THREE SET INEQUALITIES IN INTEGER PROGRAMMING

by

MICHAEL JOHN MCADOO

B.S., Kansas State University, 2005

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

2007

Approved by:

Major Professor

Dr. Todd Easton

Abstract

Integer programming is a useful tool for modeling and optimizing real world problems. Unfortunately, the time required to solve integer programs is exponential, so real world problems often cannot be solved. The knapsack problem is a form of integer programming that has only one constraint and can be used to strengthen cutting planes for general integer programs. These facts make finding new classes of facet-defining inequalities for the knapsack problem an extremely important area of research.

This thesis introduces three set inequalities (TSI) and an algorithm for finding them. Theoretical results show that these inequalities will be of dimension at least 2, and can be facet defining for the knapsack problem under certain conditions. Another interesting aspect of these inequalities is that TSIs are some of the first facet-defining inequalities for knapsack problems that are not based on covers. Furthermore, the algorithm can be extended to generate multiple inequalities by implementing an enumerative branching tree.

A small computational study is provided to demonstrate the effectiveness of three set inequalities. The study compares running times of solving integer programs with and without three set inequalities, and is inconclusive.

Table of Contents

List of Figures

List of Tables

Chapter 1 - Introduction

Integer programming (IP) is an important class of mathematical programming problems used to optimize linear systems that require the variables to be integer. This thesis presents a new class of inequalities, called three set inequalities (TSIs) that can be used as a tool to help solve IPs. Both theoretical and computational results relating to TSIs are presented.

In recent years, integer programs have been applied to numerous applications. IPs are beneficial because, if one can solve them, then one is guaranteed to obtain the best solution. However, this guarantee of optimality has a computational tradeoff, and integer programs currently may require exponential time to solve. The computational problems are so extreme that many IPs cannot be solved, even using supercomputers.

One example of the usefulness of IPs optimized the scheduling and deployment of San Francisco Police Department patrol officers [18, 28]. The criteria used in this study were the level of public safety, level of officer morale, and cost of operations. The computerized system that was developed used a mathematical model to incorporate each of these goals and increased SFPD's net income by 14 million dollars and decreased response times by 20 percent.

In addition to the above application, integer programs have been used to solve a number of real-world problems, including airline scheduling [16, 18, 20], sports scheduling [10, 18, 20], construction site selection [18, 20, 23], manufacturing job scheduling [18, 23, 29], and telephone network optimizations [16, 20, 23]. The number of uses these programs have is the reason that finding better ways to solve them is so important.

The specific form of integer programs that this thesis focuses on is the knapsack problem (KP). The knapsack problem seeks to optimize a set of yes/no decisions subject to a single non-negative constraint. A classic example of this problem is a camper going backpacking. He wishes to bring the best combination of equipment he can. Each piece of equipment (tent, food, water, fire supplies, etc.) has a value to the camper that is assigned a numerical representation. Each piece of equipment also has a corresponding weight. The camper can only bring as much equipment as he can carry.

Knapsack problems are widely used in financial decision making. Two examples of these applications are resource allocation [13, 23] and portfolio management [6, 24, 27]. In resource allocation, a company wishes to maximize its return from resources invested into each division or product subject to the total resources available. In portfolio management, the goal is to maximize return while minimizing risk.

The knapsack problem is widely studied because of its importance to integer programs. Any single constraint of a binary integer program can be viewed as a knapsack constraint. Therefore, advancements to KPs can frequently be applied to any general integer program.

1.1 Complexities of Integer Programs

Formally, an integer programs is defined as

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{Subject to:} \quad & Ax \leq b \\ & x \geq 0 \text{ and integer} \end{aligned}$$

where $A \in R^n$, $x \in R^n$, $b \in R^{1 \times m}$.

The only difference between this form and the common form of linear programs is the integer restriction. The problem with IPs is the time and resources needed to solve such a problem. Integer programs are $\mathcal{NP}$-complete [21], meaning that all known algorithms require exponential time. Although many small IPs can be solved quickly, more complex IPs can take extraordinary amounts of time to solve and frequently use the entire memory of a computer without obtaining an optimal or even a feasible solution.

Because solving IPs is difficult but beneficial, considerable effort has been made to develop methods that can decrease solution times for IPs. The two most common algorithms use a linear relaxation. Linear relaxation is the solution to the IP without the integer constraint. Linear programming can solve much faster than an IP, so the IP is reformulated as a linear program. The optimal value in the linear program (called the linear relaxation point) is found using any of the methods available to solve LPs. Once the linear relaxation point is found, either branch and bound or cutting planes can be used to find the solution to the IP.

Branch and bound uses the linear relaxation as a starting point to search for the optimal integer solution. Every linear relaxation solution that is found during the branch and bound process is given a corresponding node on the branching tree. Once a node's relaxation point has been found, any variable with a fractional value may be chosen as the branching variable. Two child nodes with corresponding branches are created from this parent node. One branch requires the branching variable to be greater than or equal to its relaxation value rounded up to the nearest integer. The other branch requires the branching variable to be less than or equal to the relaxation solution rounded down to the nearest integer. Using these values, two new relaxation points are found and two more nodes are created in the tree. The process is repeated until all nodes have been fathomed.

A fathomed node is finished, and no more nodes or branches are created below any fathomed nodes. Fathoming a node in a branch and bound algorithm occurs under three circumstances. If a node is found that can not produce a feasible solution to the linear relaxation, then that node is fathomed. If a node is found that returns an integer solution, then the node is fathomed. Although other feasible solutions may exist below that node, none will be better than that node's solution. This property is applied to produce the third fathoming condition. If a node has a linear relaxation solution with a value lower than the value of a previously discovered integer solution, then that node is fathomed.

An alternative to the branch and bound method is to use cutting planes to reduce the linear relaxation space. This method attempts to find a hyperplane that intersects the solution space below the current linear relaxation point without eliminating any integer solutions. Once such a hyperplane has been put in place, a new linear relaxation point is found and branch and bound can be implemented or additional cutting planes can be added until an integer solution is returned as the solution to the LP.

Some cutting planes are more useful than others. Facet-defining cutting planes are the most useful cutting planes. If all facet-defining cutting planes are included as constraints for an IP, then the solution to the linear relaxation is guaranteed to also be the solution to the IP.

1.2 Motivation

The goal of this research was to develop a method for finding a set of facet-defining cutting planes not based on cover inequalities. These inequalities could then be used to help solve integer programs. This research also sought to use a method that could be developed into an interior method for solving integer programming, an approach that has not been explored much at this point.

1.3 Contribution

This research developed a new class of valid inequalities for the knapsack problem called three set inequalities. These inequalities can be facet defining under certain conditions, and are among the first non-cover facet-defining inequalities for the knapsack problem. A method for finding these inequalities, called the three set inequality algorithm, is also provided along with relevant examples. This method is extended to a branching tree that can find numerous such inequalities. A small computational study is performed and reveals some benefits and issues associated with this algorithm.

1.4 Outline

Chapter 2 provides necessary background information on integer programming, polyhedral theory and facet-defining inequalities. This chapter also explains lifting and cover inequalities and the theory related to these areas of research. Some of the reasons for the exponential time required to run an IP and current theories on reducing that time will also be explored. Finally, cutting planes and the knapsack problem will be discussed.

In Chapter 3 three set inequalities will be introduced and explained. Included in this chapter will be a formal definition of these inequalities and an algorithm that finds these inequalities. Some extensions and issues of this algorithm along with the theoretical proofs are also presented.

Chapter 4 will provide a computational study of three set inequalities applied to knapsack instances. This chapter will compare running times and number of nodes evaluated using three set inequalities against standard integer programming techniques.

Chapter 5 will conclude this thesis with additional comments on three set inequalities. Some exciting areas of future research are also discussed along with possible ideas to pursue these important problems.

Chapter 2 - Background Information

In order to understand the value of three set inequalities, it is necessary to have a good understanding of some of the background polyhedral theory for both general IPs and specifically for knapsack polyhedra.

As stated above, an integer program can be formally stated as:

$$\begin{aligned} &\text{Maximize} && c^T x \\ &\text{Subject to:} && Ax \leq b \\ & && x \geq 0 \text{ and integer} \end{aligned}$$

where $A \in R^{n \times m}$, $x \in Z^n$, $b \in R^{1 \times m}$.

Every IP has a corresponding linear relaxation. The linear relaxation is used as part of various methods to solve IPs, and is defined as:

$$\begin{aligned} &\text{Maximize} && c^T x \\ &\text{Subject to:} && Ax \leq b \\ & && x \geq 0 \end{aligned}$$

where $A \in R^n$, $x \in R^n$, $b \in R^{1 \times m}$.

Solving IPs can be difficult and time consuming. A great deal of research has been performed to improve the solving times and ease of IPs. One of the major areas this research has been done in is polyhedral theory.

2.1 Polyhedral Theory

Polyhedral theory is an important body of knowledge that helps describe and develop solutions to both linear and integer programs. The feasible region of any linear program can be represented as a polyhedron, and it is this polyhedron that polyhedral theory seeks to describe.

First, a set, $T \subseteq \mathbf{R}^n$, is convex if and only if $\lambda x^1 + (1-\lambda)x^2 \in T$ for every $\lambda \in [0,1]$ and every point x^1 and $x^2 \in T$. Another way of stating this is that the line between any two points contained in T is entirely contained in T. Similarly, if $S \subseteq \mathbf{R}^n$, then the convex hull, $conv(S)$, is defined as the intersection of all convex sets that contain S.

A halfspace is the set of points that satisfy a linear inequality, $\{x \in \mathbf{R}^n : a^T x \leq b\}$. A polyhedron is a finite intersection of halfspaces. Clearly, a polyhedron is a convex set. Furthermore, the solution space of a linear program is also a polyhedron. A polytope is defined as a bounded polyhedron.

Given an integer program, Max $c^T x$, subject to $Ax \leq b$, $x \geq 0$ and integer, let P be the set of feasible solutions. Thus, $P = \{x \in \mathbf{Z}^n : Ax \leq b, x \geq 0\}$. The goal of polyhedral theory in integer programming is to completely describe $conv(P)$, which is now referred to as P^{ch}. The fact that P^{ch} is a polyhedron is vital to IP research.

Two types of polyhedron points are critical to this research. Let $x' \in P^{ch}$, then x'

is an extreme points if and only if there does not exist $x_1, x_2 \in P^{ch}, x_1 \neq x_2$, such that $x' = 0.5x_1 + 0.5x_2$. Any point that is not an extreme point is called an inner point. See [23] for a more complete discussion on these topics. Observe that the extreme points of P^{ch} are always integer and that an optimal solution to any LP will always occur at an extreme point.

To describe P^{ch}, a knowledge of the dimension of a set of points is also critical. It is important to determine the dimension so that we can know what spaces are critical in P^{ch}. Affine independence can be used to determine the dimension of a space.

The points $x^1, ..., x^q \in \mathbf{R}^n$ are affinely independent if and only if the unique solution to $\sum_{i=1}^{q} \lambda_i x^i = 0$ and $\sum_{i=1}^{q} \lambda_i = 0$ is $\lambda_i = 0$ for $i = 1, ..., q$. The dimension of a space is equal to the maximum number of affinely independent points minus one. The problem with determining a polyhedron's dimension by the maximum number of affinely independent pointsis that it may be challenging to know whether or not the maximum is obtained.

Two theorems are very important to this area of polyhedral theory. The first relates the dimension of a space to its rank, which is defined as the number of linearly independent vectors that can be found. The second theorem relates the dimension of a face to a space and is given further on in the thesis.

Theorem 2.1 *If $P^{ch} \subseteq R^n$, then dim(P^{ch})+rank($A'^{=}, b'^{=}$) $= n$ where $A'^{=}$ and $b'^{=}$ are the constraints that are met at equality by every $x \in P^{ch}$.*

Proof: [23]. If a polyhedron has dimension equal to n, then it is considered to be full-dimensional. In the case that the polyhedron is the empty set ($\emptyset$), the dimension is defined to be -1.

An inequality $\alpha^T x \leq \beta$ is a valid inequality for P^{ch} if and only if the inequality is satisfied by every point in P. That is, the inequality cannot eliminate a feasible point. A valid inequality is also called a cut or cutting plane.

Each cut or cutting plane, $\alpha^T x \leq \beta$, induces a face of P^{ch} and the face takes the form $\{x \in P^{ch} : \alpha^T x = \beta\}$. A face is proper if it is not P^{ch} or $\emptyset$. A valid inequality that does not define a proper face of P^{ch} is redundant and can be removed.

