SYNCHRONIZED SIMULTANEOUS LIFTING IN BINARY KNAPSACK POLYHEDRA

by

JENNIFER ELAINE BOLTON

B.S., Kansas State University, 2009

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

2009

Approved by:

Major Professor

Dr. Todd Easton

ABSTRACT

Integer programs (IP) are used in companies and organizations across the world to reach financial and time-related goals most often through optimal resource allocation and scheduling. Unfortunately, integer programs are computationally difficult to solve and in some cases the optimal solutions are unknown even with today's advanced computing machines.

Lifting is a technique that is often used to decrease the time required to solve an IP to optimality. Lifting begins with a valid inequality and strengthens it by changing the coefficients of variables in the inequality. Often times, this technique can result in facet defining inequalities, which are the theoretically strongest inequalities.

This thesis introduces a new type of lifting called synchronized simultaneous lifting (SSL). SSL allows for multiple sets of simultaneously lifted variables to be simultaneously lifted which generates a new class of inequalities that previously would have required an oracle to be found. Additionally, this thesis describes an algorithm to perform synchronized simultaneous lifting for a binary knapsack inequality called the Synchronized Simultaneous Lifting Algorithm (SSLA). SSLA is a quadratic time algorithm that will exactly simultaneously lift two sets of simultaneously lifted variables.

Short computational studies show SSLA can sometimes solve IPs to optimality that CPLEX, an advanced integer programming solver, alone cannot solve. Specifically, the SSL cuts allowed a 76 percent improvement over CPLEX alone.

Dedication

My work is dedicated to my parents, John and Susan, and my siblings, Amanda and Thomas, for their unconditional love and continuing support.

Acknowledgments

I would like to first and foremost acknowledge Dr. Todd Easton. Without his patience and technical knowledge, this research would not have been possible. Additionally, I would like to recognize the efforts of Dr. John Wu and Dr. John Boyer in their work on my review committee. Lastly, my sincere thanks go to the faculty, staff, and students of the Kansas State University Department of Industrial and Manufacturing Systems Engineering for their support of my educational and professional endeavors.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Integer programs are widely used in industries across the world because of their usefulness in a large variety of business applications and their ability to find the optimal solution to a problem. The traditional application of an integer program (IP) seeks to maximize profits while not exceeding a company's constraints (usually resource limitations or demand constraints).

The value of integer programming is best explained through the description of ittyts applications. One example of an integer program used in industry is in the scheduling of asset divestitures [7]. Homart Development Company is ranked among the largest commercial land developers in the United States. At any given time, Homart has over 100 properties under evaluation for divesture. The company used a binary integer program to determine the optimal time to divest the properties. It is estimated that this binary integer programming technique added 40 million dollars in profit to the divesture plan. Another example of the application of an integer program in industry is from the Reynolds Metals Company. This company used a binary integer program to select automatically dispatched delivery vehicles which saved over 7 million dollars annually [26].

Integer Programs have similarly been used in a wide variety of applications including airline scheduling [1, 33], sports scheduling [11, 35], portfolio management [8, 29], genetic research [10, 13], cancer research [24, 25] and in transporting goods [2, 20, 30, 34].

Of particular interest in this thesis is a special type of integer program called a knapsack problem. This type of problem is best described by considering a hiker packing her knapsack for a trip. The hiker has n potential items that she can pack and she assigns a benefit to each item that may be packed. Each item is also assigned a weight. She seeks to maximize her total benefit of the items in her knapsack, but is limited by the total weight she can carry.

The knapsack problem is important in its own right. One of the most common uses for the knapsack problem is in resource allocation to maximize the benefit obtained when using limited resources [9]. Another common application is in the scheduling of jobs on a machine that are precedence constrained [21]. Often times the goal in this application is to minimize the sum of weighted completion times.

Unfortunately, integer programs are $\mathcal{NP}$-hard in general [19], which means that finding the optimal solution to an IP requires exponential time, unless $\mathcal{P} = \mathcal{NP}$. In the former case, concessions must be made for the solution to be found more quickly.

This is not ideal because often, the solution that this yields is suboptimal and must be evaluated to determine whether it is still implementable and whether it provides a benefit to the business.

Exhaustive enumeration is the simplest way to solve an IP. However, this technique is very time consuming and therefore very costly. Additionally, exhaustive enumeration becomes computationally impossible even for small problems. It is important to note that if there are n binary variables, exhaustively enumerating all the solutions will lead to 2^n evaluations. Therefore, it is imperative to ensure that exhaustive enumeration is not used to solve large problems.

Several strategies have been developed to reduce the time and storage required to exhaustively enumerate all solutions. One type of strategy, called branch and bound [23], is a branching strategy that eliminates a branch of the solution tree when it becomes evident that the optimal solution cannot come from that line of solutions. In the worst case, branch and bound can exhaustively enumerate all of the solutions.

Another technique commonly used to solve IPs involves cutting planes. To implement this, the integer constraint is relaxed and the problem is solved as if it is a linear program (in which the optimal solution is called the linear relaxation). Cutting planes are used to eliminate areas of the linear relaxation, while not eliminating any feasible integer solutions. The strongest kind of inequalities (or cutting planes) are said to be facet defining.

One of the most common techniques to create valid and useful cutting planes is through a technique called lifting. The basic idea of this technique is to take a valid inequality and make it stronger by modifying the coefficient of a set of variables. Lifted inequalities take the following form: $\sum_{i\in E}\alpha_i x_i+\sum_{i\in N/E}\alpha_i x_i \leq \beta$. While lifting was first developed by Gomory [14], it has been a major area of interest in integer programming and hence has attracted a significant amount of research for the past forty years.

The most basic type of lifting is called sequential lifting and involves lifting one variable at a time into the inequality. This results in each variable having a different lifting coefficient [4]. Unfortunately, this type of lifting is sequence dependent meaning that the order matters when the variables are lifted into the inequality.

