6x_3 + 7x_4 + 4x_5 \leq 17$, it can easily be seen that $C = \{1, 2, 3, 5\}$ is a cover because $3 + 5 + 6 + 4 = 18 > 17$. The corresponding valid inequality for C is $x_1 + x_2 + x_3 + x_5 \leq 3$. This inequality is a cutting plane because $(1, 1, \frac{5}{6}, 0, 1) \in P^{LR}$ and clearly violates this inequality. Since $a_4 = 7 \geq a_3 \geq a_2 \geq a_5 \geq a_1$, index 4 is in the extended cover, $E(C) = \{1, 2, 3, 4, 5\}$, with the valid inequality $x_1 + x_2 + x_3 + x_4 + x_5 \leq 3$. This extended inequality is facet defining for P_{KP}^{CH} because the 5 points, $(0, 1, 1, 0, 1)$, $(1, 0, 1, 0, 1)$, $(1, 1, 0, 0, 1)$, $(1, 1, 1, 0, 0)$ and $(1, 1, 0, 1, 0)$, are in P_{KP} and are affinely independent.

Chapter 3

Equality Cuts and Infeasibility Conditions

This chapter provides the theoretical advancement of this research. It begins by introducing equality sets and their respective equality cuts along with several theoretical results related to equality cuts. Some results on the existence of equality cuts are shown for the MDMKP problem by finding anticovers and covers. The chapter concludes by discussing infeasibility conditions and showing examples.

3.1 Equality Cuts

The primary contribution of this thesis is the introduction of the concept of a valid equality for an integer program. A valid equality is a hyperplane in $\mathbb{R}^n$ and takes the form $\sum_{i\in N} \alpha_i x_i = \beta$. However, not all hyperplanes are valid equalities for an IP.

Formally, given an IP an equality tuple takes the form (S, α, β) where $S \subseteq N$, $\alpha \in \mathbb{R}^{|S|}$ and $\beta \in \mathbb{R}$. An equality tuple is valid if $\sum_{i\in N} \alpha_i x'_i = \beta$ for all $\mathbf{x}' \in P$. The associated equality $\sum_{i\in N} \alpha_i x_i = \beta$ is said to be a valid equality of P^{CH}.

At first glance, a valid equality may seem inherently defined by the linear relaxation, but this is not always the case. A valid equality is defined as an equality cut if there exists some $x' \in P^{LR}$ such that $\sum_{i \in N} \alpha_i x'_i \neq \beta$. Thus, any equality constraint in an IP is always a valid equality, but never an equality cut.

The idea of an equality cut may seem absurd to someone familiar with research in IP. Recall from Chapter 2, that classical IP research only defines a valid inequality as $\sum_{i \in N} \alpha_i x_i \leq \beta$. Furthermore, only valid inequalities can define a facet, and thus an equality cut can never be facet defining. Additionally, the face defined by the equality cut must contain P^{CH}. Consequently, any equality cut cannot induce a proper face and must therefore be improper. Due to these reasons, valid equalities have not been pursued by the IP research community and to the best of the author's knowledge, this work provides the first extensive endeavor into this realm of research.

There are surprisingly several theoretical results related to valid equalities. First, a valid equality exists if and only if P^{CH} is not full dimensional.

Theorem 3.1.1. *Given an IP, P^{CH} has a valid equality of the form $\sum_{i \in N} \alpha_i x_i = \beta$ if and only if $dim(P^{CH}) < n$.*

Proof: Assume P^{CH} has a valid equality of the form $\sum_{i \in N} \alpha_i x_i = \beta$. Since every $x' \in P$ satisfies $\sum_{i \in N} \alpha_i x_i = \beta$, this equality is contained in $M^=$ for P^{CH} and the $rank(M^=) \geq 1$. Because $dim(P^{CH}) + rank(M^=) = n$, $dim(P^{CH}) \leq n - 1$.

Conversely, assume that $dim(P^{CH}) < n$. Since $dim(P^{CH}) + rank(M^=) = n$, the $rank(M^=) \geq 1$. The definition of $M^=$ is a collection of hyperplanes that are satisfied by every $\mathbf{x} \in P$. Thus, there must exist at least one $\sum_{i \in N} \alpha_i x_i = \beta$ such that every $\mathbf{x} \in P$ satisfies this equality and (N, α, β) is a valid equality tuple.

$\square$

Even though valid equalities exist, that does not imply that they could ever be useful. The following theorem answers this question and demonstrates that adding an equality cut to a linear relaxation space can arbitrarily decrease the dimension of the linear relaxation space. Furthermore, such a cut must decrease P^{LR}'s dimension by at least 1.

Theorem 3.1.2. *If P^{CH} has an equality cut of the form $\sum_{i \in N} \alpha_i x_i = \beta$, then including this equality cut to the linear relaxation decreases the dimension of P^{LR} by $q \in \mathbb{Z}$ where $1 \leq q \leq dim(P^{LR})$.*

Proof: Assume that $\sum_{i \in N} \alpha_i x_i = \beta$ is an equality cut for P^{CH}. Since this is an equality cut there exists an $\mathbf{x}' \in P^{LR}$ such that $\sum_{i \in N} \alpha_i x'_i \neq \beta$. Define P'^{LR} to be $P^{LR} \cap \sum_{i \in N} \alpha_i x_i = \beta$. Since $\mathbf{x}' \notin P'^{LR}$, $\mathbf{x}'$ is trivially affinely independent from every point in P'_{LR}. Therefore, $dim(P'^{LR}) \leq dim(P^{LR}) - 1$.

To prove the second statement consider the integer program with $N = \{1, ..., q+r\}$ and the feasible points, P, defined by the following constraints.

$$\begin{aligned} 3x_1 + \textstyle\sum_{i=2}^{q} x_i &\leq 3 \\ 2x_1 + x_2 &\geq 2 \\ \textstyle\sum_{i=q+1}^{r+q} x_j &\leq 1 \\ x_i &\in \{0,1\} \text{ for } i \in \{1, ..., q+r\} \end{aligned}$$

Due to the construction of P, $x_1 = 1$ is clearly a valid equality of P^{CH}. Since the point $\mathbf{x}' = \frac{2}{3}\mathbf{e}_1 + \mathbf{e}_2$ is in P^{LR} and $x'_1 \neq 1$, $x_1 = 1$ is an equality cut.

To demonstrate the arbitrary reduction in dimension, it is first necessary to find the dimension of P^{LR}. Clearly $dim(P^{LR}) \leq q + r$ due to the number of variables. Consider the following $q+r+1$ points, $\mathbf{e}_1$, $\frac{1}{2}\mathbf{e}_1 + \mathbf{e}_2$, $\frac{1}{2}\mathbf{e}_1 + \mathbf{e}_2 + \frac{1}{2}\mathbf{e}_j$ for each $j \in \{3, ..., q\}$, $\frac{2}{3}\mathbf{e}_1 + \frac{2}{3}\mathbf{e}_2 + \sum\limits_{j=3}^{q} \frac{1}{3q}\mathbf{e}_j$, and $\mathbf{e}_1 + \mathbf{e}_k$ for each $k \in \{q+1, ..., q+r\}$. Each point is clearly in P^{LR}. Furthermore, these points are trivially affinely independent. Thus, P^{LR} is full dimensional, $dim(P^{LR}) = q + r$.

Let $\mathbf{x}'$ be any point in P'^{LR}. Clearly $x'_1 = 1$ and $x'_j = 0$ for all $j \in \{2, ..., q\}$ due to the first constraint. Thus, the maximum dimension of P'^{LR} can be at most r. The points $\mathbf{e}_1$ and $\mathbf{e}_1 + \mathbf{e}_k$ for each $k \in \{q+1, ..., q+r\}$ are in P'^{LR}, thus $dim(P'^{LR}) = r$. Consequently, including the equality cut reduced the dimension of P^{LR} by q where $1 \leq q \leq dim(P^{LR})$.

$\square$

Theorem 3.1.2 provides some potential benefit of equality cuts. If an equality cut exists, then including this cut to P^{LR} reduces its dimension by at least one. Even if the equality cut only reduces the dimension by 1, the maximum depth that could be achieved by a branch and bound algorithm is decreased by one. Consequently, if a complete branch and bound tree was required, slightly over 50% of the nodes are at the greatest depth and these nodes would all be eliminated. Thus one would expect equality cuts to be quite useful.

Since finding a valid inequality is $\mathcal{NP}$-complete[7], finding a valid equality is also NP-complete. A technique to find a valid equality is to find two opposite valid inequalities and then combine them to form a valid equality. That is, if $\sum_{i \in N} \alpha_i x_i \leq \beta$ and $\sum_{i \in N} -\alpha_i x_i \leq -\beta$ are valid inequalities of P^{CH}, then $\sum_{i \in N} \alpha_i x_i = \beta$ is a valid equality of P^{CH} as the following proposition shows.

Proposition 3.1.3. *Given an IP, if $\sum_{i \in N} \alpha_i x_i \leq \beta$ and $\sum_{i \in N} -\alpha_i x_i \leq -\beta$ are valid inequalities of P^{CH}, then $\sum_{i \in N} \alpha_i x_i = \beta$ is a valid equality of P^{CH}.*

Proof: Assume $\sum_{i \in N} \alpha_i x_i \leq \beta$ and $\sum_{i \in N} -\alpha_i x_i \leq -\beta$ are valid inequalities of P^{CH}. Multiplying the second inequality by a -1 results in $\sum_{i \in N} \alpha_i x_i \geq \beta$ being valid for P^{CH}. Thus, $\sum_{i \in N} \alpha_i x_i = \beta$ is a valid equality of P^{CH}.

$\square$

The next section describes how to use Proposition 3.1.3 and the other results from this section to obtain one class of equality cuts for MDMKP. This class of cuts uses both covers and anticovers.

3.2 Anticover Cover Equality Cuts

The previous section introduced equality cuts in the general sense. Clearly, there are an infinite number of possibilities to attempt to create equality cuts. This section describes how to implement the theory of covers to create equality cuts. To achieve these outcomes, anticovers are explained and then combined with a cover to create an equality tuple. This section concludes with an example problem that demonstrates these ideas.

3.2.1 Anticovers

As shown in Chapter 2, a demand constraint is a knapsack constraint with the appropriate substitution. This association enables a natural extension of the idea of a cover in a knapsack constraint to an anticover in the associated demand constraint. This section provides some of the basic definitions and properties of anticovers.