A facet-defining inequality is an inequality that defines a face of dimension one less than the dimension of P^{ch}. Facets are important because of the role they can play in solving IPs. If a facet is found, the portion of space in the linear relaxation above the facet will be completely cut off, and any other inequalities used to describe that face will be dominated by the facet. If all facets are found, an optimal solution to the linear relaxation is guaranteed to be an integer point and thus it will also be the optimal solution to the IP.

The following theorem is frequently used to prove that an inequality defines a facet.

Theorem 2.2 *Let $\alpha^T x \leq \beta$ be a valid in equality of P^{ch}. Then if there exists an $x' \in P^{ch}$ such that $\alpha^T x' < \beta$, then the dimenstion of the face induced by $\alpha^t x \leq \beta$ is at most dim(P^{ch})-1.*

Proof: [17].

Given the theory explored so far, it is useful to look at a two dimensional example. The following example shows the basics of polyhedral theory.

Example 2.1

$$\begin{aligned} \text{Maximize} \quad & 4x_1 + 3x_2 \\ \text{Subject to:} \quad & x_1 + 2x_2 \leq 7 \\ & 3x_1 + x_2 \leq 10 \\ & x_1, x_2 \geq 0 \\ & x_i \in Z^1 \end{aligned}$$

This problem is modeled graphically in Figure 2.1. An examination of the graph makes it obvious that the problem is indeed 2-dimensional (as there are only two variables). The integer points within the polyhedron are shown by the solid circles. The optimal solution to the linear relaxation is shown by the empty circles. The lines represent the constraints.

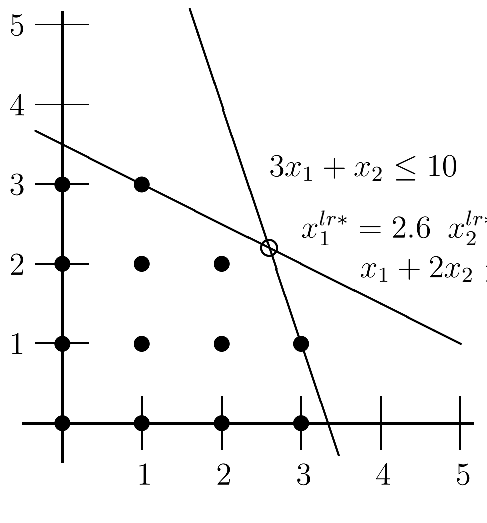

Figure 2.1: *Graph of the IP and Linear Relaxation from Example 2.1*

The convex hull of this polyhedron is the space defined by all of the facets. Figure 2 shows a graphical representation of P^{ch}. As in the IP graph, it is clear that P^{ch} is 2-dimensional and so is also full-dimensional.

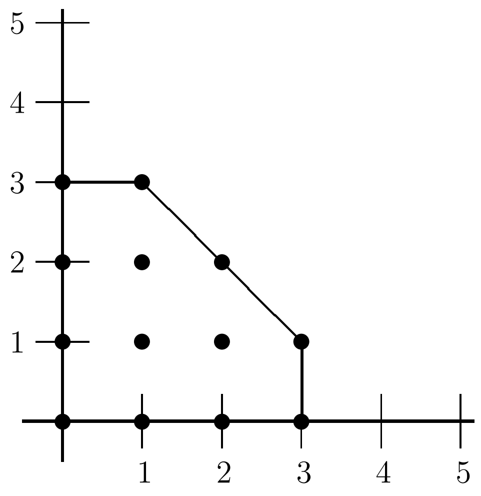

Figure 2.2: *Graph of the Convex Hull from Example 2.1*

A valid inequality for this polyhedron exists at $x_2 \leq 4$. However, as Figure 3 clearly shows, this does not cut off any space from the linear relaxation, and so is not a cutting plane.

Figure 4 shows the inequality $x_2 - x_1 \leq 3$, which does cut off a portion of the linear relaxation without cutting off any integer points. In fact, this cut intersects a feasible integer point, so it is a face of this polyhedron. However, because it only intersects one feasible integer point, its dimension is too low for it to be a facet.

A facet of this polyhedron can be found by using the valid inequality $x_2 \leq 3$. This face intersects two affinely independent integer points, which is the most possible for a cutting plane in a two dimensional problem. This fact can be seen in Figure 5. A formal proof that

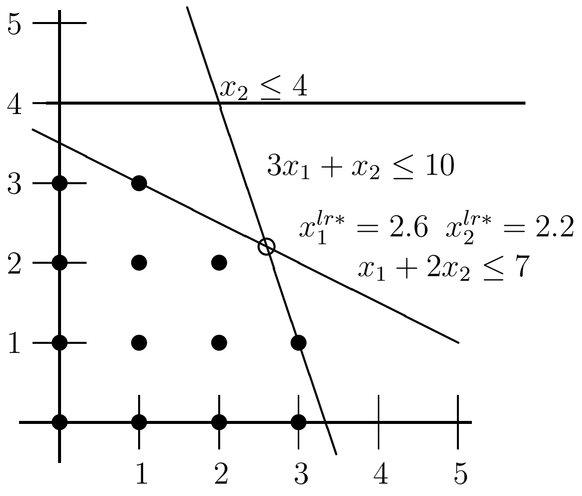

Figure 2.3: *Graph of a Valid Inequality for the IP from Example 2.1*

this cut is facet-defining is shown below:

1. As stated above, $\dim(P^{ch}) = 2$.

2. As seen in Figure 5, clearly no integer points are cut off, so the cut $x_2 \leq 3$ is a valid inequality

3. There are two points, $(3, 0)$ and $(0, 3)$ that meet this constraint at equality, and they are clearly affinely independent. So, $\dim(F) \geq \dim(P^{ch}) - 1 = 1$. Because there is a feasible point that does not meet the constraint at equality at $(0, 0)$, $\dim(F) \leq \dim(P^{ch}) - 1 = 1$. So, $\dim(F) = 1$, and $x_2 \leq 3$ is a facet-defining inequality.

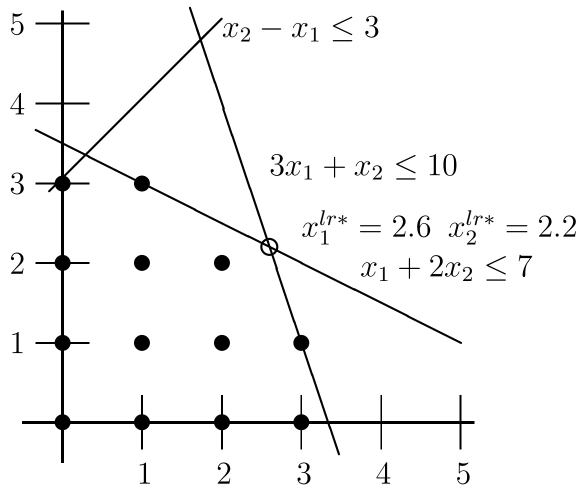

Figure 2.4: *Graph of a Face of the IP from Example 2.1*

2.1.1 Knapsack Polyhedra

The knapsack problem is a specific type of integer program to which the above polyhedral theory can be applied. The knapsack formulation is the same as a basic IP with only one constraint and binary variables. Without loss of generality, the knapsack problem can be assumed to be sorted in the form $a_1 \geq a_2 \geq ... \geq a_i$ and $\sum a_i \geq b$.

Formally, a knapsack problem is defined as

$$\begin{array}{ll} \text{Maximize} & c^T x \\ \text{Subject to:} & a^T x \leq b \\ & x \in \{0,1\}^N \end{array}$$

Because the knapsack problem is formulated for sets of solutions containing only ones and zeroes, it is ideally suited to model decision systems. One of these types of problems is the capital budgeting problem, in which a decision maker wishes to maximize profit from

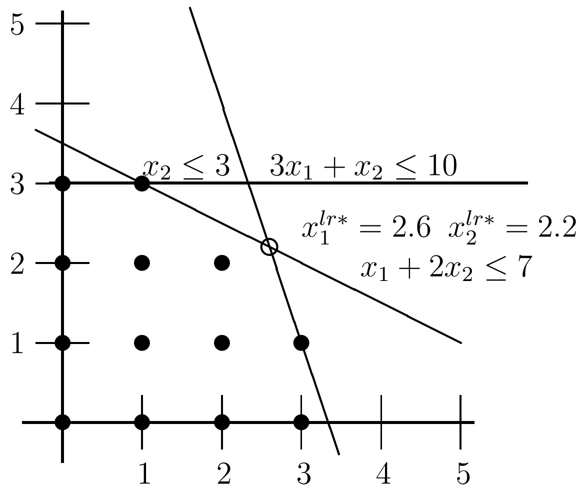

Figure 2.5: *Graph of a Facet-defining Inequality of the IP from Example 2.1*

choosing how much to budget to a set of projects or divisions [7].

Any other integer program constraint with binary variables can also easily be reformatted into a KP constraint. This is useful because it can help develop facets for the IP by finding facet-defining inequalities for the KP. The application of the knapsack problem to other IPs is important because KPs are relatively easy to solve and can provide facets quickly.

The requirements for a knapsack constraint are that it must be a $\leq$ constraint and that it must have only positive values for its coefficients and right hand side. Reformatting a constraint from $\geq$ to $\leq$ is easy, as both sides are simply multiplied by -1. Reformatting a constraint from $=$ to $\leq$ is a little more complicated, as two separate constraints have to be made: one $\geq$ and one $\leq$. The $\geq$ constraint then must be reformatted as already discussed, and two KPs must be solved separately. To reformat negative coefficients, let x_i be a variable with a negative coefficient. Let $x_i = 1 - x_i'$, and substitute x_i' into the inequality. This will give a positive coefficient for x_i', and the KP can be solved. To show how an IP might be reformulated as a KP, see the following example.

Example 2.2 Assume a general integer program needs to be solved. One of its constraints is $-12x_1 - 3x_2 + 9x_3 = 6$. To speed up solving time, it is recommended that the IP be solved with only the equality constraint as a knapsack problem.

To solve this, it will be necessary to break the constraint into two constraints: $-12x_1 - 3x_2 + 9x_3 \leq 6$ and $-12x_1 - 3x_2 + 9x_3 \geq 6$. Because the principles used in solving the second constraint can also be used on the first, this example will only look at formulating the second constraint. First, the constraint needs to be a $\leq$ constraint. By multiplying both sides by -1, the constraint becomes $12x_1 + 3x_2 - 9x_3 \leq -6$. All of the coefficients also need to be positive. To achieve positive values, let $x_3 = 1 - x_3'$. Substituting this equation into the inequality and adding 9 to both sides yields the new inequality $12x_1 + 3x_2 + 9x_3' \leq 3$. This inequality is a valid knapsack constraint, and the IP can be solved as a knapsack.

An example of a knapsack problem can be seen below.

Example 2.3 Two hikers are preparing to go on a hiking trip. They will carry all of their supplies in two knapsacks. They have assigned relative values to the fifteen items they have to choose from and have also found the weight in pounds (lbs) of each of the items. They know they can only carry a combined 114 lbs. Table 1 shows the values and weights of the fifteen items. An integer programming representation of this problem can be seen below.

Table 2.1: *Values and Weights of Items in Example 2.3*

Item	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	75	36	12	34	54	23	19	34	89	49	43	21	30	67	50
Weight	21	21	20	15	14	13	13	13	8	8	8	8	7	7	7

$$\begin{aligned}
\text{Max} \quad & 75x_1 + 36x_2 + 12x_3 + 34x_4 + 54x_5 + 23x_6 + 19x_7 + 34x_8 + 89x_9 + \\
& 49x_{10} + 43x_{11} + 21x_{12} + 30x_{13} + 67x_{14} + 50x_{15} \\
\text{s.t.} \quad & 21x_1 + 21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8 + \\
& 8x_9 + 8x_{10} + 8x_{11} + 8x_{12} + 7x_{13} + 7x_{14} + 7x_{15} \leq 114 \\
& 0 \leq x \leq 1 \text{ and integer.}
\end{aligned}$$

This example will be used for examples throughout the rest of this thesis.

2.1.2 Cover Inequalities

One of the most important types of cuts used in integer programming are cover cuts. Cover inequalities describe these cuts, and are useful because they frequently describe facets in an integer program and also can be found quickly.

A cover in a knapsack problem is a set of indices that takes the form $C \subset N$ such that $\sum_{i \in C} a_i \geq b$. Clearly, the structure of this inequality dictates that at least one variable in the equality will have to be omitted in any valid solution, so the number of variables that will be included in the solution can be at most $|C|$-1. Therefore, all covers induce a valid inequality of the form $\sum_{x \in C} x_i \leq |C| - 1$ and are called cover cuts.