Another significant type of lifting is called simultaneous lifting, which was developed by Zemel in 1978 [40]. Simultaneous lifting lifts multiple variables into an inequality with one alpha value. One advantage of this type of lifting is that order does not matter since all variables are lifted at once.

The ultimate goal of lifting is to create stronger inequalities. This thesis is based on simultaneous lifting and introduces a new type of lifting called synchronized simultaneous lifting (SSL). SSL can yield a new class of theoretically strong inequalities some of which are facet defining.

1.1 Motivation and Research Contributions

There have been many recent advancements in the area of lifting. These advancements are mainly concentrated in new inequalities that generate better inequalities in a shorter period of time. In 2005, Hooker and Easton [12] investigated the link between graph theory and cutting planes in integer programming to derive a new method to simultaneously lift in variables to knapsack problems. Then in 2007, Gutierrez [17] developed a technique that allows the practitioner to simultaneously lift multiple general integer variables in order to generate a new class of inequalities. However, this method requires solving an integer program. At the same time, Sharma [31] developed a lifting technique for knapsack problems that did not require solving any integer programs and generates many inequalities. Unfortunately, Sharma's technique only works for knapsack problems. Shortly thereafter, Kubik [22] developed a psuedo-polynomial time algorithm and a polynomial time algorithm for knapsack problems to lift multiple sets into the knapsack constraint.

By synthesizing the concepts of these researchers, this thesis presents a new lifting method called synchronized simultaneous lifting (SSL). SSL allows for the simultaneous lifting of two sets of simultaneously lifted variables in any bounded integer programs. Since the algorithm allows for two lifting coefficients for sets of simultaneously lifted variables, some of the resulting inequalities could not have been created by any previous lifting methods without the consultation of an oracle (fortune-teller). Additionally, these inequalities are sometimes facet defining.

A new algorithm, called the Synchronized Simultaneous Lifting Algorithm (SSLA) is introduced. SSLA finds new SSL inequalities by solving simple algebraic equations and produces the lifting coefficients for the sets of simultaneously lifted variables in a binary knapsack constraint. Additionally, SSLA runs in quadratic time. One of the main advantages of this algorithm is that neither of the beginning sets needs to be a cover.

A small computational study has shown that in approximately half of the problems considered, SSLA enabled CPLEX, an advanced computational software package, to solve problems that the package otherwise would not have been able to solve due to lack of sufficient memory. The study was conducted on problems of 50 and 100 variables and shows that SSLA is helpful in solving integer programs.

In summary, this thesis formally introduces SSL and SSLA to the world.

1.2 Outline

The remainder of this thesis is organized as follows: Chapter two outlines the basic concepts required to understand the advances that this research provides. These concepts are concentrated in the areas of integer programming and polyhedral theory and include topics such as convexity, dimension, affine independence, cutting planes, facet defining inequalities, the knapsack problem, covers, sequential lifting and simultaneous lifting.

The third chapter of this thesis describes the theory behind and the implementation of synchronized simultaneous lifting. It outlines the implementation of the SSLA using an example where SSLA yields facet defining inequalities. Theorems and proofs pertinent to the SSLA algorithm are also included in this chapter along with a step by step determination of the running time.

Chapter four of this thesis shows how SSLA is applied in a computational study and the results of these trials. The results of the trials are interpreted and it is shown that the SSLA is easy to implement and the resulting inequalities can be useful.

Finally, the fifth chapter illustrates how this research contributes to the area of IP. It summarizes the research presented in this thesis and suggests how future research might build upon the foundation provided in this thesis.

Chapter 2

Background Information

The following sections in this chapter provide the necessary definitions and concepts required for the basic understanding of this thesis. The majority of the required understanding lies within the realm of polyhedral theory. These concepts include linear relaxations, convexity, polyhedrons, dimension, half-spaces, and affine independence. These concepts are required to understand the concepts of cutting planes, facet defining inequalities, and lifting, which are the focus of this research.

In order to understand the advancement of this research, it is first important to understand the fundamentals of integer programming and polyhedral theory. An integer program contains a linear objective function that seeks to either minimize or maximize the outcome of the function subject to a set of linear constraints. Additionally, as the name suggests, the solution space of the function is limited to integers. Technically, an integer program is defined as follows:

$$Z^{IP} = \text{Maximize } c^T x$$

$$\text{subject to } Ax \leq b$$

$$x \in \mathbf{Z}_+^n$$

where $c \in \mathbf{R}^n$ represents the objective coefficients, $A \in \mathbf{R}^{m \times n}$ and $b \in \mathbf{R}^{m \times 1}$ represent constraints. The feasible points of an integer program are denoted by P and are defined as $P = \{x \in \mathbf{Z}_+^n : Ax \leq b\}$. Here the set of indices of the x variables is denoted as $N = 1, ..., n$.

Unfortunately, it is very difficult to find P as finding the optimal element of P is $\mathcal{NP}$-hard. The standard technique is to relax the integer constraint and use linear programming many times to find the optimal solution. When the integer constraint is relaxed, the resulting problem is called the linear relaxation of the integer problem. The linear relaxation is defined as follows:

$$Z^{LR} = \text{ Maximize } c^T x$$

$$\text{subject to } Ax \leq b$$

$$x \in \mathbf{R}_+^n$$

It is important to understand that the integer constraint limits the solution to the problem such that $Z^{IP} \leq Z^{LP}$. The set of feasible solutions that results from the linear relaxation, P^{LR}, is formally represented as $P^{LR} = \{x \in \mathbf{R}_+^n : Ax \leq b\}$.

As stated previously, integer programs are extremely difficult to solve because the traditional simplex method will not solve them to optimality. This causes many costly and difficult issues because there are a vast number of types of problems that involve a great deal of theory. An excellent resource on integer programming is Nemhauser and Wolsey [27].