Given a demand constraint of the form $\sum_{i\in N} a'_i x_i \geq b'$, a set $AC \subseteq N$ is an anticover if $\sum_{i\in N\setminus AC} a'_i < b'$. Throughout this section and without loss of generality, assume that $a'_1 \geq a'_2 \geq ... \geq a'_n$ and that $AC = \{i_1, i_2, ..., i_{|AC|}\}$ is listed in this order.

Similar to covers, anticovers can be used to generate valid inequalities. If $x'_i = 0$ for all $i \in AC$, then $\mathbf{x}'$ can never satisfy the demand constraint. Thus, every anticover induces a valid inequality of the form $\sum_{j\in AC} x_j \geq 1$.

Anticover inequalities can be strengthened as follows. An anticover has parameter p if $\sum_{j=1}^{p-1} a'_{i_j} + \sum_{i\in N\setminus AC} a'_i < b'$ and $\sum_{j=1}^{p} a'_{i_j} + \sum_{i\in N\setminus AC} a'_i \geq b'$. If an anticover has parameter p, then $\sum_{j\in AC} x_j \geq p$ is a valid inequality as the following proposition shows.

Proposition 3.2.1. *Given a demand constraint and anticover $AC \subseteq N$ with parameter p, then $\sum_{j\in AC} x_j \geq p$ is a valid inequality of P_{DP}^{CH}.*

Proof: Let $AC = \{i_1, ..., i_{|AC|}\} \subseteq N$ be an anticover with parameter $p \leq |N|$. For contradiction, let $x' \in P_{DP}$ such that $\sum_{j\in AC} x'_j \leq p-1$. Since $x' \in P_{DP}$, $\sum_{i\in AC} a_i x'_i + \sum_{i\in N\setminus AC} a_i x'_i \geq b'$.

Thus, $b' \leq \sum_{i \in AC} a_i x'_i + \sum_{i \in N \setminus AC} a_i x'_i \leq \sum_{j=1}^{p-1} a_{i_j} + \sum_{i \in N \setminus AC} a_i \leq \sum_{j=1}^{p-1} a_{i_j} + \sum_{i \in N \setminus AC} a_i$. Consequently, AC has a parameter of $p-1$ or less, a contradiction. Therefore $\sum_{j \in AC} x_j \geq p$ is a valid inequality.

□

Since KPs are associated with DC's, it is not surprising that covers have an associated anticover in the associated demand constraint. The following theorem describes this equivalence.

Theorem 3.2.2. *Given a KP and its substituted DCP. Then C is a supporting cover of KP if and only if AC is an anticover in DCP with parameter p where $|AC| - |C| + 1 = p$ and $C \subseteq AC \subseteq E(C)$.*

Proof: Given a KP, $\sum_{i \in N} a_i x_i \leq b$ and its associated DCP, $\sum_{i \in N} a_i x_i \geq \sum_{i \in N} a_i - b = b'$. Assume $C \subseteq N$ is a supporting cover with extended cover $E(C)$. Let $AC \subseteq E(C)$ such that $AC \supseteq C$. Since C is a cover, $\sum_{i \in C} a_i > b$. Clearly $\sum_{i \in N \setminus AC} a_i + \sum_{i \in AC \setminus C} a_i = \sum_{i \in N} a_i - \sum_{i \in C} a_i < \sum_{i \in N} a_i - b = b'$. Thus, AC is an anticover. Due to the sorted order of AC, setting $|AC| - |C|$ elements to 0 can never be feasible and so AC's parameter must be at least $|AC| - |C| + 1$.

Examine the point $\mathbf{x}' = \sum_{i \in N \setminus C} \mathbf{e}_i + \mathbf{e}_j$ where j is the first member of C. This point must be in P_{KP} because C is supporting. Substituting $\mathbf{x}'$ into the DC constraints results in $\sum_{i \in N \setminus C} a_i + a_j$. Since C is a supporting cover, $\sum_{i \in C \setminus \{j\}} a_i \leq b$. Thus, $b' = \sum_{i \in N} a_i - b \geq \sum_{i \in N \setminus C} a_i + a_j - \sum_{i \in C \setminus \{j\}} a_i$. Since $\sum_{i \in N \setminus C} a_i + a_j$ can be rewritten as $\sum_{i \in N \setminus AC} a_i + \sum_{j=1}^{|AC|-|C|+2} a_{i_j}$, AC's parameter is strictly less than $|AC| - |C| + 2$. These two conclusions and the fact that p must be integer imply that AC's parameter is $|AC| - |C| + 1$.

Conversely, assume that $AC = \{i_1, i_2, ..., i_{|AC|}\}$ is listed in descending order and is an anticover with parameter p in the DC instance. Let C be the last $|AC| - p + 1$ indices of AC, $C = \{i_p, i_{p+1}, ..., i_{|AC|}\}$. Since AC has parameter p, $\sum_{j=1}^{p-1} a_{i_j} + \sum_{i \in N \setminus AC} a_i < b'$.

Since $b' = \sum_{i \in N} a_i - b$, one obtains $\sum_{j=1}^{p-1} a_{i_j} + \sum_{i \in N \setminus AC} a_i < \sum_{i \in N} a_i - b$. This implies $\sum_{j=p}^{|AC|} -a_{i_j} < -b$. Thus, $\sum_{j=p}^{|AC|} a_{i_j} > b$ or equivalently $\sum_{j \in C} a_j > b$ and C is a cover.

Since AC has parameter $p = |AC| - |C| + 1$, $\sum_{j=1}^{p} a_{i_j} + \sum_{i \in N \setminus AC} a_i \geq b'$. Since $b' = \sum_{i \in N} a_i - b$, one obtains $\sum_{j=1}^{p} a_{i_j} + \sum_{i \in N \setminus AC} a_i < \sum_{i \in N} a_i - b$. This implies $\sum_{j=p+1}^{|AC|} -a_{i_j} \geq -b$. Thus, $\sum_{j=p+1}^{|AC|} a_{i_j} \leq b$ or equivalently $\sum_{j \in C \setminus \{i_p\}} a_j \leq b$. Thus, C is a supporting cover of the KP.

$\square$

Since covers and anticovers are now equivalent, it is simple to extend results between the two. Common results such as facet defining properties and lifting can now easily be applied. Here we provide the result on facet defining to aid the reader in making these straightforward translations.

Theorem 3.2.3. *Given a DP with constraint $\mathbf{a'}^T\mathbf{x} \geq b'$, an anticover $AC = \{i_1, i_2, ..., i_{|AC|}\}$ with parameter p is facet defining on DP if the following three conditions are met.*

i) $\sum_{j=2}^{p+1} a'_j + \sum_{i \in N \setminus AC} a'_i \geq b'$

ii) $\sum_{j=1}^{p-1} a'_{i_j} + a'_{i_{|AC|}} + \sum_{i \in N \setminus AC} a'_i \geq b'$

iii) $\sum_{j=1}^{p} a'_{i_j} + \sum_{i \in (N \setminus AC) \setminus \{k\}} a'_i \geq b'$ *where k is the first index in $N \setminus AC$.*

Proof: Given a demand constraint $\mathbf{a'_i x_i} \geq \mathbf{b'}$, let $AC = \{i_1, i_2, ..., i_{|AC|}\}$ be an anticover with parameter p such that $\sum_{j=2}^{p+1} a'_{i_j} + \sum_{i \in N \setminus AC} a'_i \geq b'$, $\sum_{j=1}^{p-1} a'_{i_j} + a'_{i_{|AC|}} + \sum_{i \in N \setminus AC} a'_i \geq b'$ and $\sum_{j=1}^{p} a'_{i_j} + \sum_{i \in N \setminus AC \setminus \{k\}} a'_i \geq b'$ where k is the first index in $N \setminus AC$. Let $F = \{\mathbf{x} \in P_{DC}^{CH} : \sum_{i \in AC} x_i = p\}$. For AC to be facet defining $dim(F)$ must be one less than the $dim(P_{DC}^{CH})$.

As shown in Chapter 2, $dim(P_{DC}^{CH})$=$|N|$. Now consider the set of points $S' = \{\sum_{i=1}^{p+1} \mathbf{e}_i + \sum_{i \in N \setminus AC} \mathbf{e}_i - \mathbf{e}_j$ for each $j \in \{1, ..., p+1\}\} \bigcup \{\sum_{i=1}^{p-1} \mathbf{e}_i + \mathbf{e}_j + \sum_{i \in N \setminus AC} \mathbf{e}_i$ for each $j \in \{p+2, ..., |AC|\}\} \bigcup \{\sum_{i \in \{1,...,p\}} \mathbf{e}_i + \sum_{i \in N \setminus AC} \mathbf{e}_i - \mathbf{e}_j$ for each $j \in N \setminus AC\}\}$. Each of these points is in P_{DC}, due to the assumptions and the fact that DC is sorted in descending order. It can be seen that $S' \subseteq F$ and $|S'| = |N|$, making $dim(F)$$\geq |N| - 1$. Since the point

$\mathbf{x} = \mathbf{1} \in P_{DC}^{CH}$ and is not in F, the $dim(F) < dim(P_{DC}^{CH})$. Therefore $dim(F)+1 = dim(P_{DC}^{CH})$ and $\sum_{i \in AC} x_i \geq p$ is a facet defining inequality.

□

With a formal understanding of anticovers, it is now possible to use both anticovers and covers to create valid equalities. This idea is the topic of the next section.

3.2.2 Anticover Cover Equality Cuts

This section describes the existence of Anticover-Cover Equality cuts (ACE) in P_{DKP}^{CH}. In order for an ACE to be valid, there must exist a cover in the knapsack constraint coupled with an anticover in the demand constraint such that the cover anticover pair meet the conditions of Proposition 3.1.3. If such a situation exists, then a new set is created, which is called an equality set (ES). This equality set coupled with $\mathbf{1}$ and $|C| - 1$ creates an valid equality tuple (ES, $\mathbf{1}$, $|C| - 1$) with a corresponding valid equality of P_{DKP}^{CH}.

Formally, given a DKP instance, a cover C in the knapsack constraint and an anticover AC in the demand constraint. If $C = AC$ and the anticover's parameter has $p = |C| - 1$, then the cover inequality is $\sum_{i \in C} x_i \leq |C| - 1$ and the anticover inequality is $\sum_{i \in AC} x_i \geq p$, which is equivalent to $\sum_{i \in C} x_i \geq |C| - 1$. Thus $\sum_{i \in C} x_i = |C| - 1$ is a valid equality of P_{DKP}^{CH} as shown in the following theorem.