The most important covers are called minimal covers. Formally, a cover C is a minimal cover if and only if $\sum_{i \in C \setminus \{j\}} a_i \leq b \; \forall \; j \in C$. In other words, a cover is minimal if and only if $C \setminus \{j\}$ is not a cover for all $j \in C$. In the knapsack example given above, the set of variables $\{1, 2, 3, 4, 5, 6, 7\}$ is a minimal cover, as any combination of six of these seven variables can be taken without violating feasibility.

An extended cover, $E(C)$, can be obtained from a cover by adding variables with higher constraint coefficients into the original cover. Formally, $E(C) = C \cup J$, where $J = \{j_1, j_2, ..., j_q\}$, and $j_1 < j_2 < ... < j_q < i_1$. The set of variables in the extended cover is $\{j_1, ..., j_q, i_1, ..., i_p\}$ and the inequality will be valid and take the form $\sum_{i \in E(C)} x_i \leq |C| - 1$.

Example 2.4 For example, take the minimal cover {2,3,4,5,6,7,8,10} from the KP given in Example 2.3. Clearly this is a minimal cover, as any combination of seven variables in the cover can be taken without violating $21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8 + 8x_{10} \leq 114$, and this cover yields the cover inequality $x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_{10} \leq 8$. Because x_1 is the only variable with a higher coefficient than all of the other coefficients in the original cover, adding {1} yields the extended cover {1,2,3,4,5,6,7,8,10}. So, the extended cover inequality is $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_{10} \leq 8$.

Extended cover inequalities are useful because they frequently yield facet-defining inequalities. Given a knapsack instance with a minimal cover C, an extended cover inequality is facet-defining if it meets one of the following conditions[22, 23]:

1. $C = N$, where N is the set of variables $i = 1, 2, ..., n$

2. $E(C) = N$ and (1) $a_1 + \sum_{i \in C \setminus \{i_1, i_2\}} a_i \leq b$

3. $C = E(C)$ and (2) $a_p + \sum_{i \in C \setminus \{i_1\}} a_i \leq b$, where $p = min\{i \in N \setminus E(C)\}$

4. $C \subset E(C) \subset N$ and (1) and (2).

In the minimal cover from Example 2.3, it can be shown that the cover inequality $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \leq 6$ is a facet-defining inequality by condition 3 above. Clearly, the cover above is also the extended cover. The minimum index p is $\{8\}$ in this case, and $a_8 = 13$. So, $a_8 + \sum_{i \in C \setminus \{i_1\}} a_i = 108$, which is less than 114. Thus, this extended cover inequality is facet defining.

2.1.3 Lifting

Lifting was introduced by Gomory [12] in 1969. Lifting seeks to improve the coefficients of a valid constraint in an integer program. In other words, given a valid inequality, a lifting process is applied to make the constraint stronger, cutting off more linear relaxation space and may eventually lead to a facet-defining inequality.

The basic idea behind lifting is to have an inequality that is valid on some small space and to increase the strength of the inequality by increasing the dimension of the underlying polytope. So, define the S restricted space of P to be $P_S = \{x \in P : x_i = 0 \ \forall \ i \in N \setminus S\}$ and let $P_S^{ch} = conv(P_S)$.

There are a number of lifting techniques in common use, including uplifting [1, 2], downlifting [5, 16, 30], sequential lifting [1, 2, 16], simultaneous lifting [11, 16], and approximate lifting. The lifting techniques that relate directly to three set inequalities are sequential uplifting and simultaneous uplifting techniques, which are explained below.

Sequential uplifting is a widely used lifting process. Let $\sum_{i=2}^{n} \alpha_i x_i \leq \beta$ be a valid inequality over $P^{ch}_{\{2,3,\ldots,n\}}$. Sequentially uplifting x_1 into this inequality seeks to yield a valid inequality of the form $\alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i x_i \leq \beta$. Solving the following optimization problem helps to determine valid values for α_1.

$$\begin{array}{ll} \text{Maximize} & \sum_{i=2}^{n} \alpha_i x_i \\ \text{Subject to:} & Ax \leq b \\ & x_1 = 1 \\ & 0 \leq x_i \leq 1 \text{ and integer.} \end{array}$$

Let Z^* be the optimal value to this integer program. Then $\alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i x_i \leq \beta$ is a valid inequality as long as $\alpha_1 \leq \beta - Z^*$.

Exact sequential lifting seeks to obtain an α_1 value that is as large as possible where $\alpha_1 = \beta - Z^*$, while approximate sequential lifting will sacrifice some strength in the inequality $\alpha_1 \leq \beta - Z^*$ in order to avoid solving the above IP.

Numerous individuals have provided fundamentally important results in sequential lifting. Wolsey [30] presented the first method to exactly sequentially lift general integer variables, which requires the solution to many IPs. Recently, Gutierrez [16] improved upon this result and general integer variables can now be sequentially lifted with a single IP.

Due to the importance of the knapsack polyhedron, numerous individuals have provided results on sequential lifting over this polyhedron. In particular, Balas provided both upper and lower bounds on lifting coefficients (Thus this can be considered an approximate sequential lifting technique). Other results by Balas and Zemel and Zemel [1, 2] go on to provide additional results. More recently some work has been done on sequentially lifting over mixed integer programs [4, 15, 25, 14].

One of the biggest advantages of sequential lifting is that if $\sum_{i=2}^{n} \alpha_i x_i \leq \beta$ defines a face of dimension r in $P^{ch}_{\{2,3,\ldots,n\}}$, then $\alpha_1 x_1 + \sum_{i=2}^{n} \alpha_i x_i \leq \beta$ defines a face of dimension at least $r+1$ in P^{ch} as long as $\alpha_1 = \beta - Z^*$, which is the theoretical advantage of exact simultaneous lifting.

Sequential lifting is frequently used on cover inequalities. The reason for this is that if you start with a minimal cover, it is possible to lift in all of the other variables and find a facet-defining inequality over the entire space. This is because a minimal cover inequality is facet defining over P^{ch}_C.

Example 2.5 For an example of sequential lifting, let us return to Example 2.3 and observe that $C = \{4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$ is a minimal cover. Lifting x_1 into this cover inequality, which is $x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \leq 11$, begins by solving.

$$
\begin{aligned}
\text{Maximize}\quad & x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \\
\text{Subject to:}\quad & 21x_1 + 21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8 \\
& +8x_9 + 8x_{10} + 8x_{11} + 8x_{12} + 7x_{13} + 7x_{14} + 7x_{15} \le 114 \\
& x_1 = 1 \\
& 0 \le x_i \le 1 \text{ and integer.}
\end{aligned}
$$

In this optimization, $Z^* = 10$. Therefore, $\alpha_1 \le \beta - Z^* = 11 - 10 = 1$. So the sequentially uplifted inequality is now $x_1+x_4+x_5+x_6+x_7+x_8+x_9+x_{10}+x_{11}+x_{12}+x_{13}+x_{14}+x_{15} \le 11$. Suppose that x_2 is the next variable lifted. So the following IP is solved.

$$
\begin{aligned}
\text{Maximize}\quad & x_1 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \\
\text{Subject to:}\quad & 21x_1 + 21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8 \\
& +8x_9 + 8x_{10} + 8x_{11} + 8x_{12} + 7x_{13} + 7x_{14} + 7x_{15} \le 114 \\
& x_2 = 1 \\
& 0 \le x_i \le 1 \text{ and integer.}
\end{aligned}
$$

Again, we get $Z^* = 10$. As before, $\alpha_8 \le \beta - Z^* = 11 - 10 = 1$, and the uplifted inequality is now $x_1 + x_2 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \le 11$. This algorithm can be repeated for x_3. The same result is achieved, and the final sequentially uplifted inequality is $x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8+x_9+x_{10}+x_{11}+x_{12}+x_{13}+x_{14}+x_{15} \le 11$.

The above inequality will be facet defining over P^{ch}. This can be easily seen by observing that it is an extended cover and meets the criteria set above. Figure 6 also shows a set of

$$\begin{vmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&1&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&0&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&0&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&0&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&0&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&0&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&0&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&0&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&0
\end{vmatrix}$$

Figure 2.6: *Matrix of Affinely Independent Points from Example 2.5*

affinely independent points that prove it is facet defining. Each of these points is represented as a column of the matrix, and this notation will be used throughout the remainder of this thesis.

It is also notable the effect lifting has on dimension of the constraint. The minimal cover inequality shows that by cyclically permuting which of the 12 variables is not taken, 12 affinely independent points can be found, from which it can be concluded that $\dim(F) \geq 11$. With each variable lifted in, the dimension of the constraint is increased by at least one by the point that finds z^*. As a result, the final $dim(F) \geq 14$.

It should be noted that the order in which a set of points is lifted into an inequality is important. Different orders may yield different inequalities, though all such inequalities are still valid and facet defining. Furthermore, the average coefficient values of these inequalities will also yield a valid inequality. Simultaneous lifting is a method developed to attempt to improve upon this average of sequentially lifted inequalities.

While sequential uplifting lifts one variable at a time into an inequality, simultaneous uplifting seeks to create a new inequality by lifting in several variables at once. Similar to sequential lifting, simultaneous lifting takes a valid inequality as input. Let $S \subset N$ and $\sum_{i\in S} \alpha_i x_i \leq \beta$ be a valid inequality of P_S^{ch}. Now let $F \subset N \setminus S$ be the set of variables being simultaneously lifted into this inequality. A simultaneously lifted inequality could take the form $\alpha \sum_{i\in F} w_i x_i + \sum_{i\in S} \alpha_i x_i \leq \beta$ where w_i is called the scaling coefficient for the i^{th} variable.

As in the case of sequential lifting, simultaneous lifting can be done exactly or approximately. Exact simultaneous lifting will provide the largest α possible, while approximate will create inequalities where α may be able to be strengthened. Very few results lift on simultaneous lifting and the only approximate techniques exist in what is known as sequence independent lifting [14, 15].

In 1981, Zemel [3] developed the first technique to simultaneously uplift sets of integer variables. This result can only simultaneously lift over binary integer programs. This method solves exponentially many integer programs and finds some associated extreme points to a linear relaxation space. The end result is all of the facet-defining inequalities that could be obtained from lifting over the starting inequality. Clearly, this is intractable even on the fastest computers.

In 2007, Gutierrez [16] presented two alternate techniques to simultaneously lift general integer variables. Her methods could either require the solution to one or many integer programs. The result is a single valid inequality. Unlike Zemel's results, her technique is only guaranteed to increase the dimension of the face by one. Her method using many integer programs is used to demonstrate how to simultaneously uplift general integer variables. The following IP is critical to using this method.

$$\begin{array}{ll}
\text{Maximize} & \alpha \sum_{i \in F} w_i x_i + \sum_{i \in S} \alpha_i x_i \\
\text{Subject to:} & Ax \leq b \\
 & \sum_{i \in F} x_i \geq 1 \\
 & 0 \leq x_i \leq 1 \text{ and integer.}
\end{array}$$

The method begins by assigning α to M, a very large initial value. If the optimal solution to the IP is larger than β, then the optimal x^* from the IP is used to solve for α. In other words, $\alpha = \frac{\beta - \sum_{i \in S} \alpha_i x_i^*}{\sum_{i \in F} w_i x_i^*}$. This new α is input into the objective function and the IP is resolved. This is repeated until $Z^* \leq \beta$, which indicates that the inequality $\alpha \sum_{i \in F} w_i x_i + \sum_{i \in S} \alpha_i x_i \leq \beta$ is valid.

Example 2.6 Returning to Example 2.3, we have a cover inequality of $x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \leq 11$ that is valid. All three variables that are not in this inequality are simultaneously lifted with the scaling coefficients all set to 1 ($w_1 = w_2 = w_3 = 1$). This procedure begins by solving

Maximize $\quad 10000(x_1 + x_2 + x_3) + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} +$

$x_{12} + x_{13} + x_{14} + x_{15}$

Subject to: $\quad 21x_1 + 21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8$

$+8x_9 + 8x_{10} + 8x_{11} + 8x_{12} + 7x_{13} + 7x_{14} + 7x_{15} \leq 114$

$\sum_{i=1}^{3} x_i \geq 1$

$0 \leq x_i \leq 1$ and integer.

Solving this IP results in $Z^* = 30006 > 11$ and $x^* = (1,1,1,0,0,0,0,0,0,1,1,1,\ 1,1,1)$. Solving $6 + 3\alpha = 11$ for α, the new α is 1.6667. This value is put back into the original IP, and it is resolved.