2.1 Polyhedral Theory

Critical to the idea of polyhedral theory is the concept of convexity. A set $S \subseteq \mathbf{R}^n$ is a convex set if, and only if, all the points on a straight line connecting any two points contained in the set are also in the set. This is represented formally as follows: S is convex if, and only if, $x \in S$ where $x = \sum_{i=1}^{n} \lambda_i x^i$ for every finite set of points $\{x^i : i = 1, ..., n\}$ in S and every $\lambda \in \mathbf{R}_+^n$ with $\sum_{i=1}^{n} \lambda_i = 1$. Given a set $S \subseteq \mathbf{R}^n$ the intersection of all of the convex sets that contain the set S is called the convex hull of S and is denoted by S^{ch}.

The feasible region of one linear inequality is called a half-space. Formally, a half space is defined as $\{x \in \mathbf{R}^n : \sum_{i=1}^{n} a_i x_i \leq b\}$. An intersection of finitely many half spaces is called a polyhedron. Clearly, P^{LR} is a polyhedron.

The relationship between polyhedral theory and integer programs can be confusing because integer programs actually have two important polyhedra. The convex hull of the integer solution space is one type of polyhedron, P^{ch}. Formally, $P^{ch} = conv(\{x \in \mathbf{Z}_+^n : Ax \leq b\})$. The other polyhedron, P^{LR}, is found by relaxing the integer constraint of an integer program as previously defined. The goal of integer programming research with respect to polyhedral theory is to change P^{LR} to P^{ch} by adding cutting planes, which is the topic of the next section.

When making the transition from P^{LR} to P^{ch}, dimension is critical. The dimension of a convex hull is typically expressed by the number of linearly independent vectors that are contained within that polyhedron. Alternately, one can find the dimension of a convex hull using affine independence. The points $x^1, ..., x^q$ in R^n are affinely independent if, and only if, the unique solution to $\sum_{i=1}^{q} \lambda_i x^i = 0$ and $\sum_{i=0}^{q} \lambda_i = 0$ is $\lambda_i = 0$ for all $i = 1, ..., q$. The dimension of a polyhedron is equal to the maximum number of affinely independent points minus one.

2.2 Cutting Planes

The following section introduces the concepts of valid inequalities, cutting planes, faces, facet defining inequalities and facets. These concepts are fundamentally important to this research because they form the foundation on which this research is based.

An inequality $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ is valid for an integer program, if every $x \in P$ satisfies the inequality. Valid inequalities are also called cutting planes. One goal of a cutting plane is to eliminate portions of P^{LR}. Essentially, cutting planes eliminate portions of the linear relaxation solution space without eliminating any feasible integer points. Formally, an inequality $\Sigma_{j=1}^{n} \alpha_j x_j \leq \beta$ is valid for P^{ch} if, and only if, $\Sigma_{j=1}^{n} \alpha_j x'_j \leq \beta$ is satisfied for every $x' \in P$.

Cutting planes are especially useful if they contain at least one integer point that meets the inequality at equality because it can be said that the cutting plane is a proper face of P^{ch}. Formally, every valid inequality $\Sigma_{j=1}^{n} \alpha_j x_j \leq \beta$ defines a face $F \subseteq P^{ch}$ that takes the form $F = \{x \in P^{ch} : \Sigma_{j=1}^{n} \alpha_j x_j = \beta\}$. If $F \neq \emptyset$, then F supports P^{ch}.

The most useful cutting planes are called facet defining inequalities, or facets. Facets have a dimension of one less than the dimension of P^{ch}. In other words, a facet of P^{ch} has dimension one less than the dimension of P^{ch}. Facet defining inequalities are important in the realm of polyhedral theory research because they are necessary and sufficient in describing P^{ch}. This means that any other inequality that is applied to P^{LR} will be redundant to a facet defining inequality in describing P^{ch}. Including all facet defining cutting planes result in integer extreme points, which eliminates the need for techniques like branch and bound.

2.3 The Binary Knapsack Problem

This research is applied to a specialized type of integer program, called the binary knapsack problem. The problem is named for the famed example of packing a knapsack for a hiking trip. The hiker has n potential items that can either be taken or left behind. Each item has some sort of benefit, c_i, and is assigned a nonnegative weight, a_i. The hiker seeks to maximize the overall benefit of the items packed while staying under the maximum weight she can carry.

This problem can be solved by modeling it as an integer program. Let $x_i = 1$ if the hiker chooses to take item i, and $x_i = 0$ if not. The IP formulation of the binary knapsack problem is as follows:

Maximize $\sum_{i=1}^{n} c_i x_i$

subject to $\sum_{i=1}^{n} a_i x_i \leq b$

$x \in \mathbf{B}\ forall\ i = 1, 2, ..., n$

This thesis is particularly concerned about the feasible region of a knapsack problem. Let PKP be used to represent the set of feasible solutions, $PKP = \{x \in \mathbf{B}^n : \Sigma_{j=1}^{n} a_j x_j \leq b\}$ with the the convex hull (and associated polyhedron) represented by PKP^{ch}.

Example 2.3.1 A hiker considers taking 19 items on a hiking trip. The hiker has assigned each item with a benefit value and a weight. The benefits and weights are listed in Table 2.1. The weights are listed in 100 gram units and the hiker can carry 14.5 kg (or 145 100-gram units).

Object	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Benefit	15	35	14	31	11	18	18	21	10	16	13	6	12	10	7	9	15	8	15
Weight	34	34	33	33	32	28	15	15	15	14	14	13	13	13	13	12	12	11	9

Table 2.1: Benefit and Weight of Items

The hiker can either choose to take the item ($x_j = 1$) or not take the item ($x_j = 0$). The hiker is seeking to maximize the benefit of the sum of the items in her pack while staying within the amount of weight that she can carry. This problem can be modeled as follows:

Maximize $15x_1 + 35x_2 + 14x_3 + 31x_4 + 11x_5 + 18x_6 + 18x_7 + 21x_8 + 10x_9 + 16x_{10}$

$$+13x_{11} + 6x_{12} + 12x_{13} + 10x_{14} + 7x_{15} + 9x_{16} + 15x_{17} + 8x_{18} + 15x_{19}$$

$$\begin{aligned} \text{subject to} \quad & 34x_1 + 34x_2 + 33x_3 + 33x_4 + 32x_5 + 28x_6 + 15x_7 + 15x_8 + 15x_9 + 14x_{10} + 14x_{11} \\ & +13x_{12} + 13x_{13} + 13x_{14} + 13x_{15} + 12x_{16} + 12x_{17} + 11x_{18} + 9x_{19} \leq 145 \\ & x_j \in \{0,1\}, j \in \{1, ..., 19\}. \end{aligned}$$

Solving this KP shows that the hiker achieves a maximum benefit of 163 by taking items $2, 4, 7, 8, 10, 13, 17,$ and 19. The hiker should carry a total weight of 14.5 kg.