Theorem 3.2.4. *Given a DKP instance, a cover C for the knapsack constraint and an anticover AC of the demand constraint. If $C = AC$ and the anticover's parameter has $p = |C| - 1$, then C is an equality set and $\sum_{i \in C} x_i = |C| - 1$ is a valid equality for P_{DKP}^{CH}.*

Proof: Let C be a cover for the knapsack constraint and AC be an anticover of the demand constraint such that $C = AC$ and AC's parameter $p = |C| - 1$. By the definition of a cover every point in P_{DKP}^{CH} satisfies the inequality $\sum_{i \in C} x_i \leq |C| - 1$. Similarly by the definition of an anticover with parameter p, every point in P_{DKP}^{CH} satisfies the inequality $\sum_{i \in AC} x_i \geq p$.

Since $C = AC$ and $p = |C| - 1$, $\sum_{i \in AC} x_i \geq p$ can be rewritten as $\sum_{i \in C} x_i \geq |C| - 1$ by substitution. Thus, every point in P^{CH}_{DKP} satisfies the equality $\sum_{i \in C} x_i = |C| - 1$. Thus, C is an equality set with valid equality $\sum_{i \in C} x_i = |C| - 1$.

$\square$

Since a MDMKP is an extension of DKP, Theorem 3.2.4 trivially extends to P^{CH}_{MDMKP}. Additionally, an extended cover has a valid inequality of the form $\sum_{i \in E(C)} x_i \leq |C| - 1$. Thus, Theorem 3.2.5 can be strengthened by expanding from covers to extended covers as shown in the following corollary.

Corollary 3.2.5. *Given a MDMKP instance, let C be a cover of a knapsack constraint and AC be an anticover of a demand constraint. If $C \subseteq AC \subseteq E(C)$ and the anticover's parameter has $p = |C| - 1$, then AC is an equality set and $\sum_{i \in AC} x_i = |C| - 1$ is a valid equality for P^{CH}_{MDMKP}.*

$\square$

The following example demonstrates these concepts and shows how an anticover and cover can be used to create equality cuts. Additionally, it will be shown that by introducing the cut the dimension of the linear relaxation space decreases.

Example 3.2.1. Given a KDP instance with constraints K_1 and D_1

$K_1 : 3x_1 + 13x_2 + 7x_3 + 5x_4 + 11x_5 \leq 23$

$D_1 : 5x_1 + 2x_2 + 7x_3 + 11x_4 + 3x_5 \geq 19$

$x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}$.

Observe that $C = \{1, 3, 4, 5\}$ is a cover on K_1, with the corresponding inequality $x_1 + x_3 + x_4 + x_5 \leq 3$. The extended cover is $C = \{1, 2, 3, 4, 5\}$ with a valid inequality of $x_1 + x_2 + x_3 + x_4 + x_5 \leq 3$.

To establish an equality cut for this extended cover inequality, one needs an AC to be either C or $E(C)$. The set $AC = \{1, 3, 4, 5\}$ is an anticover with parameter 2 because

$a'_2 + a'_4 = 13 < 19 = b'$ and $a'_2 + a'_4 + a'_3 = 20 \geq 19 = b'$. Even though there are 3 coefficients being summed, a'_2 does not contribute to the p parameter because 2 is not in AC. Thus, the anticover inequality is $x_1 + x_3 + x_4 + x_5 \geq 2$, which when coupled with the cover inequality does not satisfy Proposition 3.1.1 and does not create a valid equality.

The set $AC = \{1, 2, 3, 4, 5\}$ is an anticover with parameter 3 because $a'_4 + a'_3 = 18 < 19$ and $a'_4 + a'_3 + a'_1 = 23 \geq 19 = b'$. Thus, its valid inequality is $x_1 + x_2 + x_3 + x_4 + x_5 \geq 3$. Since p is the same as $|C| - 1$, a valid anticover cover equality exists for P^{CH}_{DKP} of the form $x_1 + x_2 + x_3 + x_4 + x_5 = 3$.

It has been shown that $(AC, \mathbf{1}, 3)$ is a valid equality tuple. However, the associated valid equality cut must cut off linear relaxation solutions for it to be useful. Consider the linear relaxation solution $\mathbf{x}'^{LR} = (\frac{1}{3}, 0, 1, 1, 0)$. Checking $\mathbf{x}'^{LR}$ against K_1 and D_1, one gets $3(\frac{1}{3}) + 13(0) + 7(1) + 5(1) + 11(0) = 13 \leq 23$ and $5(\frac{1}{3}) + 2(0) + 7(1) + 11(1) + 3(0) = 19\frac{2}{3} \geq 19$. Thus $\mathbf{x}'^{LR}$ is feasible. Checking this point against the valid equality, $\frac{1}{3} + 0 + 1 + 1 + 0 = 2\frac{1}{3} \neq 3$, demonstrates that this is an equality cut. Since this is an equality cut, the dimension of the linear relaxation decreases by at least one due to Theorem 3.1.2.

The $\mathbf{x}'$ from above shows that the anticover cut eliminates points. Consider $\mathbf{x}''^{LR} = (1, 0, 1, 1, \frac{7}{11})$. Clearly $\mathbf{x}''$ is feasible in the linear relaxation. But, $1 + 0 + 1 + 1 + \frac{7}{11} = 3\frac{7}{11} \neq 3$. Thus, the cover cut eliminates linear relaxation space. Consequently, using the equality cut eliminates more linear relaxation space than either the cover or anticover cut individually.

To further emphasize the potential benefit of ACE equality cuts, consider solving this problem using branch and bound. Branch and bound has a maximum depth of 5 in this problem, with a total of up to $2^6 - 1 = 63$ nodes. In contrast, adding the ACE equality cut has a maximum depth of 4. Any branch after having branched on two 0s or 2 1s, must have a linear relaxation that is integer or infeasible. Thus, a maximum number of nodes evaluated is bounded by $2^5 - 1 = 31$.

To show the benefit of the equality cut, consider solving this instance with branch and bound. There are an infinite number of objective functions that could be considered. Here let the objective function be equal to minimize $\sum_{i \in N} x_i$. The branching tree requires 15 nodes and is given in Figure 3.1. In contrast, including the equality cut results in an integer solution at the root node as shown in Figure 3.2.

Since numerous objective functions could be evaluated, another objective function, minimize $5x_1 + 2x_2 + 2x_3 + 4x_4 + x_5$, was included to this set of constraints. Without the equality cut, the branching tree required 13 nodes as shown in Figure 3.3. With the equality with the branching tree is only 3 nodes as shown in Figure 3.4. This example also demonstrates that equality cuts could be quite valuable.

3.2.3 Finding ACE Cuts in Branching Trees

In practice, it is unlikely that equality cuts are easy to find at the root node of a branching tree. However, because each node in a tree represents an IP, equality cuts could exist at nodes deeper in the branching tree. The branching forces variables to fixed values, which increases the likelihood of finding ACE cuts. When found, these cuts are then applied as local cuts at that node in a branch and cut algorithm.

During a branch and cut application, the algorithm has made various branches to arrive at a particular node T_k. The linear relaxation at T_k has variables set at their upper and lower bound as required by the branching structure. Let $B^1 = \{i \in N : x_i = 1$ is a constraint in $T_k's$ linear relaxation$\}$ and $B^0 = \{i \in N : x_i = 0$ is a constraint in $T_k's$ linear relaxation$\}$.

Figure 3.1: *Branching Tree for Example 3.2.1 Without Equality Cut*

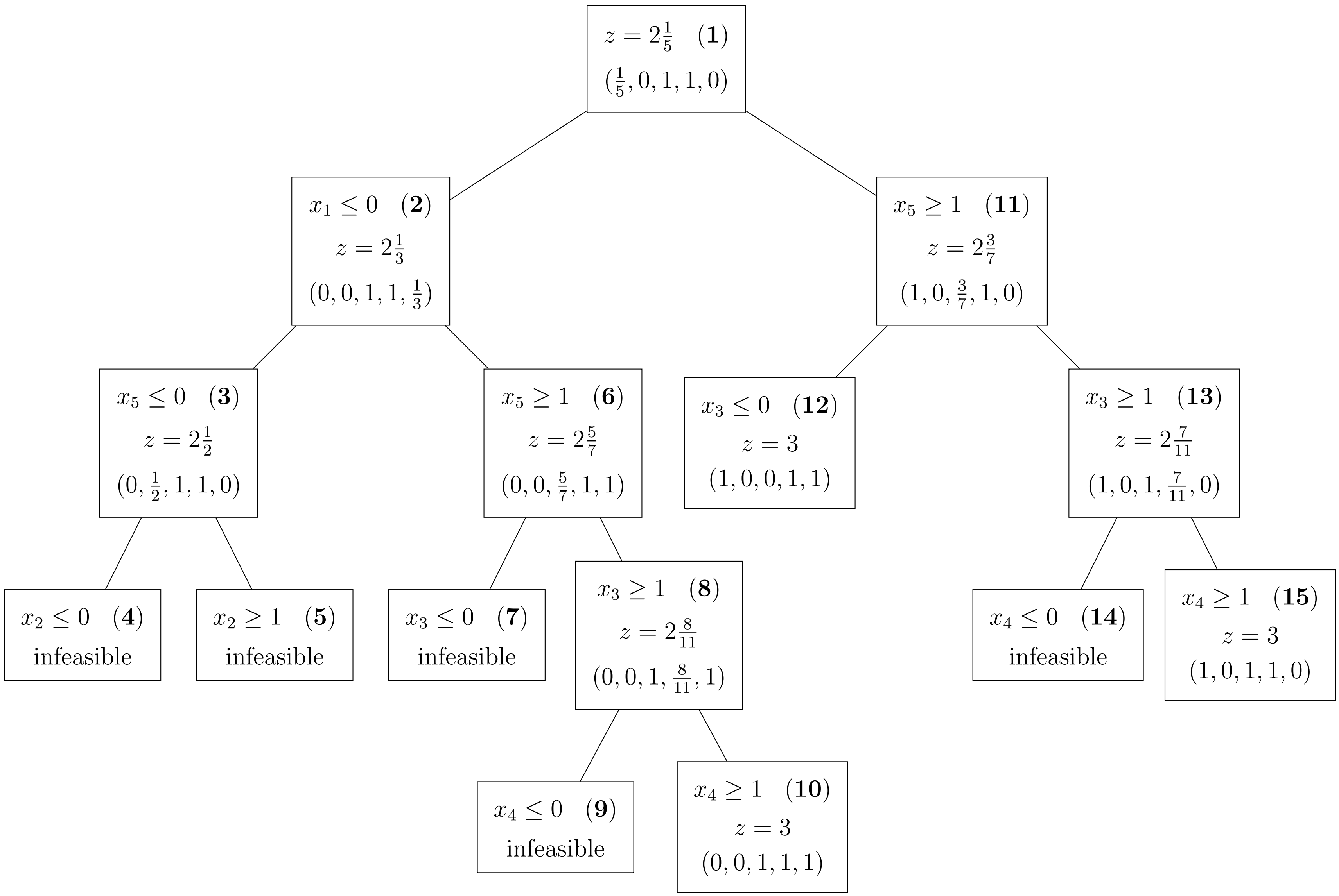


Figure 3.2: *Branching Tree for Example 3.2.1 with Equality Cut*

$$\boxed{\begin{array}{c} z = 3 \quad (\mathbf{1}) \\ (0, 0, 1, 1, 1) \end{array}}$$

The following simple algorithm, Finding ACE in Branching Trees, seeks for equality cuts at a given node. This algorithm does not necessarily find any or all cuts at a node. The algorithm requires a MDMKP instance. Let the first m constraints be the knapsack constraints and let q be the number of demand constraints.