Maximize $\quad 1.6667(x_1 + x_2 + x_3) + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} +$

$x_{12} + x_{13} + x_{14} + x_{15}$

Subject to: $\quad 21x_1 + 21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8$

$+8x_9 + 8x_{10} + 8x_{11} + 8x_{12} + 7x_{13} + 7x_{14} + 7x_{15} \leq 116$

$\sum_{i=1}^{3} x_i \geq 1$

$0 \leq x_i \leq 1$ and integer

The new solution is $Z^* = 11.6667 > 11$ with $x^* = (0,0,1,0,0,1,1,1,1,1,1,1,1,\ 1,1)$. Solving $10+\alpha = 11$ for α, we get $\alpha = 1$. Putting this value back into the IP yields a $Z^* = 11$ and so the inequality is valid. The algorithm returns the inequality $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} \leq 11$. As seen above, this inequality is valid and facet-defining.

As this example shows, sequential and simultaneous lifting do not always yield especially useful inequalities. In both of these examples, the inequalities are extended cover inequalities. The main idea behind three set inequalities will seek to use the principles of these methods to yield stronger and possibly more useful facet-defining inequalities.

Chapter 3 - Three Set Inequalities

The remainder of this thesis is dedicated to explaining three set inequalities (TSIs) and their usefulness. First, the theory behind three set inequalities and an algorithm, called Three Set Inequalities Algorithm (TSIA), for finding TSIs will be presented. Then, circumstances under which TSIs are facet defining will be shown. Finally, computational results using TSIs compared to standard IP methods will be presented in Chapter 4.

Prior to giving TSIA, a few definitions and notations are required. Let $J \subset N$, $K \subset N$, and $L \subset N$ where J, K, and L are mutually exclusive sets, $J \cap K = \emptyset$, $K \cap L = \emptyset$, and $J \cap L = \emptyset$. Let α_J, α_K, and α_L be any real numbers. Then, the three set inequality over sets J, K, and L with parameters α_J, α_K, and α_L is defined as.

$$TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L} = \alpha_J \sum_{j \in J} x_j + \alpha_K \sum_{k \in K} x_k + \alpha_L \sum_{l \in L} x_l \leq 1.$$

A key step in TSIA is to find a $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$ hyperplane passing through three feasible integer solutions. The following definitions are used to define this hyperplane. For any $x \in P$, define $s'_J(x)$, $s'_K(x)$, and $s'_L(x)$ to be $|\{x_j : x_j = 1, j \in J\}|$, $|\{x_k : x_k = 1, k \in K\}|$, and $|\{x_l : x_l = 1, l \in L\}|$, respectively.

The input to TSIA is the feasible region of a binary integer program $\{x \in \{0,1\}^n : Ax \leq b\}$, mutually exclusive sets J, K, and $L \subset N$, and three initial feasible integer solutions x^1, x^2, and x^3 such that the vectors given by $(s'_J(x^1)$, $s'_K(x^1)$, $s'_L(x^1))$; $(s'_J(x^2)$,

$s'_K(x^2)$, $s'_L(x^2)$); and ($s'_J(x^3)$, $s'_K(x^3)$, $s'_L(x^3)$) are linearly independent. Since these points are linearly independent, there exists a $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$ hyperplane that passes through all of these points. The values of α_J, α_K and α_L can be obtained by solving the following system of equations.

$$\begin{vmatrix} s'_J(x^1) & s'_K(x^1) & s'_L(x^1) \\ s'_J(x^2) & s'_K(x^2) & s'_L(x^2) \\ s'_J(x^3) & s'_K(x^3) & s'_L(x^3) \end{vmatrix} \begin{vmatrix} \alpha_J \\ \alpha_K \\ \alpha_L \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}.$$

The condition that the points induce linearly independent vectors may seem too restrictive because all that is needed is affine independence to generate a hyperplane. However, if the points are only affinely independent, then either there exists no solution to the above system of equations or an infinite number of solutions. For instance, if the three points are $(1, 0, 0)$, $(0, 1, 0)$ and $(1, 1, 0)$, then there is no solution and the appropriate hyperplane is $x_3 = 0$, which is not a TSI inequality. In the infinite number of solutions case, there exists a row which is redundant and can be removed. However, this implies that the original three points are not affinely independent, which violates the above assumption and should never occur. Therefore, TSIA assumes that the three points are linearly independent, but some obvious modifications could be made and some alternate valid inequalities could be generated by this algorithm.

Finding three such linearly independent points is clearly $\mathcal{NP}$-complete, since just finding a single integer solution is $\mathcal{NP}$-complete [21]. However, in the case of a knapsack instance, such points can easily be generated. These three feasible points generate an α_J, α_K and α_L by solving the above equations. TSIA assumes that $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L} \alpha_J \sum_{j \in J} x_j + \alpha_K \sum_{k \in K} x_k + \alpha_L \sum_{l \in L} x_l \leq 1$ is a valid inequality, which is verified or shown to be false by solving the following integer program.

$$\begin{array}{ll} \text{Maximize} & \alpha_J \sum_{j \in J} x_j + \alpha_K \sum_{k \in K} x_k + \alpha_L \sum_{l \in L} x_l \\ \text{Subject to:} & Ax \leq b \\ & x \in \{0,1\}^N. \end{array}$$

If the optimal solution is less than or equal to one, then $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$ is a valid inequality. If not, then the x^* from this IP violates the inequality and it will be used to provide new values for α_J, α_K, and α_L. A more detailed discussion of the geometry of this concept will be provided later in this chapter. Formally, TSIA is

The Three Set Inequality Algorithm (TSIA)

Input:

The feasible region of a binary integer program $P = \{x \in \{0,1\}^n : Ax \leq b\}$.

Mutually exclusive sets J, K, $L \subset N$.

x^1, x^2, $x^3 \in P$ such that the vectors given by $(s'_J(x^1), s'_K(x^1), s'_L(x^1))$,

$(s'_J(x^2), s'_K(x^2), s'_L(x^2))$, and $(s'_J(x^3), s'_K(x^3), s'_L(x^3))$ are linearly independent.

Initialization:

```
        $z^* := 2$.
```

Main Step:

```
    while $z^* > 1$ do.
        Solve the following system of equations.
```

$$\begin{vmatrix} s'_J(x^1) & s'_K(x^1) & s'_L(x^1) \\ s'_J(x^2) & s'_K(x^2) & s'_L(x^2) \\ s'_J(x^3) & s'_K(x^3) & s'_L(x^3) \end{vmatrix} \begin{vmatrix} \alpha_J \\ \alpha_K \\ \alpha_L \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}.$$

```
        Solve the following IP generating $z^*$ and $x^*$.
```

$$\begin{array}{ll} \text{Maximize} & \alpha_J \sum_{j\in J} x_j + \alpha_K \sum_{k\in K} x_k + \alpha_L \sum_{l\in L} x_l \\ \text{Subject to:} & Ax \le b \\ & x \in \{0,1\}^N. \end{array}$$

```
        If $z^* > 1$ then.
            Select some $p \in \{1, 2, 3\}$ such that when $x^p$ is replaced by $x^*$, the
                vectors $(s'_J(x^1), s'_K(x^1), s'_L(x^1))$, $(s'_J(x^2), s'_K(x^2), s'_L(x^2))$, and
                $(s'_J(x^3), s'_K(x^3), s'_L(x^3))$ are linearly independent.
    end (while).
```

Termination:

Report $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L} = \alpha_J \sum_{j\in J} x_j + \alpha_K \sum_{k\in K} x_k + \alpha_L \sum_{l\in L} x_l \leq 1$ as a valid inequality.

This algorithm has several theoretical and computational issues, which will be discussed in detail near the end of this chapter. The following KP example is used to show TSIA.

Example 3.1 To better understand the process used in this algorithm, it is useful to look at the following example. Consider the KP presented in Example 2.3, which is restated here.

$$\begin{aligned} \text{Max} \quad & 75x_1 + 36x_2 + 12x_3 + 34x_4 + 54x_5 + 23x_6 + 19x_7 + 34x_8 + 89x_9 + \\ & 49x_{10} + 43x_{11} + 21x_{12} + 30x_{13} + 67x_{14} + 50x_{15} \\ \text{s.t.} \quad & 21x_1 + 21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8 + \\ & 8x_9 + 8x_{10} + 8x_{11} + 8x_{12} + 7x_{13} + 7x_{14} + 7x_{15} \leq 114 \\ & 0 \leq x \leq 1 \text{ and integer.} \end{aligned}$$

The three selected sets are $J = \{1, 2, 3\}$, $K = \{4, 5, 6, 7, 8\}$, and $L = \{9, 10, 11, 12, 13, 14, 15\}$. The starting three feasible points are identified as $x^1 = (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, $x^2 = (0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0)$, and $x^3 = (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1)$. Using these as the input, the values of the s's can be generated as $s'_J(x^1) = 3$, $s'_K(x^1) = 0$, and $s'_L(x^1) = 0$; $s'_J(x^2) = 0$, $s'_K(x^2) = 5$, and $s'_L(x^2) = 0$; $s'_J(x^3) = 0$, $s'_K(x^3) = 0$, and $s'_L(x^3) = 7$. Clearly, the three corresponding vectors are linearly independent. From these values, the following system of equations is generated and solved for the α's.

$$\begin{vmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 7 \end{vmatrix} \begin{vmatrix} \alpha_J \\ \alpha_K \\ \alpha_L \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}.$$

The values of α's are $\alpha_J = 0.333$, $\alpha_K = 0.200$, and $\alpha_L = 0.143$. These values are entered into an IP solver, generating the following IP.

$$\begin{aligned} \text{Max} \quad & 0.333\sum_{j=1}^{3} x_j + 0.200\sum_{k=4}^{8} x_k + 0.143\sum_{l=9}^{15} x_l \\ \text{s.t.} \quad & 21x_1 + 21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8 + \\ & 8x_9 + 8x_{10} + 8x_{11} + 8x_{12} + 7x_{13} + 7x_{14} + 7x_{15} \leq 114 \\ & 0 \leq x \leq 1 \text{ and integer.} \end{aligned}$$

The solution to this IP is $z^* = 1.933$ at the point $x^* = (0,0,1,0,0,1,1,1,1,1,1,\ 1,1,1,1)$. This point clearly violates $TSI^{J,K,L}_{0.333,0.2,0.143} = 0.333\sum_{j\in J} x_j + 0.2\sum_{k\in K} x_k + 0.143\sum_{l\in L} x_l \leq 1$, because putting x^* into the left side of $TSI^{J,K,L}_{0.333,0.2,0.143}$ yields a value of 1.933, which is greater than 1. Replacing the point $x^3 = (0,0,0,0,0,0,0,0,\ 1,1,1,1,1,1,1)$ with this optimal point, a new set of s's is generated as $s'_J(x^1) = 3$, $s'_K(x^1) = 0$, and $s'_L(x^1) = 0$; $s'_J(x^2) = 0$, $s'_K(x^2) = 5$, and $s'_L(x^2) = 0$; $s'_J(x^3) = 1$, $s'_K(x^3) = 3$, and $s'_L(x^3) = 7$. Using these values the following system of equations is solved to find the new values of α.

$$\begin{vmatrix} 3 & 0 & 0 \\ 0 & 5 & 0 \\ 1 & 3 & 7 \end{vmatrix} \begin{vmatrix} \alpha_J \\ \alpha_K \\ \alpha_L \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}.$$

Solving this system of equations results in $\alpha_J = 0.333$, $\alpha_K = 0.200$, and $\alpha_L = 0.009524$. Using the new set of α's, the new IP seen below is formulated and solved.

$$\begin{array}{ll} \text{Max} & 0.333\sum_{j=1}^{3} x_j + 0.200\sum_{k=4}^{8} x_k + 0.009524\sum_{l=9}^{15} x_l \\ \text{s.t.} & 21x_1 + 21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8 + \\ & 8x_9 + 8x_{10} + 8x_{11} + 8x_{12} + 7x_{13} + 7x_{14} + 7x_{15} \leq 114 \\ & 0 \leq x \leq 1 \text{ and integer.} \end{array}$$

This time the solution found is $z^* = 1.667$, with the point $x^* = (0,1,1,1,1,1,1,1,0,0,0,0,0,0,0)$, which would clearly violate $TSI^{J,K,L}_{0.333,0.2,0.009524} = 0.333\sum_{j\in J} x_j + 0.2\sum_{k\in K} x_k + 0.009524\sum_{l\in L} x_l \leq 1$. The point, $x^2 = (0,0,0,1,1,1,1,1,0,0,0,0,0,\ 0,0)$, is replaced by x^* and the new system of equations is as follows.

$$\begin{vmatrix} 3 & 0 & 0 \\ 2 & 5 & 0 \\ 1 & 3 & 7 \end{vmatrix} \begin{vmatrix} \alpha_J \\ \alpha_K \\ \alpha_L \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}.$$

The new set of α's is $\alpha_J = 0.333$, and $\alpha_K = \alpha_L = 0.0667$. Again, a new IP is formulated and solved.

$$
\begin{aligned}
\text{Max}\quad & 0.333\textstyle\sum_{j=1}^{3} x_j + 0.0667\sum_{k=4}^{8} x_k + 0.0667\sum_{l=9}^{15} x_l \\
\text{s.t.}\quad & 21x_1 + 21x_2 + 20x_3 + 15x_4 + 14x_5 + 13x_6 + 13x_7 + 13x_8 + \\
& 8x_9 + 8x_{10} + 8x_{11} + 8x_{12} + 7x_{13} + 7x_{14} + 7x_{15} \leq 114 \\
& 0 \leq x \leq 1 \text{ and integer.}
\end{aligned}
$$

The new solution is $z^* = 1.400$, and $x^* = (1,1,1,0,0,0,0,0,1,1,1,0,1,1,1)$. The point $x^1 = (1,1,1,0,0,0,0,0,0,0,0,0,0,0,0)$ is replaced with x^* and the following system of equations is solved.

$$
\begin{vmatrix} 3 & 0 & 6 \\ 2 & 5 & 0 \\ 1 & 3 & 7 \end{vmatrix}
\begin{vmatrix} \alpha_J \\ \alpha_K \\ \alpha_L \end{vmatrix}
=
\begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix}.
$$

This time, $\alpha_J = 0.207$, $\alpha_K = 0.117$, and $\alpha_L = 0.063$. The IP is reformulated and solved in the same way as has been seen above. The best solution found is $z^* = 1.054$ at $x^* = (1,1,1,0,0,0,0,1,1,1,0,1,1,0,1)$. The oldest point, $x^3 = (0,1,0,1,0,0,1,1,1,1,1,1,1,1,1)$ is replaced.