2.4 Covers

Covers are of particular importance in the area of polyhedral theory research with respect to PKP^{ch}. A cover is a set of variable indices from a binary knapsack constraint such that setting all x_j in the set equal to one is infeasible (or larger than the maximum allowed by the constraint). Formally, this is represented as follows: $C \subseteq N$ such that $\Sigma_{j \in C} a_j > b$. A minimal cover is a cover such that when one indice is removed from the set, the set is no longer a cover. In other words, $\Sigma_{j \in C \setminus \{k\}} a_j \leq b$ for each $k \in C$.

Each cover (whether minimal or not) defines a cover inequality. Cover inequalities are valid and take the form $\Sigma_{j \in C} x_j \leq |C| - 1$. This is true because the sum of all of the coefficients $\Sigma_{j \in C} a_j$ is greater than the maximum amount allowed by the constraint, thus at least one variable must be set to zero for every feasible solution. Cover inequalities are useful in the realm of polyhedral theory research because they are easy to find and are usually a good starting place for lifting. Additionally, they are usually faces which also impacts their usefulness in lifting.

Example 2.4.1 Consider again the binary knapsack constraint from Example 2.3.1.

$$34x_1+34x_2+33x_3+33x_4+32x_5+28x_6+15x_7+15x_8+15x_9+14x_{10}+14x_{11}+13x_{12}$$
$$+13x_{13}+13x_{14}+13x_{15}+12x_{16}+12x_{17}+11x_{18}+9x_{19} \leq 145$$

In this case, an example of a cover would be $C = \{7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,18, 19\}$ because $15 + 15 + 15 + 14 + 14 + 13 + 13 + 13 + 13 + 12 + 12 + 11 + 9 = 169 \geq 145$. Additionally, a minimal cover would be $C = \{8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}$ because $15 + 15 + 14 + 14 + 13 + 13 + 13 + 13 + 12 + 12 + 11 + 9 = 154 \geq 145$. Notice that when any of the variables are removed from C, the set becomes feasible and is no longer a cover. Therefore, the cover inequality generated by the minimal cover $C = \{8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}$ would be $x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} \leq 11$.

2.5 Lifting

Lifting seeks to create a stronger cutting plane from a starting valid inequality. In formal terms, suppose $\Sigma_{i\in E}\alpha_i x_i + \Sigma_{i\in N\setminus E}\alpha_i x_i \leq \beta$ is a valid inequality in P^{ch} where $E \subset N$. The goal of lifting is to create a valid inequality of P^{ch} that takes the form $\Sigma_{i\in E}\alpha'_i x_i + \Sigma_{i\in N\setminus E}\alpha_i x_i \leq \beta'$.

Lifting was first developed by Gomory [15] and is a technique commonly used to strengthen inequalities by increasing the dimension of the cutting plane. This is done by changing the coefficient, α, for the variables in the inequality. There are many different kinds of lifting that have been developed by many different researchers, [4, 5, 18, 28, 36, 37, 40]. The various types of lifting are exact or approximate lifting, sequential or simultaneous lifting, and up lifting or down lifting.

2.5.1 Up vs. Down Lifting

The concept of a restricted space is fundamental to understand lifting. The basic idea of a restricted space is to examine a polyhedron where a subset of the variables, say E, are forced to a fixed value, k_i for each $i \in E$. Formally, let $P^{ch}_{E,K} = conv\{x \in P : x_i = k_i \text{ for all } i \in E\}$ where $k_i \in \mathbf{Z}$ and $K = (k_1, k_2, ..., k_{|E|})$.

Up lifting is the most common lifting technique and assumes $K = (0, 0, ..., 0)$. This thesis focuses on uplifting, so for simplicity's sake, $P^{ch}_{E,K}$ is referred to as P^{ch}_{E} if $K = (0, 0, ..., 0)$. Down lifting assumes that all of the k_i's are set to the upper bound of x_i for all $i \in E$. To date, down lifting has only been done sequentially.

2.5.2 Exact vs. Approximate Lifting

Exact lifting provides the strongest inequality possible given a starting valid inequality. However, this method demands that the strongest α' possible be used to lift each variable. This means that if the right hand side of the inequality is decreased, or the α' coefficients on the left side are increased, the inequality is no longer valid. Additionally, exact lifting often requires the solution to an integer program which can become very computationally intensive. However, some researchers have still developed polynomial time techniques to find exact lifting coefficients [6, 12, 31].

The computational intensity required by exact lifting makes the idea of approximate lifting sound very attractive. While the inequalities generated using approximate lifting techniques can often be improved, the advantage of approximate lifting is the time required to generate a valid inequality. Some types of approximate lifting involve sequential lifting [4] and simultaneous lifting [3, 16, 32, 38].

2.5.3 Sequential vs. Simultaneous Lifting

Two major types of lifting are sequential and simultaneous lifting. The main difference between these two types of lifting is the size of E. In sequential lifting, $|E| = 1$ meaning the variables are lifted one at a time. Simultaneous lifting lifts more than one variable at a time, so $|E| \geq 2$.