Finding ACE in Branching Trees (FACEBT)

```
ES ← ∅
c ← |N|
ac ← 0
for i := 1 to m do
    k ← 0
    s ← Σ_{j∈B^1} a_{ij}
    Ascending Sort(A_{i.})
    for j ∉ B^0 ∪ B^1 do
        s ← s + a_{ij}
        if s ≤ b_i then
            k ← k + 1
        end if
    end for
    c ← min(k, c)
end for
for i := m + 1 to q do
    k ← 0
    s ← Σ_{j∈B^1} a_{ij}
    Descending Sort(A_{i.})
    for j ∉ B^0 ∪ B^1 do
        if s < b_i then
            s ← s + a_{ij}
            k ← k + 1
        end if
    end for
```

Figure 3.3: *Branching Tree without Equality Cut and Objective Min* $5x_1 + 2x_2 + 2x_3 + 5x_4 + x_5$

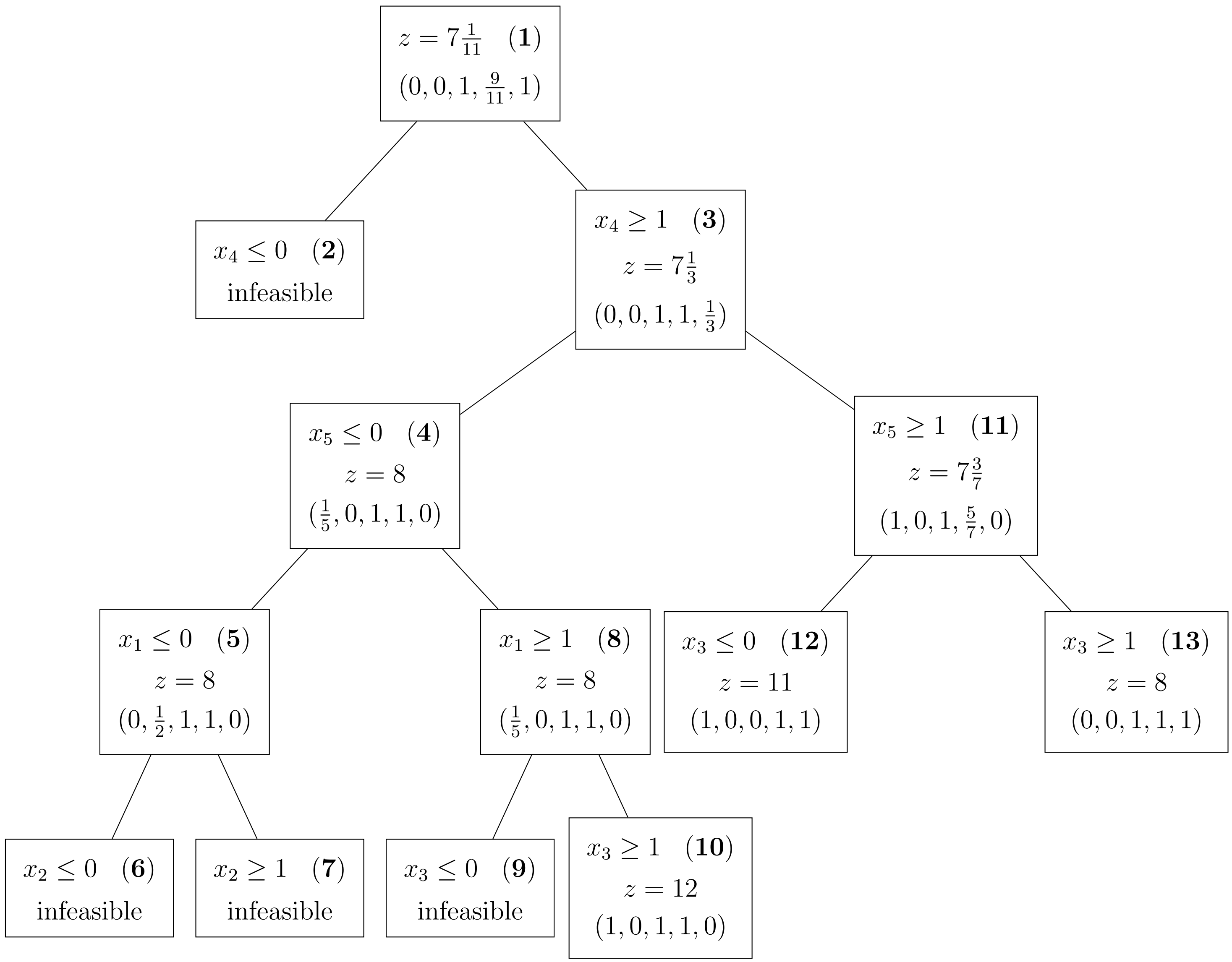
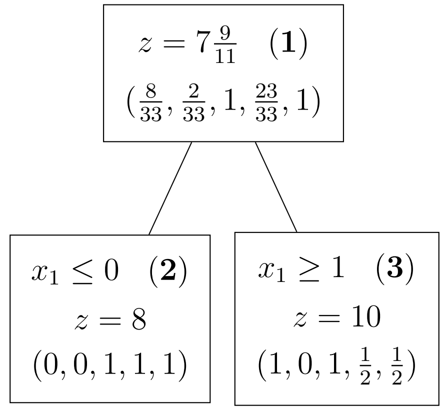


Figure 3.4: *Branching Tree with Equality Cut and Objective Min* $5x_1 + 2x_2 + 2x_3 + 5x_4 + x_5$



    $ac \leftarrow \max(k, c)$
end for
if $c = ac$ and $ac > 0$ **then**
    $ES \leftarrow N \setminus (B^0 \cup B^1)$
end if
report ES, c

The running time is on the order $O((m + q)n\log(n))$. However, $\mathbf{A}$ and $\mathbf{A}'$ only need to be sorted the first time the algorithm is run during branch and bound reducing the running time at every subsequent node to $O((m + q)n)$.

3.2.4 Infeasibility Conditions

At times, applying the algorithm for finding an equality cut during the computational study did not result in covers and anticovers that met the conditions of Theorem 3.2.5. As the branching tree progressed, a parent node failed to create an equality set, but a child had an anticover with parameter p that was at least as big as the number of elements in the cover. Such a condition implies an infeasible IP. This section discusses results related to infeasible conditions, extends the idea in general and demonstrates through an example that the associated linear relaxation space may be full dimensional even when $P = \emptyset$.

Formally, given an IP instance such that $P = \emptyset$, then the IP is clearly infeasible. Since the IP is infeasible, every inequality and equality is valid. Thus, such inequalities as $x_1 \leq -1$ and $x \geq 1$ are valid. Including these two inequalities results in an infeasible linear relaxation, which creates an infeasibility condition.

Given an IP, let $\mathbf{\Lambda x} \leq \gamma$ be a collection of valid inequalities of P^{CH} where $\mathbf{\Lambda} \in \mathbb{R}^{s \times n}$ and $\gamma \in \mathbb{R}^{s \times 1}$. If $\{\mathbf{x} \in \mathbb{R}^n : \mathbf{\Lambda x} \leq \gamma\} = \emptyset$, then $\mathbf{\Lambda x} \leq \gamma$ is called an infeasibility condition. Clearly, including the cuts of an infeasibility condition is only helpful if $P^{LR} \neq \emptyset$.

This infeasibility condition definition implies that the linear relaxation is infeasible only due to the cuts added and is not based upon P^{LR}. Certainly other definitions could be pursued, one obvious choice would be $P^{LR} \cap \{\mathbf{x} \in \mathbb{R}^n : \mathbf{\Lambda x} \leq \gamma\} = \emptyset$. The end goal of an infeasibility condition is to label a node in the branching tree as infeasible when branching would have occurred without the cuts.

During the computational study numerous anticover cover infeasibility conditions occurred. Given a MDMKP instance along with a cover C from a knapsack constraint and an anticover AC from a demand constraint such that $AC \subseteq E(C) \subseteq N$, $|AC| \geq |C|$ and AC's parameter has $p \geq |C|$, then $\sum_{i \in AC} x_i \leq |C| - 1$ and $\sum_{i \in AC} x_i \geq p \geq |C|$. Since $\sum_{i \in AC} x_i$ cannot be both less than or equal to $|C| - 1$ and greater than or equal to $|C|$, AC implies an anticover cover infeasibility condition and AC is called an anticover cover infeasible set (ACIS).

It is simple to modify FACEBT to add an additional check for ACIS. These steps are left for the reader. However, one naturally questions whether or not identifying an infeasible condition could ever be computationally useful. The following example demonstrates the potential power of ACIS.