The corresponding new system of equations is solved as seen before to find $\alpha_J = -0.333$, $\alpha_K = 0.333$, and $\alpha_L = 0.333$. The new $z^* = 3.667$, and the new $x^* = (0,0,0,1,1,1,1,1,1,1,1,1,1,1,0)$. Again, the oldest point $x^2 = (0,1,1,1,1,1,1,1,0,\ 0,0,0,0,0,0)$ is replaced, and the new values of α are found to be $\alpha_J = 0.152$, $\alpha_K = 0.091$, and $\alpha_L = 0.091$.

Solving the new IP yields $z^* = 1.061$ and $x^* = (0,0,1,0,0,1,1,1,1,1,1,1,1,1,1)$, which

Table 3.1: *Iteration Summary for Example 3.1*

x^1	x^2	x^3	α_J	α_K	α_L	z^*	x^*	x^i out
$(3,0,0)$	$(0,5,0)$	$(0,0,7)$	0.333	0.2	0.143	1.93	$(1,3,7)$	x^3
$(3,0,0)$	$(0,5,0)$	$(1,3,7)$	0.333	0.2	0.00952	1.67	$(2,5,0)$	x^2
$(3,0,0)$	$(2,5,0)$	$(1,3,7)$	0.333	0.0667	0.0667	1.4	$(3,0,6)$	x^1
$(3,0,6)$	$(2,5,0)$	$(1,3,7)$	0.207	0.117	0.063	1.05	$(3,1,5)$	x^3
$(3,0,6)$	$(2,5,0)$	$(3,1,5)$	-0.333	0.333	0.333	3.67	$(0,5,6)$	x^2
$(3,0,6)$	$(0,5,6)$	$(3,1,5)$	0.152	0.091	0.091	1.06	$(1,3,7)$	x^1
$(1,3,7)$	$(0,5,6)$	$(3,1,5)$	0.1875	0.125	0.0625	1.00	$(1,3,7)$	none

replaces $x^1 = (1,1,1,0,0,0,0,0,0,1,1,1,1,1,1)$ in the system of equations. The solution to the system of equations gives the new values of the α's as $\alpha_J = 0.1875$, $\alpha_K = 0.125$, and $\alpha_L = 0.0625$.

Solving the new IP results in $z^* = 1$, so the algorithm is finished and reports the valid inequality $TSI^{IJK}_{0.1875,0.125,0.0625} = 0.1875\sum_{j=1}^{3} x_j + 0.125\sum_{k=4}^{8} x_k + 0.0625\sum_{l=9}^{15} x_l \leq 1$. Table 2 provides a summary of this example.

To show that fifteen affinely independent points can be generated from this solution, see Figure 7. As in the previous point diagram, each point is represented by a column in the matrix. These points show that this inequality is facet defining.

$$
\begin{array}{c|ccc|ccccc|ccccccc}
 & \multicolumn{3}{c|}{J} & \multicolumn{5}{c|}{K} & \multicolumn{7}{c}{L} \\
\hline
 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
J & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
K & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
L & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0
\end{array}
$$

Figure 3.1: *Matrix of Affinely Independent Points for* $TSI^{IJK}_{0.1875,0.125,0.0625}$

3.1 Extensions and Issues of TSIA

The nature of TSIA causes both some exciting avenues for extensions and also creates some problems. This section describes a technique to generate many TSIA inequalities through an enumerative branching tree and also describes some of TSIA's shortcomings such as not necessarily terminating.

It is important to note that a couple of factors will affect what valid inequality TSIA finds. Initial α values, which are directly related to the initial feasible integer points, and the order of replacement will affect what inequalities are found. An enumeration tree can be applied to this method to see all possible facet-defining inequalities that can be found from a given starting set of α's and a given set of initial points.

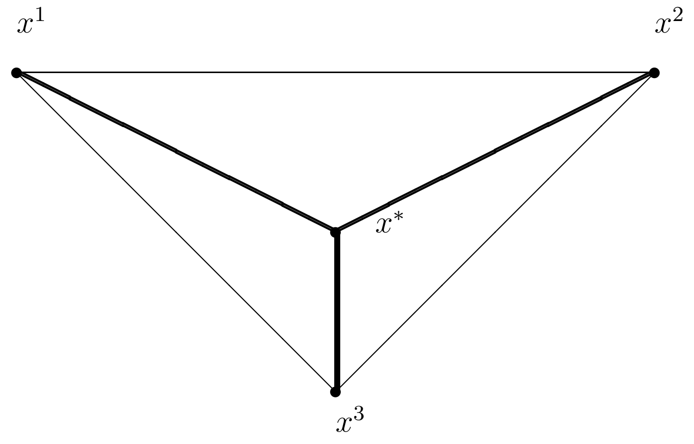

Figure 3.2: *Example of a Tetrahedron Created by Four Points*

The importance of order becomes clearer when the geometry of TSIA is considered. Observe that the points that satisfy $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$, $\alpha_J \sum_{j\in J} x_j + \alpha_K \sum_{k\in K} x_k + \alpha_L \sum_{l\in L} x_l \leq 1$ at equality define a hyperplane. If the solution to the IP from TSIA is larger than 1, then there exists a point, x^*, that violates $\alpha_J \sum_{j\in J} x_j + \alpha_K \sum_{k\in K} x_k + \alpha_L \sum_{l\in L} x_l \leq 1$. Examining the four points x^1, x^2, x^3 and x^* in three dimension results in a tetrahedron as shown in Figure 8.

This tetrahedron can be represented by 4 hyperplanes. Clearly, the hyperplane going through x^1, x^2 and x^3 is not valid and can be defined by $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$ at equality. If TSIA drops x^1, then the next value of α_J, α_K and α_L will be such that $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$ at equality crosses through the points x^*, x^2 and x^3. Similarly two other hyperplanes could be generated, which leads to the idea of an enumerative branching tree that could determine all TSI inequalities from a given set of starting points.

The following example describes this branching tree in detail. Clearly, each unfathomed node will have three child nodes representing the three hyperplanes that could be generated at that iteration. As a result, this enumeration tree is trinary. Because the tree increases in size by a factor of three with each successive level, only the path through the tree and each node's sibling nodes are displayed. The current method for choosing which node to branch on is a simple rotation.

Example 3.2 In Example 3.1, the basic three set inequality algorithm is used to find a single inequality. Now, this same algorithm is used in combination with a branching tree on the same problem to show that multiple valid inequalities can be found for the same problem depending upon which x_i is replaced in each iteration. As before, $x^1 = (1,1,1,0,0,0,0,0,0,0,0,0,0,0,0)$, $x^2 = (0,0,0,1,1,1,1,1,0,0,0,0,0,0,0)$, and $x^3 = (0,0,0,0,0,0,0,0,1,1,1,1,1,1,1)$ are used as the initial feasible points, which yields $\alpha_J = 0.333$, $\alpha_K = 0.2$, and $\alpha_L = 0.143$. The branching tree corresponding to Example 3.1 is shown in Figure 9. Each node shows the current three points, the current α_J, α_K, and α_L, and the current solution z^*. Fathomed nodes are marked with valid solutions and are highlighted. A depth-first process is used to find a solution, and the tree is drawn to reflect this process.

Each node represents a hyperplane described by a TSI inequality. When a point is found that violates the TSI inequality, three new hyperplanes are generated, which is why three child nodes are created for each parent node.

As can be seen in Figure 9, the substitution pattern used in the TSIA example is cyclical, replacing the oldest point in the set each time. Note that in node 7 of Figure 9, an inequality