Sequential Lifting

Sequential lifting changes the α values of the variables one at a time. This process is typically done interatively until all of the possible variables are lifted into the inequality. The sequential up lifting algorithm assumes that $\Sigma_{j=2}^{n}\alpha_j x_j \leq \beta$ is valid for $P_{\{1\}}^{ch}$, and seeks to create a valid inequality $\alpha_1 x_1 + \Sigma_{j=2}^{n}\alpha_j x_j \leq \beta$ for P^{ch}. The methodology to perform sequential uplifting was developed by Wolsey [36].

The methodology to perform sequential up lifting of binary variables seeks to solve the following integer program:

$z^* = \text{Maximize} \quad \Sigma_{j=2}^{n}\alpha_j x_j$

$$\begin{aligned} \text{Subject to} \quad & Ax \leq b \\ & x_1 = 1 \\ & x_1 \in \{0,1\},\ x_2, ..., x_n \in \mathbf{Z}^n \end{aligned}$$

Once the optimal solution is found, $\alpha_1 = \beta - z^*$. It is important to note that Wolsey showed that each sequentially lifted variable increases the dimension of the inequality in P_E^{ch}.

Example 2.5.1 Reconsider the knapsack polytope from Example 2.3.1. Observe that $C = \{8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}$ is a cover. So the cover inequality is $x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} \leq 11$. To sequentially up lift x_7, solve the following IP:

$$z^* = \text{Maximize} \quad x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19}$$

$$\begin{aligned} \text{subject to} \quad & 34x_1 + 34x_2 + 33x_3 + 33x_4 + 32x_5 + 28x_6 + 15x_7 + 15x_8 + 15x_9 + 14x_{10} \\ & +14x_{11}+13x_{12}+13x_{13}+13x_{14}+13x_{15}+12x_{16}+12x_{17}+11x_{18}+9x_{19} \leq 145 \\ & x_7 = 1 \\ & x_i \in \{0,1\},\ i = 1, ..., 19 \end{aligned}$$

The solution z^* to the above integer program is $z^* = 10$. This means that $\alpha_7 = \beta - z^*$, or $\alpha_7 = 11 - 10 = 1$. The resulting valid inequality is $x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} \leq 11$.

To lift x_6, solve $z^* = \text{Maximize}\ \ x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19}$ subject to $34x_1 + 33x_2 + ... + 9x_{19} \leq 145$ and $x_6 = 1$. The solution $z^* = 9$, which means that $\alpha_6 = \beta - z^* = 11 - 9 = 2$. The resulting valid inequality is

$2x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} \leq 11$.

Next, solve $z^* =$ Maximize $2x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19}$ subject to $34x_1 + 33x_2 + ... + 9x_{19} \leq 145$ and $x_5 = 1$ results in $z^* = 9$. So $\alpha_5 = \beta - z^* = 11 - 9 = 2$. The resulting valid inequality is $2x_5 + 2x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} \leq 11$.

When the process is repeated, for x_4, x_3, x_2, and x_1, the final sequentially lifted inequality is $2x_1 + 2x_2 + 2x_3 + 2x_4 + 2x_5 + 2x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} \leq 11$.

The following points show that this inequality is facet defining:

$$\begin{array}{ccccccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0
\end{array}$$

Figure 2.1: Affinely Independent Points to show the sequentially lifted inequality is facet defining

It is important to note that the order in which the variables are sequentially lifted determines the inequality. In this example, there are 7! different orders that can yield 7! different inequalities. In this case, the same inequality would have resulted from lifting the variables in a different order, however that is just a coincidence. Most times, a different lifting order will result in a different inequality, but frequently there are numerous repeated inequalities.

Simultaneous Lifting

Simultaneous lifting allows for more than one variable to be lifted into the inequality at once. This type of lifting finds α values for the entire set that is lifted at one time. In 1978, Zemel [39] developed the first exact method to simultaneously lift multiple variables. Unfortunately, this method requires solving exponentially many integer programs and can only be applied to binary knapsack problems. Zemel's method generates many inequalities by finding the extreme point of the polar, but is too computationally intensive to be efficiently implemented.

Hooker and Easton [12] improved on Zemel's method by developing a linear time algorithm to simultaneously lift variables into cover inequalities for a PKP^{ch}. This is done by defining a set C as a minimal cover. The remaining variables to be lifted are defined as set E. The basic idea is to use the knapsack structure to determine feasible points and then to choose an α value that meets the "most extreme" point at equality.

In 2009, Kubik expanded on this theory by creating a pseudo-polynomial time algorithm that allows multiple simultaneously lifted sets to be sequentially lifted into a valid inequality for PKP^{ch}. This thesis focuses on a method to simultaneously lift simultaneously lifted sets of variables by determining two αs for two sets of lifting coefficients.

Chapter 3

Synchronized Simultaneous Lifting

This chapter presents the major advancements of this research. The largest advancement is the introduction of a new type of lifting, called synchronized simultaneous lifting (SSL). SSL finds a new class of lifted inequalities that could not previously be discovered using any previous lifting methods. A quadratic algorithm is presented that finds numerous SSL inequalities in the knapsack polyhedron. Some theoretical properties describe when SSL inequalities are facet defining.

3.1 Synchronized Simultaneous Lifting

In any of the eight lifting processes mentioned in Chapter 2, the input to the lifting process is an integer programming polyhedron, a valid inequality and a lifting set. In contrast, SSL does not require a valid inequality as input. Instead, the input to SSL are two sets of mutually exclusive variables to be lifted. Thus, the set of variables are $\{x_i : i \in E\}$ and $\{x_i : i \in F\}$ where $E, F \subset N$ and $E \cap F = \emptyset$. The resulting inequalities take the form $\sum_{i\in E} \alpha_i x_i + \sum_{i\in F} \alpha_i x_i \leq 1$ or $\sum_{i\in E} \alpha_i x_i + \sum_{i\in F} \alpha_i x_i \leq 0$. These inequalities are sometimes facet defining.