Example 3.2.2. Given a pair of knapsack and demand constraints K_1 and D_1

$K_1 : 2x_1 + 5x_2 + 6x_3 + 7x_4 + 1x_5 \leq 17$

$D_1 : 1x_1 + 6x_2 + 7x_3 + 5x_4 + 2x_5 \geq 17.$

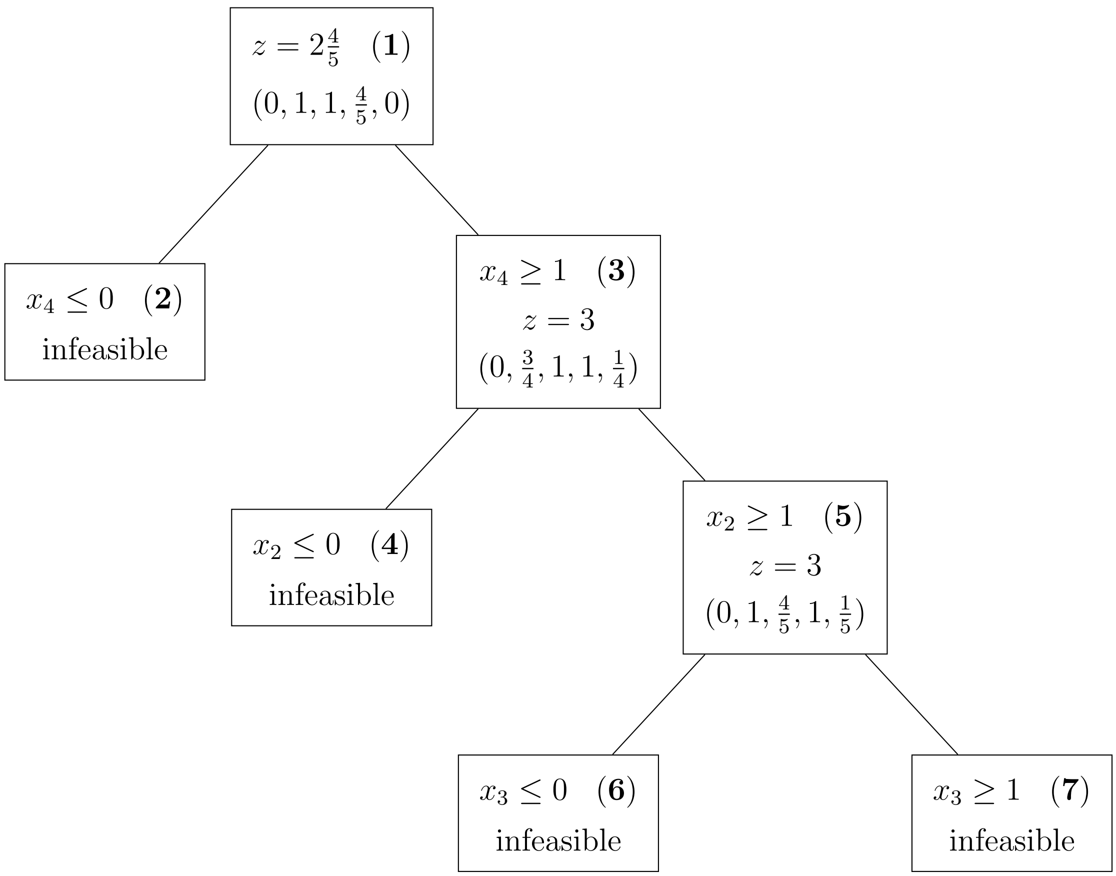
Observe that $C = \{2, 3, 4\}$ is a valid cover on K_1, with corresponding inequality $x_2 + x_3 + x_4 \leq 2$. The set $AC = \{2, 3, 4\}$ is an anticover with parameter 3 because $a'_1 + a'_5 + a'_3 + a'_2 = 16 < 17$ and $a'_1 + a'_5 + a'_3 + a'_2 + a'_4 = 21 \geq 17$. Thus, the valid anticover cut is $x_2 + x_3 + x_4 \geq 3$. Coupling the anticover cut and the cover cut creates a set of cuts that can never be satisfied by any point in $\mathbb{R}^n$. Thus, this cover and anticover form an ACIS. Adding these two cuts to P^{LR} results in an infeasible LR, $P_{KDP} = \emptyset$, and branch and bound terminates at the root node.

To demonstrate that ACIS is useful, observe that $P^{LR} \neq \emptyset$. In fact, P^{LR} is full dimensional as shown by the following points.$\mathbf{x}^1 = (0, \frac{1}{2}, 1, 1, 1)$, $\mathbf{x}^2 = (\frac{1}{8}, 1, 1, \frac{4}{5}, 0)$, $\mathbf{x}^3 = (0, 1, 1, \frac{6}{7}, 0)$, $\mathbf{x}^4 = (\frac{1}{4}, \frac{1}{2}, 1, 1, \frac{7}{8})$, $\mathbf{x}^5 = (0, 1, 1, \frac{1}{2}, 1)$, $\mathbf{x}^6 = (\frac{1}{2}, 1, \frac{1}{2}, 1, 1)$ Clearly, $\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \mathbf{x}^4, \mathbf{x}^5, \mathbf{x}^6 \in P^{LR}$. Furthermore, they are all affinely independent, so the dimension of P^{LR} is 5. Notice that by adding the cuts of the infeasibility condition, the dimension of P^{LR} is reduced to -1. Thus, these cuts fully reduce the dimension of P^{LR} and terminate branch and bound rapidly.

As further evidence of ACIS's usefulness, Figure 3.2 presents the branching tree without adding the infeasibility cuts. Again the objective function minimizes $\sum_{i \in N} x_i$. Observe that these infeasibility cuts saved 6 nodes. Alternatively, if the objective was minimize $5x_1 + 2x_2 + 2x_3 + 5x_4 + x_5$, one might expect a different number of nodes to be evaluated. In this example that was not the case and the exact same branching tree was obtained, resulting in a 6 node improvement with the addition of the infeasibility condition.

Now that both equality cuts and infeasibility conditions have been presented along with some evidence that they could be helpful, the next chapter presents a computational study. This computational study shows that this new theory can be useful.

Figure 3.5: *Branching Tree for Example 3.2.2*



Chapter 4

Computational Results

The purpose of this section is to provide computational results to support the usefulness of equality cuts and infeasibility conditions in the MDMKP. The computational results show that equality sets frequently exist and are easy to find in MDMKP instances. Additionally, performance gains were achieved in some instances using the theory presented in this thesis.

The computational study was performed on an Intel(R) Core(TM) i7-3770 3.4 GHz processor with 16.0 GB of RAM. All MDMKP instances were solved using CPLEX 12.5. An important computational note is that CPLEX was set up to store the node file on the hard drive instead of RAM due to the size of the instances. This setting changes the way CPLEX explores the tree so even though the node files for small instances may have fit completely in RAM, they were still stored on the hard drive for accurate comparisons.

4.1 Benchmark Instances

This computational study was performed on three sets of 15 MDMKP benchmark instances from the OR-Library[2]. The instances were generated using the procedure suggested by[6].

Each instance takes the form,

$$\begin{aligned} \text{Maximize} \quad & \mathbf{c}^T\mathbf{x} \\ \text{Subject to} \quad & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{A'x} \geq \mathbf{b'} \\ & \mathbf{x} \in \{0,1\}^n \end{aligned}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}_+^{m\times n}$, $\mathbf{A'} \in \mathbb{R}_+^{q\times n}$, $\mathbf{b} \in \mathbb{R}_+^m$, and $\mathbf{b'} \in \mathbb{R}_+^q$. The $\mathbf{A}$ and $\mathbf{A'}$ coefficients are generated according to a uniform distribution between 0 and 1000. The RHS b for a knapsack constraint i is $b_i = \alpha \sum_{j\in N} a_{ij}$. Similarly, b' for a demand constraint i is $b'_i = \alpha \sum_{j\in N} a'_{ij}$. The alpha values for each set were .25, .5, and .75 for instances 1-5, 6-10, and 11-15 respectively. The objective coefficient for variable i is calculated as $c_i = \sum_{j=1}^{m} a_{ji} + \lfloor 500 r_i \rfloor$, where r_i is a real number drawn from a continuous uniform distribution $U(0,1)$.

Many of the MDMKP benchmark instances could not be solved. This computational studied focused on problems with 100 variables and problems with 250 variables. These instances had either 5 demand and 5 knapsack constraints or 10 demand and 10 knapsack constraints. These instances are associated with files mdmkp_ct1.txt, mdmkp_ct2.txt , and mdmkp_ct4.txt from the OR-Library website. The OR Library website offers 6 objective functions for each type of problem and this study only solved the first objective function for the large instances, but all 6 were solved for the small instances. Thus, this study compares a total of 120 IPs.

4.2 Implementation

To compare the usefulness of equality cuts, each instance was first solved using the default CPLEX settings and then solved using three different conditions. The first added any equality cuts found, another added only infeasibility cuts and the final added any equality cut or infeasibility condition found. The results are then compared based upon the number of nodes evaluated.

The implementation in CPLEX used the default MIP solver along with a user defined cut callback routine. A cut callback routine is a user defined algorithm that is executed at each node in the branching tree to look for cuts. If a cut is found, CPLEX adds the cut forming a single child node with the IP from before the callback plus the added cut.

During the computational study, ACE cuts were not found until deep in the tree. Thus, to save computational effort one might want to set a threshold depth that must be reached before looking for ACE cuts. However, in this computational study no such threshold was used in an attempt to try and understand where ACE exist in the branching tree. This was accomplished by looking for ACE cuts and ACIS at every node and simply records the minimum depth of cut (DOC) and minimum depth of an infeasibility condition (DOI).

The cut callback routine had no measurable effect on the runtime of CPLEX other than the difference realized by exploring a different number of nodes. Therefore, it was not relevant to report the runtime differences in the computation study. However, it may be of some interest to know that the small instances $n = 100, m = 5, q = 5$ each took anywhere between 1-5 minutes to solve. The instances where $n = 100, m = 10, q = 10$ each took between 2-60 minutes and the instances where $n = 250, m = 5, q = 5$ each took between 10-120 minutes to solve.

This computation study implemented the simple algorithm presented in chapter 3. At any node the algorithm only looks for equality sets containing all of the variables not in the branching tree. If an equality set ES is present with parameter p a local cut is added of the form $\sum_{i \in ES} x_i = p$. If no equality set was found using all of the variables unbranched at the current node, then no cut was added.

One of the important aspects of ACE cuts is where they are found in the branching tree.

One way to measure their location is with branching depth. In this thesis the depth is the number of variables currently branched on, not the number of branches. This distinction is only important when branching on sets is allowed. CPLEX sometimes uses branching on sets, and it is important to note that the depth that CPLEX reports is different than the depth reported in this thesis.