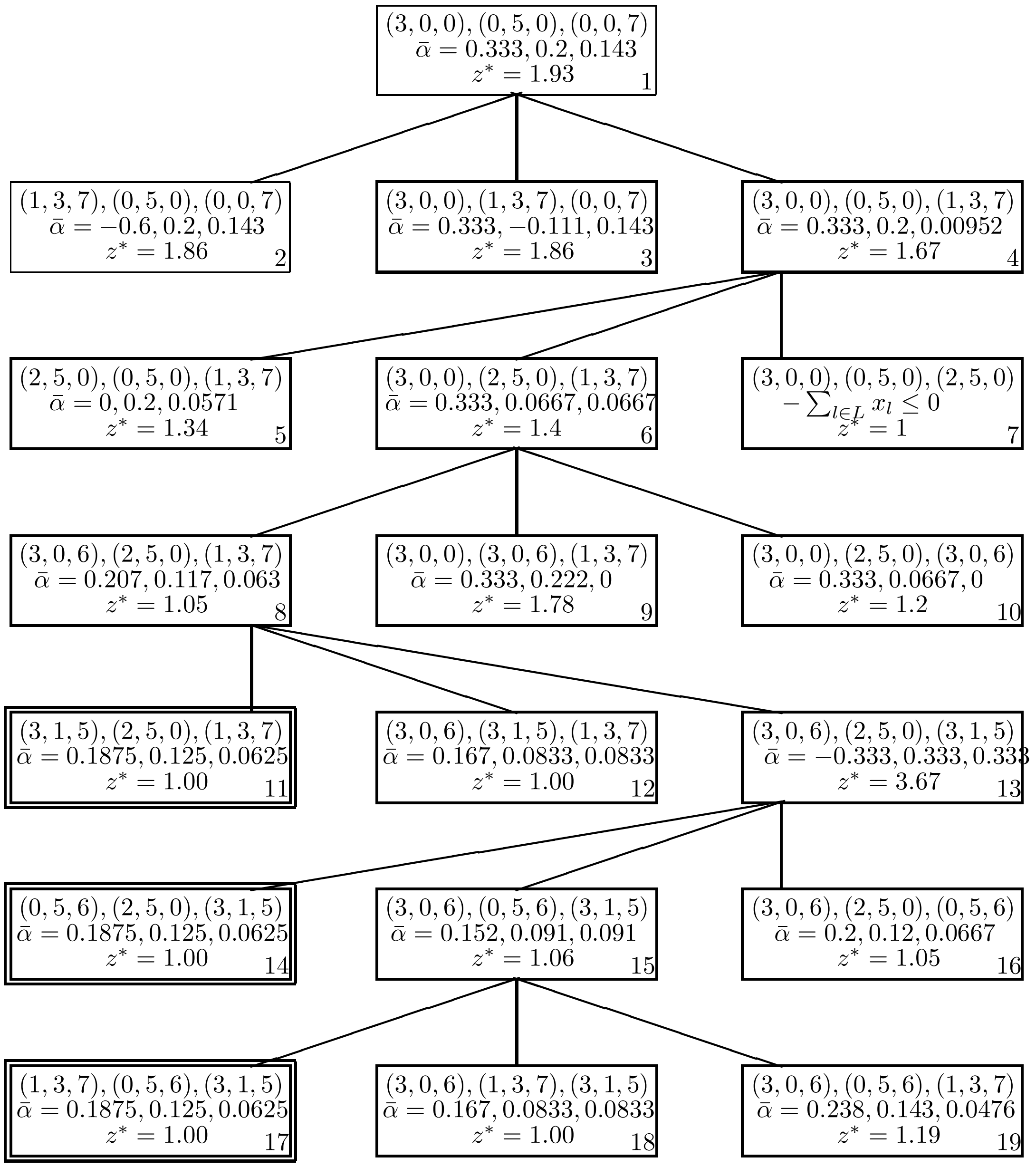
$(3,0,0),(0,5,0),(0,0,7)$ $\bar{\alpha}=0.333,0.2,0.143$ $z^*=1.93$ 1
$(1,3,7),(0,5,0),(0,0,7)$ $\bar{\alpha}=-0.6,0.2,0.143$ $z^*=1.86$ 2
$(3,0,0),(1,3,7),(0,0,7)$ $\bar{\alpha}=0.333,-0.111,0.143$ $z^*=1.86$ 3
$(3,0,0),(0,5,0),(1,3,7)$ $\bar{\alpha}=0.333,0.2,0.00952$ $z^*=1.67$ 4
$(2,5,0),(0,5,0),(1,3,7)$ $\bar{\alpha}=0,0.2,0.0571$ $z^*=1.34$ 5
$(3,0,0),(2,5,0),(1,3,7)$ $\bar{\alpha}=0.333,0.0667,0.0667$ $z^*=1.4$ 6
$(3,0,0),(0,5,0),(2,5,0)$ $-\sum_{l\in L} x_l \leq 0$ $z^*=1$ 7
$(3,0,6),(2,5,0),(1,3,7)$ $\bar{\alpha}=0.207,0.117,0.063$ $z^*=1.05$ 8
$(3,0,0),(3,0,6),(1,3,7)$ $\bar{\alpha}=0.333,0.222,0$ $z^*=1.78$ 9
$(3,0,0),(2,5,0),(3,0,6)$ $\bar{\alpha}=0.333,0.0667,0$ $z^*=1.2$ 10
$(3,1,5),(2,5,0),(1,3,7)$ $\bar{\alpha}=0.1875,0.125,0.0625$ $z^*=1.00$ 11
$(3,0,6),(3,1,5),(1,3,7)$ $\bar{\alpha}=0.167,0.0833,0.0833$ $z^*=1.00$ 12
$(3,0,6),(2,5,0),(3,1,5)$ $\bar{\alpha}=-0.333,0.333,0.333$ $z^*=3.67$ 13
$(0,5,6),(2,5,0),(3,1,5)$ $\bar{\alpha}=0.1875,0.125,0.0625$ $z^*=1.00$ 14
$(3,0,6),(0,5,6),(3,1,5)$ $\bar{\alpha}=0.152,0.091,0.091$ $z^*=1.06$ 15
$(3,0,6),(2,5,0),(0,5,6)$ $\bar{\alpha}=0.2,0.12,0.0667$ $z^*=1.05$ 16
$(1,3,7),(0,5,6),(3,1,5)$ $\bar{\alpha}=0.1875,0.125,0.0625$ $z^*=1.00$ 17
$(3,0,6),(1,3,7),(3,1,5)$ $\bar{\alpha}=0.167,0.0833,0.0833$ $z^*=1.00$ 18
$(3,0,6),(0,5,6),(1,3,7)$ $\bar{\alpha}=0.238,0.143,0.0476$ $z^*=1.19$ 19

Figure 3.3: *Branching Path for Example 3.1*

is found in the form $-\sum_{l\in L} x_l \leq 0$. Although this is a valid inequality, it is not a TSI inequality, so that node is fathomed. This occurs as a result of the three points not being linearly independent as discussed previously.

Also in Figure 9, a solution is found at node 18, in the same level as the TSI solution given. This solution is actually a simultaneously or sequentially lifted cover inequality. The inequality found is $0.167\sum_{j\in J} x_j + 0.0833\sum_{k\in K} x_k + 0.0833\sum_{l\in L} x_l \leq 1$. Thus, one exciting extension of TSIA is to generate numerous different classes of inequalities and not just TSI inequalities.

This example of TSIA found the valid inequality described by $TSI^{J,K,L}_{0.1875,0.125,0.0625} = 0.1875\sum_{j\in J} x_j + 0.125\sum_{k\in K} x_k + 0.0625\sum_{l\in L} x_l \leq 1$. This is not necessarily the only TSI inequality in the tree, it is merely the first one found.

Another noteworthy result of this tree is the fact that the TSI inequality found was actually generated in one of the child nodes in each of the previous two levels, nodes 11 and 14. The substitution method used in the TSIA example caused the algorithm to miss these inequalities. The ability to identify solutions and choice of substitution pattern are potential areas of improvement that exist for TSIA and could allow it to find valid inequalities faster.

One other interesting finding of TSI inequalities is that the inequalities yielded are not necessarily based on a cover inequality. This is significant, as the majority of the algorithms that exist produce cover-based inequalities.

Another shortcoming of TSIA is that it can only obtain inequalities that are ≤ 1. Inequalities less than or equal to 0 are ignored in this algorithm. In some cases, such as node 7 in Figure 9, an alternate hyperplane could be found that has the right hand side equal to 0. This would allow TSIA to find more valid inequalities.

Although TSIA in this example produces a valid inequality, this may not always be the case. The biggest problem with TSIA is the potential for cycling. That is, in certain areas of this enumerative branching tree one could continue to repeat and never terminate. Figure 10 has an example of TSIA that cycles. Notice that node 6 is identical to node 9 and thus this branching tree could continue indefinitely.

At this point, a complete explanation of cycling is not available, but examining the geometry may describe why this unfortunate phenomenon occurs. If during TSIA two points are used that are not on the same facet of P^{ch}, then the TSI inequality can never be valid. Therefore, if neither of these points is replaced, then cycling must occur. Thus, finding adjacent facet points would help TSIA. Some work on this has been done by Huschka [20].

Due to this reason, TSIA always replaces the oldest point in an effort to avoid keeping two nonadjacent facet points. An open question is whether or not TSIA with this replacement strategy will actually terminate. In the author's belief, such a cycling example does exist, but would be difficult to find.

Although there are a number of problems with TSIA, there are clearly a number of ways to improve upon the method. The algorithm yields a number of valid inequalities, but in order to truly be useful, these inequalities need to induce strong faces.

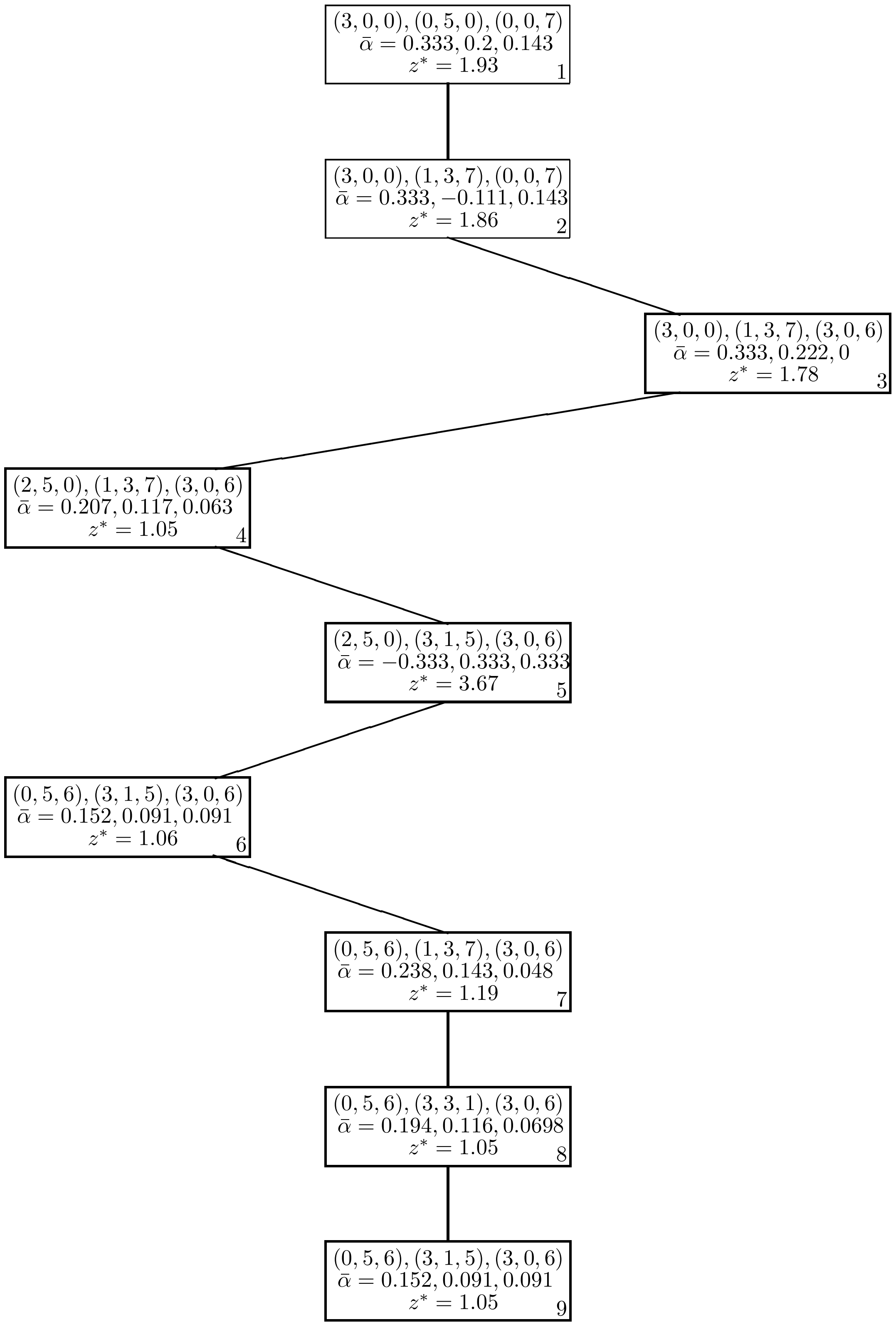

Figure 3.4: *Cycling Branching Path for Example 3.1*

3.2 Theoretical Results

In order to show that these valid inequalities are useful, conditions under which the TSIA will yield facets for the knapsack polyhedron are presented in this chapter. Even if these conditions are not met, the following theorem provides a lower bound on the dimension of the induced face and also shows that the inequality found is valid.

Theorem 3.3 *Every TSI inequality reported from TSIA is valid and induces a face of dimension at least 2.*

Proof: Let $\alpha_J \sum_{j\in J} x_j + \alpha_K \sum_{k\in K} x_k + \alpha_L \sum_{l\in L} x_l \leq 1$ be an inequality reported from TSIA. In order for TSIA to report an inequality, the solution to $z^* = \max \alpha_J \sum_{j\in J} x_j + \alpha_K \sum_{k\in K} x_k + \alpha_L \sum_{l\in L} x_l$ subject to $x \in P$ has $z^* \leq 1$. Thus there does not exist a point in P that violates $\alpha_J \sum_{j\in J} x_j + \alpha_K \sum_{k\in K} x_k + \alpha_L \sum_{l\in L} x_l \leq 1$ and so this inequality is valid.

The points x^1, x^2 and x^3 are all feasible and due to the choices of α_J, α_K and α_L each of these points satisfy $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$ at equality. These points are clearly affinely independent and so the dimension of the induced face is at least 3 -1=2.

□

This result states that every TSI inequality for any polytope induces a face of dimension at least 2. Besides providing a general new class of cutting planes, TSI inequalities have also expanded the knowledge around the knapsack polytope. The following theorem shows that TSI inequalities are a new class of facet-defining inequalities for the knapsack polytope. One important fact is that these inequalities are not based upon cover inequalities.

Before this result can be given, a few definitions are necessary. First, observe that linear independence of the s' vectors is slightly too unrestrictive for the results here. So for any $x \in P$ define $t'_J(x) = s'_J(x)$ if $s'_J(x) < |J|$ and 0 if $s'_J(x) = |J|$; $t'_K(x) = s'_K(x)$ if $s'_K(x) < |K|$ and 0 if $s'_K(x) = |K|$; and $t'_L(x) = s'_L(x)$ if $s'_L(x) < |L|$ and 0 if $s'_L(x) = |L|$.