To find SSL inequalities in a general binary polyhedron, P^{ch}, the following procedure, called Synchronized Simultaneous Lifting Procedure (SSLP), should be followed. Unfortunately, this procedure requires solving many integer programs. Let E and F be the input sets of the synchronized simultaneously lifted variables. For $p = 0, ..., |E|$ solve

$$\begin{aligned} z_p = \text{Maximize} \quad & \Sigma_{j\in F} x_j \\ \text{Subject to} \quad & Ax \leq b \\ & \textstyle\sum_{j\in E} x_j = p \\ & x \in \{0,1\}^n \end{aligned}$$

and

$$\begin{aligned} z'_p = \text{Minimize} \quad & \Sigma_{j\in F} x_j \\ \text{Subject to} \quad & Ax \leq b \\ & \textstyle\sum_{j\in E} x_j = p \\ & x \in \{0,1\}^n. \end{aligned}$$

Find the convex hull of the points given by (p, z_p) and (p, z'_p) for $p = 0, ..., |E|$ ignore any instances of p where the integer program was infeasible. The inequalities that define this convex hull are the synchronized simultaneously lifted inequalities. For convenience these inequalities are scaled to a right hand side of one whenever possible.

Theorem 3.1.1. *Given a bounded integer program, then the inequalities generated by SSLP are valid inequalities of P^{ch}.*

Proof: For contradiction, assume that there exists an inequality that is not valid. Thus there exists a point that is not on the halfspace defined by one of the inequalities. However, this contradicts the definition of a convex hull and the result follows.

□

Clearly, solving $2|E|+2$ IPs is impractical, so an algorithm was developed to simplify the process. The algorithm, called the Synchronized Simultaneous Lifting Algorithm (SSLA), can be used on binary knapsack problems to generate many SSL inequalities in quadratic time.

3.2 The Synchronized Simultaneous Lifting Algorithm for the Knapsack Polytope

One of the main advancements this research provides is the development of a quadratic running time algorithm that finds many SSL inequalities that can be facet defining. SSLA provides a significant advantage because it does not require a valid inequality as an input. The idea of SSLA is to decrease the running time to find SSL inequalities on knapsack problems.

The input to SSLA requires a knapsack constraint $\sum_{i \in N} a_i x_i \leq b$ and two lifting sets $E, F \subset N$ such that $E \cap F = \emptyset$. Denote $E = \{e_1, ..., e_{|E|}\}$ and $F = \{f_1, ..., f_{|F|}\}$. By summing the smallest a coefficients in each set the algorithm solves the first set of IPs from section 3.1 that have a maximize objective function. Notice that the minimize problems are always 0 due to a binary knapsack problem and it is uninteresting. SSLA finds the convex hull by taking a simple ratio test across the inequalities. This also ensures that any reported inequalities are valid.

Synchronized Simultaneous Lifting Algorithm (SSLA)

Initialization:

Sort E and F in non ascending order according to their a knapsack coefficients denoted by a_e and a_f, respectively.

Set $mark := 0$.

Apply subroutine $FeasiblePoints$ to create the $feasE[]$, $feasF[]$ and $Numfeaspoints$.

Main Step:

```
$inq := 1$

while $mark < Numfeaspoints - 1$

  $p^* = feasE[mark]$ and $q^* := feasF[mark]$.

  $count := mark + 1$ and $\alpha^* := \infty$.

  while $count < Numfeaspoints - 1$

  $p = feasE[count]$ and $q = feasF[count]$

    $\alpha_E := \frac{q-q^*}{p^*q-q^*p}$ and $\alpha_F := \frac{p^*-p}{p^*q-q^*p}$

    if $\alpha_E \neq 0$, then

      if $\alpha^* \geq \frac{\alpha_F}{\alpha_E}$, then
```

```
            $\alpha^* := \frac{\alpha_F}{\alpha_E}$, $mark := count$, $\alpha^*_E := \alpha_E$ and $\alpha^*_F := \alpha_F$
        else
            $\alpha^* := -1$, $mark := count$, $\alpha^*_E := \alpha_E$ and $\alpha^*_F := \alpha_F$.
        $count := count + 1$.
    end while
    Report $\alpha^{*inq}_E$ and $\alpha^{*inq}_F$ as coefficients of a valid SSL inequality.
    $inq := inq + 1$.
  end while
```

$FeasiblePoints$ **Subroutine**

```
Initialization:
  $sum := 0,\ p := 0,\ q := 0, count := 0$
  while $sum \leq b$
    if $p \leq |E| - 1$, then $sum := sum + a_{e_{|E|-p}}$ and $p := p + 1$.
    else $sum := sum + a_{f_{|F|-q}}$ and $q := q + 1$.
  end while
  if $q = 0$, then $p := p - 1$, $sum := sum - a_{e_{|E|-p}}$.
  if $q = 1$, then $q := q - 1$, $sum := sum - a_{f_{|F|-q}}$.
  if $q \geq 2$, then
    $feasE[count] := p$, $feasF[count] := 0$ and $count := count + 1$.
    $q := q - 1$ and $sum := sum - a_{f_{|F|-q}}$.
Main Step:
  while $p \geq 0$ and $q \leq |F|$
    if $sum > b$, then $sum := sum - a_{e_{|E|-p+1}}$ and $p := p - 1$.
    else
      $feasE[count] := p$, $feasF[count] := q$, $count := count + 1$
      $sum := sum + a_{f_{|F|-q}}$ and $q := q + 1$.
  end while
  $Numfeaspoints := count + 1$
Return $feasE$, $feasF$ and $Numfeaspoints$.
```

In order to determine the running time of SSLA, the algorithm will be evaluated step by step. First consider the initialization steps. To begin, E and F must be sorted in nonascending order which requires $O(|E|log(|E|)) + O(|F|log(|F|))$ effort. Next, the while loop that makes up the majority of the initialization of the $FeasiblePoints$ subroutine requires $O(|E| + |F|)$ effort. The remainder of the steps in the initialization of the $FeasiblePoints$ subroutine require $O(1)$ effort. The while loop that constitutes the main step of the $FeasiblePoints$ subroutine also requires $O(|E|+|F|)$ since each element in E and F is only examined a constant number of times. Thus, the initialization step requires $O(|E|log(|E|)) + O(|F|log(|F|))$ effort.