Several difficulties arose when trying to leverage infeasibility conditions for computation improvement. The first challenge was telling CPLEX to fathom the node. Without drastically modifying the search, the author could not find a method of telling CPLEX to fathom a node when an infeasibility condition was discovered. In order to solve this problem a series of cuts were added to fathom the node. These cuts set every variable not in the branching tree equal to .5. The intended effect was for CPLEX to fathom the node after the first branch was evaluated. Sometimes this caused a significant increase in the number of nodes evaluated when solving an IP. The author has no good explanation for why this happened, but did find several online discussions/forums where other researchers were having similar problems.

4.3 Computational Results and Discussions

One outcome of the computational study was the existence of numerous equality cuts and infeasibility conditions. Another outcome describes the computational benefits and analysis including ACE cuts and ACISs.

In the computational study, the minimum number of equality cuts found in an instance was 85 when $n = 100, m = 5, q = 5$ and the maximum was 81,997 when $n = 250, m = 5, q = 5$. Every instances had at least two infeasibility conditions and one instance with $n = 100, m = 10, q = 10$ had over 800 infeasibility conditions. This demonstrates that both ACE and ACIS commonly exist and are easy to find in the benchmark MDMKP.

More specifically, the number of cuts, minimum DOC, infeasibility conditions and minimum DOI for the instances using the first objective row, where $n = 100, m = 5, q = 5$ are provided in Table 4.1. On average there were 4,767 equality cuts and 239 infeasibility conditions. The majority of these equality cuts were obtained deep in the branch and bound tree (on average depth 84). The variance on these instances was high and two instances found over $15,000$ equality cuts while every instance found at least 400. The results for the small instances using the other 5 objective functions are included in Appendix A.

When $n = 250, m = 5, q = 5$, the number of cuts, minimum DOC, infeasibility conditions, and minimum DOI for the instances are provided in Table 4.2. On average there were 184,832 equality cuts and 5325 infeasibility conditions. The majority of these equality cuts were obtained deep in the branch and bound tree (on average depth 222). It is quite surprising that CPLEX would obtain such a deep tree and indicates that these instances are quite challenging. The variance on these instances was high and one instances found over $500,000$ equality cuts while every problem found at least 900.

For the final instances, the number of cuts, minimum DOC, infeasibility conditions and minimum DOI, where $n = 100, m = 10, q = 10$, are provided in Table 4.3. On average there were $13,944$ equality cuts and 320 infeasibility conditions. The majority of these equality cuts were obtained deep in the branch and bound tree (on average depth 79). The variance on these instances was high and one instance found over $49,000$ equality cuts while every problem found at least 700.

In addition to demonstrating the existence of equality cuts a goal of this study was to determine if the addition of equality cuts could improve the performance of commercial IP solvers. This was accomplished by evaluating the number of nodes that had to be evaluated to solve an instance with and without cuts. The number of ticks was also reported, but the percent improvement was highly correlated with the number of nodes evaluated. Thus, the percent of effort is measured in total nodes evaluated.

The number of nodes evaluated for the instances where $n = 100, m = 5, q = 5$ is provided in Table 4.4. In total there were $42,754$ less nodes evaluated when using equality cuts and infeasibility conditions than without the cuts. This is a 17% improvement in the total number of nodes evaluated. However, using only equality cuts resulted in an increase of only 12% in the number of nodes required and using only infeasibility conditions only 16%. Even though the the average indicates a improvement, individual instances saw decreases in performance as high as 29%.

The number of nodes evaluated for the instances where $n = 250, m = 5, q = 5$ is provided in Table 4.5. On average there were $903,688$ fewer nodes evaluated when using equality cuts and infeasibility conditions than without the cuts. This is an average decrease in the total number of nodes evaluated by about 8%. Using only equality cuts resulted in an increase of 15% in the number of nodes required and using only infeasibility conditions saw an increase of 4%. Individual instances saw improvements as high as 83% and as low as -31%.

The number of nodes evaluated for the instances where $n = 100, m = 10, q = 10$ is provided in Table 4.6. In total there were $12,491,283$ fewer nodes evaluated when using equality cuts and infeasibility conditions than without the cuts. This is a decrease in the total number of nodes evaluated of about 11%. Using only equality cuts resulted in an decrease of 9% in the number of nodes required and using only infeasibility conditions saw an increase of 6%. On these instances there was an overall improvement in performance. One possible reason for the improvement is the increase in the number of constraints. This potentially allows for cuts to be found more often because there are more covers and anticovers explored by the algorithm. Individual instances saw improvements as high as 32% and performance decreases as low as 33%.

Some takeaways from this study is that equality cuts and infeasibility conditions exist frequently in MDMKP instances. Additionally, performance gains as high as a 48% reduction in the number of nodes evaluated were seen. It was surprising that the introduction of cuts significantly increased the number of nodes evaluated for some instances. This is believed to be a result of CPLEX's dynamic branching scheme and a difficulty telling CPLEX to fathom a node when an infeasibility condition was found.

Even though the use of equality cuts did not always improve the computational effort, it is important to note that every instance solved contained at least one equality cut and every instance had at least one infeasibility condition. Furthermore, ACE or ACIS was found on every instance.

Table 4.1: *Cut Table for* $n = 100, m = 5, q = 5, z = 0$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	3703	83	154	90
2	1781	79	101	87
3	1957	84	78	90
4	442	85	12	91
5	4860	83	174	87
6	21162	80	1047	86
7	2773	80	132	84
8	6930	81	229	90
9	15767	80	940	87
10	944	84	38	91
11	1764	88	62	88
12	1177	89	80	89
13	3197	89	214	89
14	983	89	68	89
15	4058	82	250	82
Average	**4767**	**84**	**239**	**88**

Table 4.2: *Cut Table for* $n = 250, m = 5, q = 5, z = 0$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	188929	222	5183	231
2	24344	227	449	236
3	911	230	20	240
4	30811	225	913	233
5	117519	219	3020	229
6	289126	220	5927	232
7	118132	221	2548	233
8	284977	221	7215	226
9	589939	224	12857	231
10	139980	220	3620	232
11	152469	216	5432	227
12	42536	221	1735	232
13	5584	224	322	227
14	357881	217	12303	228
15	429337	222	18334	230
Average	**184832**	**222**	**5325**	**231**

Table 4.3: *Cut Table for* $n = 100, m = 10, q = 10, z = 0$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	8960	79	200	86
2	3969	77	147	82
3	43669	79	802	86
4	27531	77	449	85
5	1781	82	66	89
6	20827	80	365	86
7	49557	87	838	87
8	8113	87	133	87
9	8942	81	185	89
10	2103	82	38	87
11	7888	72	234	85
12	16637	77	313	86
13	5789	70	224	86
14	708	83	708	91
15	2691	79	105	81
Average	**13944**	**79**	**320**	**86**

Table 4.4: *Nodes Evaluated for* $n = 100, m = 5, q = 5$

	No Cuts	Equality Cuts and Inf. Conditions			Equality Cuts			Inf. Conditions		
Case #	Nodes	Nodes	Difference	% Difference	Nodes	Difference	% Difference	Nodes	Difference	% Difference
1	397592	371985	25607	6%	438358	-40766	-10%	392736	4856	1%
2	90483	90872	-389	0%	83589	6894	8%	75657	14826	16%
3	177549	205277	-27728	-16%	183327	-5778	-3%	187860	-10311	-6%
4	21252	23550	-2298	-11%	27166	-5914	-28%	20910	342	2%
5	200065	226030	-25965	-13%	205702	-5637	-3%	250700	-50635	-25%
6	967749	994781	-27032	-3%	902597	65152	7%	1122556	-154807	-16%
7	130289	105310	24979	19%	100257	30032	23%	114173	16116	12%
8	669165	350794	318371	48%	393104	276061	41%	498263	170902	26%
9	715917	454429	261488	37%	486778	229139	32%	291473	424444	59%
10	52614	36410	16204	31%	44517	8097	15%	43562	9052	17%
11	25580	33121	-7541	-29%	24618	962	4%	48677	-23097	-90%
12	40300	27730	12570	31%	30664	9636	24%	36616	3684	9%
13	94675	74158	20517	22%	86910	7765	8%	93047	1628	2%
14	65465	42742	22723	35%	58980	6485	10%	51167	14298	22%
15	166305	136494	29811	18%	149464	16841	10%	141480	24825	15%
Average	**254333**	**211579**	**42754**	**17%**	**214402**	**39931**	**16%**	**224592**	**29742**	**12%**

Table 4.5: *Nodes Evaluated for* $n = 250, m = 5, q = 5$

	No Cuts	Equality Cuts and Inf. Conditions			Equality Cuts			Inf. Conditions		
Case #	Nodes	Nodes	Difference	% Difference	Nodes	Difference	% Difference	Nodes	Difference	% Difference
1	9397372	12355971	-2958599	-31	13246501	-3849129	-41%	12473807	-3076435	-33%
2	7502529	4684081	2818448	38	4448791	3053738	41%	4558362	2944167	39%
3	512060	87839	424221	83	326927	185133	36%	363337	148723	29%
4	1995665	1941813	53852	3	2426646	-430981	-22%	2057904	-62239	-3%
5	7401786	8440225	-1038439	-14	8970604	-1568818	-21%	8688341	-1286555	-17%
6	24482157	21661761	2820396	12	22196285	2285872	9%	23849188	632969	3%
7	9447348	8213374	1233974	13	8084608	1362740	14%	8842506	604842	6%
8	18892602	17354343	1538259	8	17716019	1176583	6%	18756510	136092	1%
9	28809581	28085125	724456	3	50558792	-21749211	-75%	32547242	-3737661	-13%
10	16122478	14942847	1179631	7	17587354	-1464876	-9%	16220391	-97913	-1%
11	10829656	9331837	1497819	14	8346832	2482824	23%	9823396	1006260	9%
12	2750582	1775388	975194	35	2315043	435539	16%	2412899	337683	12%
13	224778	207543	17235	8	211188	13590	6%	239772	-14994	-7%
14	8953380	7302988	1650392	18	7922754	1030626	12%	9704557	-751177	-8%
15	12772361	10153878	2618483	21	19739221	-6966860	-55%	16117589	-3345228	-26%
Average	**10672956**	**9769268**	**903688**	**8**	**12273171**	**-1600215**	**-15**	**11110387**	**-437431**	**-4**