A key component for the theoretical result is that the vectors $(t'_J(x^1),\, t'_K(x^1),\, t'_L(x^1))$, $(t'_J(x^2),\, t'_K(x^2),\, t'_L(x^2))$, and $(t'_J(x^3),\, t'_K(x^3),\, t'_L(x^3))$ are linearly independent. To help with the results, assume that x^1, x^2 and x^3 are given in such an order that no swapping of rows are necessary to get the t' vectors into reduced row echelon form. Thus, the x^1 point can be used to represent points in J, x^2 for K, and x^3 for L. The following theorem presents a new class of facets for the knapsack polyhedron.

Theorem 3.4 *From a knapsack problem, let $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$ be an inequality reported from TSIA with corresponding points x^1, x^2 and x^3. If the following ten conditions are met, then $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$ defines a facet in $P^{ch}_{KP\ J\cup K\cup L}$.*

(a) The vectors $(t'_J(x^1),\, t'_K(x^1),\, t'_L(x^1))$, $(t'_J(x^2),\, t'_K(x^2),\, t'_L(x^2))$, and $(t'_J(x^3),$

$t'_K(x^3),\, t'_L(x^3))$ are linearly independent.

(b_1) If $s'_J(x^1) = 1$, then the set $\{j_1\} \cup \{k_{|K|-s'_K(x^1)+1}, ..., k_{|K|}\} \cup \{l_{|L|-s'_L(x^1)+1}, ..., l_{|L|}\}$

is not a cover.

(b_2) If $s'_J(x^1) = |J| - 1$, then the set $\{j_1, ..., j_{|J|-1}\} \cup \{k_{|K|-s'_K(x^1)+1}, ..., k_{|K|}\} \cup$

$\{l_{|L|-s'_L(x^1)+1}, ..., l_{|L|}\}$ is not a cover.

(b_3) If $s'_J(x^1) \geq 2$ and $s'_J(x^1) \leq |J|-2$, then the sets $\{j_1, j_{|J|-s'_J(x^1)+2}, j_{|J|-s'_J(x^1)+3}, ..., j_{|J|}\} \cup \{k_{|K|-s'_K(x^1)+1}, ..., k_{|K|}\} \cup \{l_{|L|-s'_L(x^1)+1}, ..., l_{|L|}\}$ *and* $\{j_{|J|-s'_J(x^1)}, ..., j_{|J|-1}\} \cup \{k_{|K|-s'_K(x^1)+1}, ..., k_{|K|}\} \cup \{l_{|L|-s'_L(x^1)+1}, ..., l_{|L|}\}$ *are not covers.*

(c_1) If $s'_K(x^2) = 1$, then the set $\{k_1\} \cup \{j_{|J|-s'_J(x^2)+1}, ..., j_{|J|}\} \cup \{l_{|L|-s'_L(x^2)+1}, ..., l_{|L|}\}$ *is not a cover.*

(c_2) If $s'_K(x^2) = |K|-1$, then the set $\{k_1, ..., k_{|K|-1}\} \cup \{j_{|J|-s'_J(x^2)+1}, ..., j_{|J|}\} \cup \{l_{|L|-s'_L(x^2)+1}, ..., l_{|L|}\}$ *is not a cover.*

(c_3) If $s'_K(x^2) \geq 2$ and $s'_K(x^2) \leq |K|-2$, then the sets $\{k_1, k_{|K|-s'_K(x^2)+2}, k_{|K|-s'_K(x^2)+3}, ..., k_{|K|}\} \cup \{j_{|J|-s'_J(x^2)+1}, ..., j_{|J|}\} \cup \{l_{|L|-s'_L(x^2)+1}, ..., l_{|L|}\}$ *and* $\{k_{|K|-s'_K(x^2)}, ..., k_{|K|-1}\} \cup \{j_{|J|-s'_J(x^2)+1}, ..., j_{|J|}\} \cup \{l_{|L|-s'_L(x^2)+1}, ..., l_{|L|}\}$ *are not covers.*

(d_1) If $s'_L(x^3) = 1$, then the set $\{l_1\} \cup \{j_{|J|-s'_J(x^3)+1}, ..., j_{|J|}\} \cup \{l_{|K|-s'_K(x^3)+1}, ..., k_{|K|}\}$ *is not a cover.*

(d_2) If $s'_L(x^3) = |L|-1$, then the set $\{l_1, ..., l_{|L|-1}\} \cup \{j_{|J|-s'_J(x^3)+1}, ..., j_{|J|}\} \cup \{k_{|K|-s'_K(x^3)+1}, ..., k_{|K|}\}$ *is not a cover.*

(d_3) If $s'_L(x^3) \geq 2$ and $s'_L(x^3) \leq |L|-2$, then the sets $\{l_{|L|-s'_L(x^3)}, ..., l_{|L|-1}\} \cup \{j_{|J|-s'_J(x^3)+1}, ..., j_{|J|}\} \cup \{k_{|K|-s'_K(x^3)+1}, ..., k_{|K|}\}$ *and* $\{l_1, l_{|L|-s'_L(x^3)+2}, l_{|L|-s'_L(x^3)+3}, ..., l_{|L|}\} \cup \{j_{|J|-s'_J(x^3)+1}, ..., j_{|J|}\} \cup \{k_{|K|-s'_K(x^3)+1}, ..., k_{|K|}\}$ *are not covers.*

Proof: Theorem 3.3 showed that the TSI inequalities returned from TSIA are valid. The origin never satisfies a TSI inequality at equality, and so the dimension of any TSI inequality is at most $|J|+|K|+|L|-1$ in $P^{ch}_{KP\ \ J\cup K\cup L}$. So it remains to be seen that there are $|J|+|K|+|L|$ affinely independent points that meet $TSI^{J,K,L}_{\alpha_J,\alpha_K,\alpha_L}$ at inequality.

This proof begins by finding $|J|$ affinely independent points. Observe that $s'_J(x^1) \neq |J|$ and $s'_J(x^1) \neq 0$ due to assumption (a). Therefore, the proof divides into the following 3 cases based upon values of $s'_J(x^1)$.

If $s'_J(x^1) = 1$, then the set $\{j_1\} \cup \{k_{|K|-s'_K(x^1)+1}, ..., k_{|K|}\} \cup \{l_{|L|-s'_L(x^1)+1}, ..., l_{|L|}\}$ is not a cover. The set J is sorted in descending order of its corresponding coefficients. Therefore, the point $e_{j"} + \sum_{k'=|K|-s'_K(x^1)+1}^{|K|} x_{k'} + \sum_{l'=|L|-s'_L(x^1)+1}^{|L|} e_{l'}$ is feasible for all $j" \in J$. This clearly results in $|J|$ linearly independent points.

If $s'_J(x^1) = |J|-1$, then the set $\{j_1, ..., j_{|J|-1}\}\cup\{k_{|K|-s'_K(x^1)+1}, ..., k_{|K|}\}\cup\{l_{|L|-s'_L(x^1)+1}, ..., l_{|L|}\}$ is not a cover. Again, the set J is sorted in descending order of its corresponding coefficients. Therefore the point $(\sum_{j'=1}^{|J|} e_{j'}) - e_{j"} + \sum_{k'=|K|-s'_K(x^1)+1}^{|K|} x_{k'} + \sum_{l'=|L|-s'_L(x^1)+1}^{|L|} e_{l'}$ is feasible for all $j" \in J$. Since upper left $|J| \times |J|$ rows and columns are a cyclical permutation of $|J|-1$ ones and $|J|$ and $|J|-1$ are relatively prime, these $|J|$ points are linearly independent as required.

The case when $s'_J(x^1) \geq 2$ and $s'_J(x^1) \leq |J|-2$ is merely a combination of the two above cases. Since $\{j_1, j_{|J|-s'_J(x^1)+2}, j_{|J|-s'_J(x^1)+3}, ..., j_{|J|}\}\cup\{k_{|K|-s'_K(x^1)+1}, ..., k_{|K|}\}\cup\{l_{|L|-s'_L(x^1)+1}, ..., l_{|L|}\}$ is not a cover, $e_{j"} + \sum_{j'=|J|-s'_J(x^1)+2}^{|J|} e'_j + \sum_{k'=|K|-s'_K(x^1)+1}^{|K|} x_{k'} + \sum_{l'=|L|-s'_L(x^1)+1}^{|L|} e_{l'}$ is a feasible point for all $j" \in \{j_1, ..., j_{|J|-s'_J(x^1)-1}\}$. For the remaining $s'_J(x^1)+1$ points, observe that $\{j_{|J|-s'_J(x^1)}, ..., j_{|J|-1}\} \cup \{k_{|K|-s'_K(x^1)+1}, ..., k_{|K|}\} \cup \{l_{|L|-s'_L(x^1)+1}, ..., l_{|L|}\}$ is not a cover. So, $\sum_{j'=|J|-s'_J(x^1)}^{|J|} e_{j'} - e_{j"} + \sum_{k'=|K|-s'_K(x^1)+1}^{|K|} x_{k'} + \sum_{l'=|L|-s'_L(x^1)+1}^{|L|} e_{l'}$ is a feasible point for all $j" \in \{j_{|J|-s'_J(x^1)}, ..., j_{|J|}\}$. These points are clearly linearly independent following a combination of the logic of the above two paragraphs.

In all three cases, the $|J|$ linearly independent points can be easily applied for sets K and L. For instance, to get $|K|$ linearly independent points the $s'_J(x^2)$ and $s'_L(x^2)$ variables with the smallest a coefficients would be set to one in J and L, respectively. The points selected for K would be chosen as was the case for the points for J in the above three paragraphs. A similar logic could be followed for L.

There are now $|J| + |K| + |L|$ points that are generated and each of these points meet the TSIA inequality at equality due to the choices of α_J, α_K and α_L. This matrix of points can be analyzed by first partitioning it into the obvious 9 submatrices, based upon J, K and L. Every row in the $J \times K$, $J \times L$, $K \times J$, $K \times L$, $L \times J$ and $L \times K$ submatrices is a constant of either a 0 or a 1. The sum of the first $|J|$ rows, the next $|K|$ rows and the next $|L|$ rows results in the same matrix as the transpose of the s' matrix that TSIA uses. Since this TSIA's matrix is linearly independent, the above matrix is also linearly independent and the result follows.

□

To help understand this theorem, the TSI inequality from Example 3.1 will be reexamined. The affinely independent points shown in Figure 7 clearly show the cases where $s'_J(x^i) = 1$ and where $s'_J(x^i) = |J| - 1$. Observe that the points in the first three columns are clearly affinely independent. All three points are identical below the first three rows, so if the first point is valid, both of the other points are clearly valid. The sum of the coefficients corresponding to the ones in this first point is 113, which is less than 114, so all of these three points are feasible.

Similarly, the last seven points in Figure 7 demonstrate the $s'_J(x^i) = |J| - 1$ case. In this situation, the largest point is the last one, so the last column is evaluated. The sum of the coefficients of all the variables set to one is 114, which is equal to 114. All of the other six points are clearly feasible due to the sorted of order of the a_i's.

Figure 11 provides an example of the third case and shows that it also gives affine independence. Figure 11 shows the matrix of affinely independent points that could be generated from node 11 in Figure 9. Observe that the middle five points in Figure 11 meet the final condition. Two points need to be evaluated to determine whether these points are feasible. The points corresponding to the fourth column and the eigth column need to be evaluated: If they are feasible, the other points in this section are also feasible. The sum of the a_i's from the first point is 106 and so it is clearly feasible. The second point's sum of the a_i's is 105 and is also clearly feasible, so all points in this set are feasible.

$$
\begin{array}{c|ccc|ccccc|ccccccc|}
 & \multicolumn{3}{c|}{J} & \multicolumn{5}{c|}{K} & \multicolumn{7}{c|}{L} \\
\hline
 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
J & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
K & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\
 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
L & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
\end{array}
$$

Figure 3.5: *Matrix of Affinely Independent Points from Example 3.1*

Chapter 4 - Computational Results

In order to show that TSIs can reduce the computing time for knapsack problems, several KPs were solved using CPLEX 10.0 [9] both at default settings and with three set inequalities. These tests were performed on a Pentium IV 1.5 GHz processor with 512 MB of RAM. The time these problems took to solve and the number of nodes evaluated by CPLEX are recorded and compared in this section. Twenty random KPs each were generated with a problem size of 25, 50, 75, and 100 variables.

A problem with computational studies on knapsack instances is the fact that most solvers will find an optimal solution relatively quickly. However, this challenge can be overcome, as shown by Chvátal [8], who provided a class of knapsack instances that requires exponentially many branches to solve when using branch and bound. Later, Hunsaker and Tovey improved upon these findings to show that some KPs require exponentially many branches even with all sequence dependent cover inequalities added to the formulation [19]. Both of these results were found using very large random numbers for all a_i and forcing all $c_i = a_i$ for all $i = 1, ..., n$. Given these conditions, there is a high probability that there exists a solution with the objective value equal to b.