Next, consider the main step of SSLA. The evaluation of α_E and α_F for each $feasE[]$ and $feasF[]$ requires examining at most the number of feasible points, $|FeasE[]|$ effort. This evaluation can take place up to $|FeasE[]|$ times, so the effort required to complete the main step of SSLA is $O(|FeasE[]|^2)$. Since $|FeasE[]|$ is bounded by $|E| + |F|$, the running time is determined to be $O(max\{|E|^2, |F|^2\})$, which is $O(n^2)$.

When the SSLA is implemented, a set of valid inequalities across PKP^{ch} results. Formally,

Theorem 3.2.1. *Given a PKP instance and let α_E^{*inq} and α_F^{*inq} be returned from SSLA for some inq. Then the inequality $\sum_{i\in E} \alpha_E^{*inq} x_i + \sum_{i\in F} \alpha_F^{*inq} x_i \leq 1$ is a valid inequality for PKP^{ch}.*

Proof: Given a PKP instance, observe that FeasiblePoints Subroutine finds the maximum number of elements in F that can be set to one given that there are p elements in E set to one for all $p = \{0, ..., |E|\}$. Since the steps are identical to [12] algorithm, their proof it is sufficient and is not difficult for the reader to generate.

For contradiction, assume that $\sum_{i\in E} \alpha_E^{*inq} x_i + \sum_{i\in F} \alpha_F^{*inq} x_i \leq 1$ is not a valid inequality for some inq. Therefore, there exists an x' such that $\sum_{i\in E} \alpha_E^{*inq} x'_i + \sum_{i\in F} \alpha_F^{*inq} x'_i > 1$. Define $p = |\{x'_i = 1 : i \in E\}|$ and $q = |\{x'_i = 1 : i \in E\}|$.

Let $mark'$ and $mark''$ be the initial value of $mark$ and the terminating value of $mark$, respectively, for the iteration where SSLA returns α_E^{*inq} and α_F^{*inq}. Define $p^* = feasE[mark']$, $q^* = feasF[mark']$, $p^{**} = feasE[mark'']$ and $q^{**} = feasF[mark'']$.

Clearly, $\alpha_E^{*inq}p^* + \alpha_F^{*inq}q^* = 1$ and $\alpha_E^{*inq}p^{**} + \alpha_F^{*inq}q^{**} = 1$.

The proof now breaks into three cases. Assume $p \geq p^* + 1$. Observe that $\frac{\alpha_E}{\alpha_F}$ is the slope of the line passing through (p^*, q^*) and (p, q) and that $\frac{\alpha_E^{*inq}}{\alpha_F^{*inq}}$ is the slope of the line passing through (p^*, q^*) and (p^{**}, q^{**}). Since (p, q) has $\alpha_E^{*inq}p + \alpha_F^{*inq}q > 1$, the slope of the line passing through at (p, q) must be more extreme than the slope of the line passing through (p^{**}, q^{**}). Thus, $\frac{\alpha_E^{*inq}}{\alpha_F^{*inq}} > \frac{\alpha_E}{\alpha_F}$, which is a contradiction to the algorithm taking the minimum of all such ratios.

Assume $p = p^*$. Then $q \geq q^*+1$, which is a contradiction to either the $FeasiblePoints$ Subroutine finding the maximum number of elements in F or the algorithm being applied correctly on the previous iteration (see previous paragraph).

Assume $p \leq p^* - 1$. Since SSLA takes the minimum ratio of α_E and α_F, the sequence $\frac{\alpha_E^{*1}}{\alpha_F^{*1}}, \frac{\alpha_E^{*2}}{\alpha_F^{*2}}, ..., \frac{\alpha_E^{*r}}{\alpha_F^{*r}}$ is a monotoically increasing sequence where r is the total number of inequalities generated as long as this ratio is defined to be ∞ if $\alpha_F^{*r} = 0$. Observe that this ratio can never be negative due to the PKP structure. Since $p \leq p^* - 1$ and violates this inequality, (p^*, q^*) would not have been selected to be used as $mark$, which is a contradiction to the method the algorithm is applied.

$\square$

Aside from generating a valid inequality, SSLA can also generate facet define inequalities. First, for each inequality generated by SSLA, there are at least two points in $(feasE[], feasF[])$ that meet this inequality at equality. Denote these points as (p'^{inq}, q'^{inq}) and (p''^{inq}, q''^{inq}). Additionally, now denote $E = \{i_1, ..., i_{|E|}\}$ and $F =$

$\{j_1, ..., j_{|F|}\}$ where the indices are sorted in such a fashion that $a_{i_k} \geq a_{i_l}$ and $a_{j_k} \geq a_{j_l}$ for all $l > k$ and $k = 1, ..., \max\{|E|, |F|\}$. For convenience denote $E^p = \{E_{i_{|E|-p+1}}, E_{i_{|E|-p+2}}, ..., E_{i_{|E|}}\}$, in other words E^p contains the indices from E that have the smallest knapsack coeffiencients in E. Similarly, define $F^q = \{F_{j_{|F|-q+1}}, F_{j_{|F|-q+2}}, ..., F_{j_{|F|}}\}$. Finally, let e_i denote the i^{th} identity point $(0, 0, 0, ...0, i, 0, ..., 0)$.

With these defined, then the following theorem provides a condition for these points to be facet defining.