Table 4.6: *Nodes Evaluated for* $n = 100, m = 10, q = 10$

	No Cuts	Equality Cuts and Inf. Conditions			Equality Cuts			Inf. Conditions		
Case #	Nodes	Nodes	Difference	% Difference	Nodes	Difference	% Difference	Nodes	Difference	% Difference
1	3123862	2456963	666899	21	2519047	604815	19%	2597872	525990	17%
2	1856783	1955221	-98438	-5	1980032	-123249	-7%	2481268	-624485	-34%
3	56956632	53193785	3762847	7	56345471	611161	1%	56265902	690730	1%
4	28502192	24841806	3660386	13	25204735	3297457	12%	28695030	-192838	-1%
5	1798029	1408332	389697	22	1462794	335235	19%	1563777	234252	13%
6	20841736	13481218	7360518	35	13645926	7195810	35%	13838108	7003628	34%
7	35066259	30170256	4896003	14	30005851	5060408	14%	39109598	-4043339	-12%
8	3353962	3608985	-255023	-8	4171169	-817207	-24%	3740382	-386420	-12%
9	1954036	2689318	-735282	-38	1879379	74657	4%	1956117	-2081	0%
10	1186696	1375700	-189004	-16	1522041	-335345	-28%	1305124	-118428	-10%
11	3272563	2864453	408110	12	3428544	-155981	-5%	3244364	28199	1%
12	7669316	7086331	582985	8	7009903	659413	9%	7233071	436245	6%
13	877668	941195	-63527	-7	638691	238977	27%	565645	312023	36%
14	100662	107584	-6922	-7	93816	6846	7%	92162	8500	8%
15	371044	432999	-61955	-17	389354	-18310	-5%	419379	-48335	-13%
Average	**11128763**	**9774276**	**1354486**	**12**	**10019784**	**1108979**	**10**	**10873853**	**254909**	**2%**

Chapter 5

Conclusions and Future Research

The goal of this thesis was to examine valid equality cuts. It was shown that valid equalities do indeed exist for binary IPs and although they can never be facet defining it was shown that an equality cut can reduce the dimension of search space by an arbitrary integer r, where $1 \leq r \leq |N|$.

After formally defining an anticover inequality, the idea of equality cuts was applied to the MDMKP by utilizing covers and an anticover inequality. It was demonstrated through an example that anticover cover equality cuts can significantly reduce the number of nodes needed to solve an IP.

In this thesis a formal definition of infeasibility conditions was given for an IP. Along with this definition infeasibility conditions for the MDMKP were presented. An example demonstrated the potential usefulness of infeasibility conditions by showing that an infeasibility condition can exist even if the linear relaxation space is full dimensional.

In addition to the theory, an algorithm was presented for finding equality cuts on MDMKP instances in polynomial time. Using the algorithm on benchmark problems at least one equality cut was found in every instance. Applying these equality cuts to benchmark instances improved the computational effort by about 7%.

5.1 Future Research

The validity of equality cuts for binary IPs was demonstrated in this thesis. Extending this theory to general integer programs is one possibility for future work. If possible this would significantly increase the number of applications for equality cuts.

Another theoretical extension of this work would be to develop new methods for finding the existence of equality sets. One idea to pursue would be to calculate the distance between two constraints at the linear relaxation solution. If the distance is below some threshold value that may indicate the existence of an equality set or an infeasibility condition.

A final idea for theoretical extension of this work would be to try combine the lifting and equality cuts. Lifting would allow a cut to be strengthened. Additionally, two cuts that do not form an equality set could possibly be lifted to form an equality set.

One of the biggest shortcomings of the computational study presented was that the equality cuts used contained all of the variables not in the tree. A major area of further research would be the development of algorithms for finding equality sets on smaller subsets of variables. These algorithms could also potentially identify infeasibility conditions.

Another applied extension of this work would be to perform a computational study on many different problems to try and identify which types of problems typically contain equality sets. This analysis could potentially be integrated into a commercial solver to determine when to generate equality cuts.

Bibliography

[1] Balas, E. and Zemel, E. (1978). Facets of the knapsack polytope from minimal covers. *SIAM Journal on Applied Mathematics*, 34(1):119–148.

[2] Beasley, J. The OR library. URL: http://people.brunel.ac.uk/ mastjjb/jeb/info.html.

[3] Bertsimas, D., Darnell, C., and Soucy, R. (1999). Portfolio construction through mixed-integer programming at grantham, mayo, van otterloo and company. *Interfaces*, 29(1):49–66.

[4] Caprara, A., Kellerer, H., Pferschy, U., and Pisinger, D. (2000). Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123(2):333 – 345.

[5] Ferreira, C., de Souza, C., and Wakabayashi, Y. (2002). Rearrangement of {DNA} fragments: a branch-and-cut algorithm. *Discrete Applied Mathematics*, 116(12):161 – 177.

[6] Frville, A. and Plateau, G. (1996). The 0-1 bidimensional knapsack problem: Toward an efficient high-level primitive tool. *Journal of Heuristics*, 2(2):147–167.

[7] Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R., Thatcher, J., and Bohlinger, J., editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US.

[8] Kaufman, D. E., Nonis, J., and Smith, R. L. (1998). A mixed integer linear programming

model for dynamic route guidance. *Transportation Research Part B: Methodological*, 32(6):431 – 440.

[9] Kremmel, T., Kubalk, J., and Biffl, S. (2011). Software project portfolio optimization with advanced multiobjective evolutionary algorithms. *Applied Soft Computing*, 11(1):1416 – 1426.

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

[11] Lee, E., Fox, T., and Crocker, I. (2003). Integer programming applied to intensity-modulated radiation therapy treatment planning. *Annals of Operations Research*, 119(1-4):165–181.

[12] Lee, E. and Zaider, M. (2003). Mixed integer programming approaches to treatment planning for brachytherapy application to permanent prostate implants. *Annals of Operations Research*, 119(1-4):147–163.

[13] Lu, L. L., Chiu, S. Y., and Jr, L. A. C. (1999). Optimal project selection: Stochastic knapsack with finite time horizon. *The Journal of the Operational Research Society*, 50(6):pp. 645–650.

[14] Merkle, R. and Hellman, M. (1978). Hiding information and signatures in trapdoor knapsacks. *Information Theory, IEEE Transactions on*, 24(5):525–530.

[15] Nemhauser, G. L. and Wolsey, L. A. (1999). *Integer and combinatorial optimization.* John Wiley and Sons, New York, New York, NY, USA.

[16] Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33(1):60–100.

[17] Pinto, R. and Rustem, B. (1998). Solving a mixed-integer multiobjective bond portfoliomodel involving logical conditions. *Annals of Operations Research*, 81(0):497–514.

[18] Ruiz, R., Maroto, C., and Alcaraz, J. (2004). A decision support system for a real vehicle routing problem. *European Journal of Operational Research*, 153(3):593 – 606. {EURO} Young Scientists.

[19] S. Arunapuram, K. M. and Solow, D. (2003). Vehicle routing and scheduling with full truckloads. *Transportation Science*, 37:170–182.

[20] Shih, W. (1979). A branch and bound method for the multiconstraint zero-one knapsack problem. *The Journal of the Operational Research Society*, 30(4):pp. 369–378.

[21] Toth, P. and Vigo, D. (1997). An exact algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31(4):372–385.

[22] Zemel, E. (1989). Easily computable facets of the knapsack polytope. *Mathematics of Operations Research*, 14(4):760–764.

Appendix A

Computational Results

Table A.1: *Cut Table for* $n = 100, m = 5, q = 5, z = 0$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	3703	83	154	90
2	1781	79	101	87
3	1957	84	78	90
4	442	85	12	91
5	4860	83	174	87
6	21162	80	1047	86
7	2773	80	132	84
8	6930	81	229	90
9	15767	80	940	87
10	944	84	38	91
11	1764	88	62	88
12	1177	89	80	89
13	3197	89	214	89
14	983	89	68	89
15	4058	82	250	82
Average	**4767**	**84**	**239**	**88**

Table A.2: *Cut Table for* $n = 100, m = 5, q = 5, z = 1$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	1853	82	57	91
2	2111	75	55	90
3	1297	81	40	88
4	986	81	50	92
5	9077	80	196	89
6	8258	80	265	88
7	8558	80	309	84
8	2006	82	98	90
9	1407	84	74	86
10	5750	80	186	87
11	454	80	21	91
12	4615	79	302	86
13	2068	82	65	90
14	5953	78	413	83
15	400	82	20	90
Average	**3653**	**80**	**143**	**88**

Table A.3: *Cut Table for* $n = 100, m = 5, q = 5, z = 2$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	85	81	2	92
2	191	88	3	94
3	1769	81	80	90
4	1076	81	39	89
5	3247	82	100	88
6	10452	82	218	87
7	2715	83	47	89
8	901	78	51	88
9	3511	82	89	90
10	14910	81	491	87
11	2335	83	90	90
12	2732	81	111	89
13	4564	82	147	89
14	4712	78	165	87
15	2388	84	132	89
Average	3706	82	118	89

Table A.4: *Cut Table for* $n = 100, m = 5, q = 5, z = 3$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	5048	81	165	88
2	350	86	17	94
3	12811	77	536	86
4	19928	80	729	87
5	5519	79	138	89
6	7051	80	265	88
7	18269	74	633	88
8	8748	81	237	88
9	8109	82	254	89
10	8498	81	252	89
11	3299	80	171	87
12	4165	75	196	86
13	1390	82	65	87
14	3151	75	137	86
15	2009	79	92	87
Average	**7223**	**79**	**259**	**88**

Table A.5: *Cut Table for* $n = 100, m = 5, q = 5, z = 4$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	13159	79	365	87
2	1832	82	56	86
3	6602	83	138	88
4	6001	79	157	89
5	5655	80	189	89
6	10097	81	307	88
7	1320	84	46	89
8	1460	83	35	90
9	10684	81	242	89
10	2562	82	57	86
11	248	82	16	87
12	1632	83	56	90
13	1559	84	58	88
14	966	79	30	91
15	5476	82	240	89
Average	**4617**	**82**	**133**	**88**

Table A.6: *Cut Table for* $n = 100, m = 5, q = 5, z = 5$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	3158	82	93	90
2	390	86	14	92
3	4198	83	146	89
4	2790	83	66	91
5	3309	84	94	88
6	12774	83	305	88
7	1688	81	41	89
8	18051	81	554	88
9	1128	81	37	91
10	9246	85	177	87
11	575	82	14	89
12	299	82	10	91
13	1680	82	51	88
14	1373	81	55	89
15	308	84	7	92
Average	**4064**	**83**	**111**	**89**