In order to generate difficult KPs, a process based on these results was used. The a_i's are generated using random integers uniformly distributed between 50,000 and 100,000, and all c_i's are set equal to the corresponding a_i's. After all of the a_i's are generated, b is set to half of the sum of the a_i's.

When implementing this algorithm, a specific form of sets are chosen as input. The KPs used all had two minimal covers that shared at least one variable. These overlapping covers were found by taking the variable with the highest a_i and adding variables in descending order of their a_i's until a cover was formed. The size of the overlap was set before solving each set of problem sizes and held constant throughout the twenty iterations for each size. This overlap represented the last number of variables in the first cover, and the first number of variables in the second cover. The second cover was generated by taking the variables in the overlap and adding variables in descending order of the corresponding a_i's until a cover is found. In other words, $C_1 \cup C_2 = \{1, 2, ..., (|C_1| + |C_2| - sizeofoverlap)\}$.

The two overlapping minimal covers, C_1 and C_2, are used to define the three sets as $J = C_1 \setminus C_2,\ K = C_1 \cap C_2,$ and $L = C_2 \setminus C_1$. The initial values for $\alpha_J,\ \alpha_K,$ and α_L are set to be $1/|J|,\ 1/|K|,$ and $1/|L|$, respectively. Because the three sets chosen are overlapping minimal covers, these three α values clearly correspond to feasible points. The three points are shown in the s' form along with the initial values of α in Table 3.

The results of these runs are summarized in Tables 4, 5, and 6. Table 4 gives a comparison of solution times between default CPLEX 10.0 and default CPLEX 10.0 with one TSI inequality. Table 5 compares the sizes of the node trees of these two situations. Table 6 gives the preprocessing time for generating TSI inequalities. All reported values are the average of the twenty problem instances.

Table 4.1: *Initial Points and α Values*

Set	$s'_X(x^1)$	$s'_X(x^2)$	$s'_X(x^3)$	α_X
J	$\|J\|$	0	0	$1/\|J\|$
K	0	$\|K\|$	0	$1/\|K\|$
L	0	0	$\|L\|$	$1/\|L\|$

Table 4 shows the current techniques used to generate TSI inequalities are not substantially beneficial. In only one of the four test instances ($n = 50$) were TSI inequalities beneficial. For the remainder of the instances TSI inequalities didn't substantially improve CPLEX's solution time.

However, Table 5 shows that the size of the branching trees typically decreased by using TSI inequalities. So one may naturally question how could fewer nodes be evaluated and still have a slower running time. The answer is that the basis has gone from one variable to two. Obviously, each iteration of the simplex method could easily take at least twice as long and so the effectiveness of the cutting plane is cancelled out by the longer time to solve each node.

Table 6 shows the single biggest problem of TSI inequalities. Running TSIA can take substantially longer than the time CPLEX requires to solve the original problem. In the case of $n = 100$, the preprocessing time took nearly one and a half hours. In contrast, the solution to the original problems took less than a minute. Twenty instances with $n = 125$

Table 4.2: *Comparison of Run Times with and without TSI Inequalities*

n	$Overlap$	$CPLEX$ 10.0 $Time$	TSI $Time$	% $Difference$
25	5	9.25	16.1	+74.1%
50	5	15	12.45	−17%
75	15	12.2	12.4	+1.6%
100	20	15.75	24.25	+54.0%

Table 4.3: *Comparison of Node Tree Sizes with and without TSI Inequalities*

n	$Overlap$	$CPLEX$ 10.0 $Nodes$	TSI $Nodes$	% $Difference$
25	5	58711.45	89881.5	+53.1%
50	5	61045.2	48344.35	−20.8%
75	15	54866.5	33980.75	−38.1%
100	20	43696.9	50362.4	+15.3%

did not finish in 5 days due to the preprocessing length. Thus, the current technique to generate TSI inequalities takes far too long to be computationally beneficial.

Clearly the choice of sets that was used for this study was not the most sophisticated that could have been developed. Selection of sets and initial points is an area of research that could significantly improve the results of this algorithm.

It is fairly evident that another great opportunity for improvement to TSIA is to replace the IP step with a polynomial time algorithm that is similar to an algorithm developed by Easton and Hooker [11] that can simultaneously lift sets of variables into a cover inequality in linear time. The resulting implementation would remove the integer programming step and even the $z > 1$ loop and replace the entire algorithm with a quadratic or cubic algorithm.

Table 4.4: *TSI Preprocessing Times*

n	*Overlap*	*Preprocessing*
25	5	0.5
50	5	420.2
75	15	1311.7
100	20	2787.9

This would dramatically reduce this enormous preprocessing time.

In conclusion, the preprocessing time required to generate TSI inequalities will make every computational study fall short of its goal of showing the overall time improvement of these inequalities. However, if the preprocessing time is reduced, then a more detailed study should be performed to demonstrate that these inequalities can be extremely useful as the theoretical results dictate.

Chapter 5 - Conclusions

Integer programming is a useful tool for modeling and optimizing real world problems. Unfortunately, the time required to solve IPs is exponential and so large problems often cannot be solved. The knapsack problem is a form of IP that has only one constraint and can be used to strengthen any constraint of a general integer program. These facts make finding new classes of facet-defining inequalities to the knapsack problem an extremely important area of research.

This thesis has introduced three set inequalities and an algorithm for finding them. Theoretical results show that these inequalities will be of dimension at least 2, and can be facet defining under certain conditions. Furthermore, TSIA can generate multiple inequalities for some problems, as seen in the enumerative branching tree provided in Chapter 3.

At this time, a computational study has little value, as the preprocessing time for generating TSI inequalities is prohibitively large. However, three set inequalities can be improved in a number of ways that would make such a study more useful.

5.1 Extensions and Future Research

In the future, three set inequalities could potentially be improved by using a more sophisticated method for selection of which points to replace in a solution. Improvements in this area could greatly improve the preprocessing time, which is a problem with the current formulation. Using an adaptation of the simultaneous lifting process developed by Easton and Sharma [26] could cut out the need to solve an integer program in preprocessing and improve the algorithm to quadratic or cubic effort.

The principles behind three set inequalities could also be applied to find ways to solve IPs with multiple covers overlapping over the same set of variables or over different sets. These q-set inequalities would take the form $QSI^{I,J,K,...,Q}_{\alpha_I,\alpha_J,\alpha_K,...,\alpha_Q} = \alpha_J \sum_{j \in J} x_j + \alpha_K \sum_{k \in K} x_k + \alpha_L \sum_{l \in L} x_l + ... + \alpha_Q \sum_{q \in Q} x_q \leq 1$. Obviously, the enumerative branching tree for this type of inequality would increase in size by a factor of q at each successive level.

One of the most exciting aspects of TSIA is that it could be extended to develop an interior point method for solving IPs. When used in conjuction with branching trees, such an approach could revolutionize the way that IPs are solved. Such an approach could branch on potential constraints rather than on variables, cutting out large non-integer regions of space.

Bibliography

[1] Balas, E., (1975). "Facets of the Knapsack Polytope", *Mathematical Programming*, **8**, 146-164.

[2] Balas, E and E. Zemel (1978). "Facets of the Knapsack Polytope from Minimal Covers," *SIAM Journal of Applied Mathematics*, **34**, 119-148.

[3] Balas, E and E. Zemel (1980). "An Algorithm for Large Zero-One Knapsack Problems," *Operations Research*, **28**, 1130-1154.

[4] Balas, E (2005). "Projection, Lifting, and Extended Formulation in Integer and Combinatorial Optimization," *Annals of Operations Research*, **140**, 125-161.

[5] Balas, E. and E. Zemel, (1984). "Lifting and Complementing Yields all the Facets of Positive Zero-One Programming Polytopes," *Mathematical Programming, Proceedings of the International Conference on Mathematical Programming*, R.W. Cottle et al., eds., 13-24.

[6] Bertsimas, D., C. Darnell and R. Soucy (1999). "Portfolio Construction through Mixed-Integer Programming at Grantham, Mayo, Van Otterloo and Company," *Interfaces*, **29**, n 1, Jan.-Feb. 1999, 49-66.

[7] "Capital Budgeting." (2006). Retrieved April 18, 2007, from http://www.netmba.com/finance/capital/budgeting/

[8] Chvátal, V. (1980). "Hard Knapsack Problem," *Operations Research*, **28(6)**, 1402-1412.

[9] CPLEX's website marketed by ILOG "http://www.ilog.com/products/cplex/".

[10] Easton, K., G. Nemhauser and M. Trick (2003). "Solving the Traveling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach," *Practice and Theory of Automated Timetabling IV. 4th International Conference, PATAT 2002*, Selected Revised Papers (*Lecture Notes in Comput. Sci.* Vol.2740), 2003, p 100-9.

[11] Easton, T. and K. Hooker. "Scaled Multiple Cover Inequalities and the Knapsack Polytope," in review for the Special Issue of *Discrete Optimization* in memory of George B. Dantzig (1914-2005).

[12] Gomory, R. (1969). "Some Polyhedra Related to Combinatorial Problems," *Linear Algebra and its Applications*, **2**, 451-558.

[13] Granmo, O. C., B. J. Oommen, S. A. Myrer, and M. G. Olsen (2007). "Learning Automata-based Solutions to the Nonlinear Fractional Knapsack Problem with Applications to Optimal Resource Allocation," *IEEE Transactions on Systems, Man and Cybernetics, Part B(Cybernetics)*, **37** n 1, 166-175.

[14] Gu, Zonghao, Nemhauser, G. L., and Savelsbergh, M. W. P. (2000). "Sequence Independent Lifting in Mixed Integer Programming," *Journal of Combinatorial Optimization*, **4**, n 1, 109-129.

[15] Gu., Z., Nemhauser, G. L., and M. W. P. Savelsbergh (1998). "Lifted Cover Inequalities for 0-1 Integer Programs: Computation," *Informs Journal on Computing*, **10**, 427-437.

[16] Gutierrez, Maria Talia (2007). “Lifting General Integer Programs,” Kansas State University Masters Thesis.

[17] Hammer, P., E. Johnson, and U. Peled, (1975). “Facets of Regular 0-1 Polytopes,” *Mathematical Programming*, **8**, 179-206.

[18] Hillier, F. S. and G. J. Lieberman, (2001). *Introduction to Operations Research*, McGraw-Hill, New York 576-581.

[19] Hunsaker, B. and C. Tovey (2004). “Simple Lifted Cover Inequalities and Hard Knapsack Problems,” *Technical Report: Industrial Engineering, University of Pittsburgh*, Pittsburgh, PA 1-13.

[20] Huschka, Bryce (2007). ”Finding Adjacent Facet-Defining Inequalities,” Kansas State University Masters Thesis.

[21] Karp, R. M. (1972). ”Reducability Among Combinatorial Problems,” *Complexity of Computer Computations*, Plenum Press, New York 85-103.

[22] Linderoth, Jeff (2003). “Cover Inequalities.” Retrieved April 18, 2007, from Rutgers University Web site: http://dimacs.rutgers.edu/reconnect/Lafayette/lectures/knaplecture.pdf

[23] Nemhauser, G. L. and L. A. Wolsey (1988). *Integer and Combinatorial Optimization*, John Wiley and Sons, New York.

[24] Parker, Brent 2007. "Project Allocation and Anticover Inequalities," Kansas State University Masters Thesis.

[25] Richard, J. P. P., I. R. de Farias, Jr., and G. L. Nemhauser (2002). "Lifted Inequalities for 0-1 Mixed Integer Programming: Basic Theory and Algorithms," *9th International Integer Programming and Combinatorial Optimization Conference Proceedings*, 161-175

[26] Sharma, Kamana (2007). "Simultaneously Lifting Sets of Variables in Binary Knapsack Problems," Kansas State University Masters Thesis.

[27] Subbu, R., G. Russo, K. Chalermkraivuth, and J. Celaya (2007). "Multi-criteria Set Partitioning for Portfolio Management: a Visual Interactive Method," *2007 First IEEE Symposium on Computational Intelligence in Multicriteria Decision Making.*" 6.

[28] Taylor, P. E. and S. J. Huxley, (1989). "A Break from Tradition for the San Francisco Police: Patrol Officer Scheduling Using an Optimization-Based Decision Support System," *Interfaces*, **19**(1), 4-24.

[29] Tomastik, R.N. (1993). "The Facet Ascending Algorithm for Integer Programming Problems," *Proceedings on the 32nd IEEE Conference on Decision and Control, 1993*, **3**, 2880-2884.

[30] Wolsey, L.A. (1975). "Faces for a Linear Inequality in 0-1 Variables," *Mathematical Programming*, **8**, 165-178.