Table A.7: *Cut Table for* $n = 100, m = 10, q = 10, z = 0$

Case #	# Equality Cuts	Min DOC	# Infeasibility Conditions	Min DOI
1	8960	79	200	86
2	3969	77	147	82
3	43669	79	802	86
4	27531	77	449	85
5	1781	82	66	89
6	20827	80	365	86
7	49557	87	838	87
8	8113	87	133	87
9	8942	81	185	89
10	2103	82	38	87
11	7888	72	234	85
12	16637	77	313	86
13	5789	70	224	86
14	708	83	708	91
15	2691	79	105	81
Average	**13944**	**79**	**320**	**86**

Table A.8: *Nodes Evaluated for* $n = 100, m = 5, q = 5, z = 0$

	No Cuts	Equality Cuts and Inf. Conditions			Equality Cuts			Inf. Conditions		
Case #	Nodes	Nodes	Difference	% Difference	Nodes	Difference	% Difference	Nodes	Difference	% Difference
1	397592	371985	25607	6%	438358	-40766	-10%	392736	4856	1%
2	90483	90872	-389	0%	83589	6894	8%	75657	14826	16%
3	177549	205277	-27728	-16%	183327	-5778	-3%	187860	-10311	-6%
4	21252	23550	-2298	-11%	27166	-5914	-28%	20910	342	2%
5	200065	226030	-25965	-13%	205702	-5637	-3%	250700	-50635	-25%
6	967749	994781	-27032	-3%	902597	65152	7%	1122556	-154807	-16%
7	130289	105310	24979	19%	100257	30032	23%	114173	16116	12%
8	669165	350794	318371	48%	393104	276061	41%	498263	170902	26%
9	715917	454429	261488	37%	486778	229139	32%	291473	424444	59%
10	52614	36410	16204	31%	44517	8097	15%	43562	9052	17%
11	25580	33121	-7541	-29%	24618	962	4%	48677	-23097	-90%
12	40300	27730	12570	31%	30664	9636	24%	36616	3684	9%
13	94675	74158	20517	22%	86910	7765	8%	93047	1628	2%
14	65465	42742	22723	35%	58980	6485	10%	51167	14298	22%
15	166305	136494	29811	18%	149464	16841	10%	141480	24825	15%
Average	254333	211579	42754	17%	214402	39931	16%	224592	29742	12%

Table A.9: *Nodes Evaluated for* $n = 100, m = 5, q = 5, z = 1$

	No Cuts	Equality Cuts and Inf. Conditions			Equality Cuts			Inf. Conditions		
Case #	Nodes	Nodes	Difference	% Difference	Nodes	Difference	% Difference	Nodes	Difference	% Difference
1	82135	105091	-22956	-28%	79713	2422	3%	59474	22661	28%
2	109179	172028	-62849	-58%	108818	361	0%	130289	-21110	-19%
3	102220	92834	9386	9%	81535	20685	20%	100911	1309	1%
4	67826	69001	-1175	-2%	66752	1074	2%	64016	3810	6%
5	353563	328433	25130	7%	341008	12555	4%	356781	-3218	-1%
6	481360	559791	-78431	-16%	580285	-98925	-21%	306404	174956	36%
7	325213	386266	-61053	-19%	367620	-42407	-13%	453625	-128412	-39%
8	81282	65271	16011	20%	64846	16436	20%	77721	3561	4%
9	154140	95644	58496	38%	137844	16296	11%	106504	47636	31%
10	228925	184226	44699	20%	186749	42176	18%	213811	15114	7%
11	15343	15163	180	1%	16163	-820	-5%	15235	108	1%
12	154674	139334	15340	10%	139652	15022	10%	146717	7957	5%
13	51781	50889	892	2%	51098	683	1%	84569	-32788	-63%
14	181459	154736	26723	15%	163833	17626	10%	177141	4318	2%
15	14884	13468	1416	10%	14104	780	5%	13965	919	6%
Average	160266	162145	-1879	-1%	160001	264	0%	153811	6455	4%

Table A.10: *Nodes Evaluated for* $n = 100, m = 5, q = 5, z = 2$

	No Cuts	Equality Cuts and Inf. Conditions			Equality Cuts			Inf. Conditions		
Case #	Nodes	Nodes	Difference	% Difference	Nodes	Difference	% Difference	Nodes	Difference	% Difference
1	10402	10242	160	2%	10236	166	2%	10396	6	0%
2	10327	9880	447	4%	9780	547	5%	10315	12	0%
3	112918	100423	12495	11%	100363	12555	11%	111250	1668	1%
4	66266	75173	-8907	-13%	90726	-24460	-37%	83715	-17449	-26%
5	259566	210545	49021	19%	224979	34587	13%	267056	-7490	-3%
6	1236134	1472375	-236241	-19%	1156930	79204	6%	1291630	-55496	-4%
7	267712	312012	-44300	-17%	368572	-100860	-38%	324940	-57228	-21%
8	54980	60008	-5028	-9%	59919	-4939	-9%	66759	-11779	-21%
9	374450	296702	77748	21%	401123	-26673	-7%	334326	40124	11%
10	1093570	1007018	86552	8%	981191	112379	10%	1009144	84426	8%
11	118776	107621	11155	9%	106415	12361	10%	95149	23627	20%
12	155731	130193	25538	16%	132514	23217	15%	158186	-2455	-2%
13	206887	185084	21803	11%	176920	29967	14%	196360	10527	5%
14	271865	274324	-2459	-1%	414175	-142310	-52%	385533	-113668	-42%
15	100143	98775	1368	1%	99865	278	0%	102494	-2351	-2%
Average	289315	290025	-710	0%	288914	401	0%	296484	-7168	-2%

Table A.11: *Nodes Evaluated for* $n = 100, m = 5, q = 5, z = 3$

	No Cuts	Equality Cuts and Inf. Conditions			Equality Cuts			Inf. Conditions		
Case #	Nodes	Nodes	Difference	% Difference	Nodes	Difference	% Difference	Nodes	Difference	% Difference
1	317699	294090	23609	7%	289734	27965	9%	311479	6220	2%
2	40251	32274	7977	20%	46090	-5839	-15%	40192	59	0%
3	834724	1083431	-248707	-30%	566156	268568	32%	577893	256831	31%
4	1118162	1294518	-176356	-16%	1232171	-114009	-10%	1201086	-82924	-7%
5	466812	313985	152827	33%	368142	98670	21%	388247	78565	17%
6	437116	342162	94954	22%	373193	63923	15%	350525	86591	20%
7	792164	662535	129629	16%	719326	72838	9%	960194	-168030	-21%
8	1206642	716761	489881	41%	717594	489048	41%	1367575	-160933	-13%
9	1238549	783375	455174	37%	1040576	197973	16%	952224	286325	23%
10	733240	637748	95492	13%	713904	19336	3%	750732	-17492	-2%
11	128869	120796	8073	6%	165082	-36213	-28%	116039	12830	10%
12	88449	89027	-578	-1%	70188	18261	21%	89988	-1539	-2%
13	92020	87572	4448	5%	88610	3410	4%	98079	-6059	-7%
14	126058	124367	1691	1%	136368	-10310	-8%	122017	4041	3%
15	98187	98791	-604	-1%	77106	21081	21%	91005	7182	7%
Average	514596	445429	69167	13%	440283	74313	14%	494485	20111	4%

Table A.12: *Nodes Evaluated for* $n = 100, m = 5, q = 5, z = 4$

	No Cuts	Equality Cuts and Inf. Conditions			Equality Cuts			Inf. Conditions		
Case #	Nodes	Nodes	Difference	% Difference	Nodes	Difference	% Difference	Nodes	Difference	% Difference
1	1143586	1143737	-151	0%	1056847	86739	8%	1451031	-307445	-27%
2	123298	108968	14330	12%	115533	7765	6%	113504	9794	8%
3	852503	846114	6389	1%	836842	15661	2%	849971	2532	0%
4	345512	468787	-123275	-36%	223626	121886	35%	127845	217667	63%
5	579958	563642	16316	3%	557653	22305	4%	54375	525583	91%
6	689863	711813	-21950	-3%	695355	-5492	-1%	692910	-3047	0%
7	145889	119315	26574	18%	113279	32610	22%	144202	1687	1%
8	104462	126534	-22072	-21%	126357	-21895	-21%	115238	-10776	-10%
9	1053397	1008507	44890	4%	1019196	34201	3%	2495715	-1442318	-137%
10	170907	166578	4329	3%	184160	-13253	-8%	185205	-14298	-8%
11	12323	11319	1004	8%	11877	446	4%	10871	1452	12%
12	71515	69483	2032	3%	69205	2310	3%	70286	1229	2%
13	70810	84387	-13577	-19%	93189	-22379	-32%	84096	-13286	-19%
14	68124	69260	-1136	-2%	78411	-10287	-15%	69493	-1369	-2%
15	314049	332279	-18230	-6%	297885	16164	5%	367965	-53916	-17%
Average	383080	388715	-5635	-1%	365294	17785	5%	455514	-72434	-19%

Table A.13: *Nodes Evaluated for* $n = 100, m = 5, q = 5, z = 5$

	No Cuts	Equality Cuts and Inf. Conditions			Equality Cuts			Inf. Conditions		
Case #	Nodes	Nodes	Difference	% Difference	Nodes	Difference	% Difference	Nodes	Difference	% Difference
1	396475	454266	-57791	-15%	360930	35545	9%	407540	-11065	-3%
2	34617	31446	3171	9%	30158	4459	13%	36359	-1742	-5%
3	591901	693549	-101648	-17%	709534	-117633	-20%	937797	-345896	-58%
4	1064026	712317	351709	33%	871299	192727	18%	796003	268023	25%
5	435201	553297	-118096	-27%	364896	70305	16%	606187	-170986	-39%
6	2788510	2364403	424107	15%	2404361	384149	14%	2532036	256474	9%
7	316143	282488	33655	11%	300018	16125	5%	310263	5880	2%
8	2292653	1942018	350635	15%	2151924	140729	6%	2719097	-426444	-19%
9	124126	129661	-5535	-4%	132004	-7878	-6%	177314	-53188	-43%
10	2098384	1989203	109181	5%	1830659	267725	13%	2178169	-79785	-4%
11	46647	46502	145	0%	85605	-38958	-84%	60076	-13429	-29%
12	32789	30108	2681	8%	29654	3135	10%	32000	789	2%
13	106467	106573	-106	0%	108381	-1914	-2%	106785	-318	0%
14	123545	145271	-21726	-18%	128037	-4492	-4%	127607	-4062	-3%
15	45198	37299	7899	17%	35991	9207	20%	45289	-91	0%
Average	699779	634560	65219	9%	636230	63549	9%	738168	-38389	-5%