Theorem 3.2.2. *Given a PKP instance, sets $E, F \subset N$ such that $E \cap F = \emptyset$, and values α_E^{*inq} and α_F^{*inq}, then the inequality $\sum_{i \in E} \alpha_E^{*inq} x_i + \sum_{i \in F} \alpha_F^{*inq} x_i \leq 1$ is facet defining over $PKP^{ch}_{N \setminus (E \cup F)}$ if $1 \leq p'^{inq} \leq |E| - 1$, $1 \leq q''^{inq} \leq |F| - 1$ and the following conditions are met. The set $E^{p'^{inq}+1} \setminus \{i_{|E|}\} \cup F^{q'^{inq}}$ is not a cover and if $p'^{inq} \leq |E| - 2$, then the set $E^{p'^{inq}-1} \cup \{i_1\} \cup F^{q'^{inq}}$ is not a cover. Additionally, The set $F^{q''^{inq}+1} \setminus \{j_{|F|}\} \cup E^{p''^{inq}}$ is not a cover and if $q''^{inq} \leq |F| - 2$, then the set $F^{q''^{inq}-1} \cup \{j_1\} \cup E^{p''^{inq}}$ is not a cover.*

Proof: Since the theorem is only concerned with $PKP^{ch}_{N \setminus (E \cup F)}$, it suffices to find $|E| + |F|$ points in $PKP_{N \setminus (E \cup F)}$ that meet $\sum_{i \in E} \alpha_E^{*inq} x_i + \sum_{i \in F} \alpha_F^{*inq} x_i \leq 1$ at equality. Thus, any variable with an index not in E or F is set to 0 and can be ingnored. Observe that setting any p' variables with indices in E and q' variables with indices in F to one or any p'' variables with indices in E and q'' variables with indices in F to one meet this inequality at equality.

Assume the conditions are met. Then, the points $\sum_{i \in E^{p'^{inq}+1}} e_i - e_k + \sum_{j \in F^{q'^{inq}}} e_j$ for each $k \in E^{p'^{inq}+1}$ is a feasible point in PKP due to the set $E^{p'^{inq}+1} \setminus \{i_{|E|}\} \cup F^{q'^{inq}}$

is not a cover and the sorted order of the knapsack. If $p'^{inq} \leq |E| - 2$, then the set $E^{p'^{inq}-1} \cup \{i_1\} \cup F^{q'^{inq}}$ is not a cover. Thus, the points $sum_{i \in E^{p'^{inq}-1}} e_i + e_k + \sum_{j \in F^{q'^{inq}}} e_j$ for each $k \in E \setminus E^{p'^{inq}+1}$. Clearly these $|E|$ points meet the SSL inequality at equality.

The remaining $|F|$ points are found in a similar method, but this time, the variables with indices in E are constant and the vertices in F are permuted. Then the points $\sum_{j \in F^{q''^{inq}+1}} e_j - e_k + \sum_{i \in E^{p''^{inq}}} e_i$ for each $k \in F^{q''^{inq}+1}$ is a feasible point in PKP due to the sorted order of the knapsack and the set $F^{q''^{inq}+1} \setminus \{j_{|F|}\} \cup E^{p''^{inq}}$ is not a cover. If $q''^{inq} \leq |F| - 2$, then the set $F^{q''^{inq}-1} \cup \{j_1\} \cup E^{p''^{inq}}$ is not a cover. Thus, the points $\sum_{j \in F^{q''^{inq}-1}} e_j + e_k + \sum_{i \in E^{p''^{inq}}} e_i$ for each $k \in F \setminus F^{q''^{inq}+1}$. Clearly these meet the SSL inequality at equality.

These $|E|+|F|$ points are affinely independent, because (p', q') and (p'', q'') are affinely independent. Furthermore, the first set of $|E|$ points are linearly independent and the second set of $|F|$ points are linearly independent. Thus, these points are affinely independent and the result follows.

□

It is now straightforward to derive conditions for SSL inequalities to be facet defining. Formally,

Theorem 3.2.3. *Given a PKP instance, sets $E, F \subset N$ such that $E \cap F = \emptyset$, and values α_E^{*inq} and α_F^{*inq}, then the inequality $\sum_{i \in E} \alpha_E^{*inq} x_i + \sum_{i \in F} \alpha_F^{*inq} x_i \leq 1$ is facet defining over PKP^{ch} if the the conditions of Theorem 3.2.2 are met and if $\{k\} \cup E^{p'^{inq}} \cup F^{q'^{inq}}$ or $\{k\} \cup E^{p''^{inq}} \cup F^{q''^{inq}}$ is not a cover where a_k has the largest a coefficient of indices in*

$N \setminus (E \cup F)$.

Proof: To the previous $|E| + |F|$ points given in Theorem 3.2.2 add on the point $e_k + sum_{i \in E^{p' inq}} e_i + \sum_{j \in F^{q' inq}} e_j$ or $e_k + sum_{i \in E^{p'' inq}} e_i + \sum_{j \in F^{q'' inq}} e_j$ depending upon which set is not a cover for each $k \in N \setminus (E \cup F)$. Clearly these points meet the inequality at equality and are affinely independent and the result follows.

$\square$

Example 3.2.4 illustrates how SSLA can be implemented in a binary knapsack problem to generate many facet defining inequalities.

Example 3.2.4 Reconsider the knapsack polytope from Example 2.3.1, which is defined as follows:

$34x_1 + 34x_2 + 33x_3 + 33x_4 + 32x_5 + 28x_6 + 15x_7 + 15x_8 + 15x_9 + 14x_{10} + 14x_{11} + 13x_{12} + 13x_{13} + 13x_{14} + 13x_{15} + 12x_{16} + 12x_{17} + 11x_{18} + 9x_{19} \leq 145$

$x_1, ..., x_{19} \in \{0, 1\}$

For the purpose of this example, arbitrarily set $E = \{1, 2, 3, 4, 5, 6\}$ and $F = \{7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19\}$. As is seen above, the a coefficients have been sorted for each of the sets. The next step in the initialization for the SSLA requires that $mark = 0$. To begin the initialization step for the $FeasiblePoints$ subroutine the following parameters are set: $sum = 0$, $p = 0$, $q = 0$, and $count = 0$. By continuing the initialization of the subroutine, a_6 is selected because $|E| - 0 = 6$ and so $sum = 0 + 28$, $p = 0 + 1$, and q remains at 0. Since $sum = 28 < 145$, the while loop is entered again.

Now a_5 is selected and sum is updated to $28 + 32 = 60$, p is set to 2 and q remains at 0. Next, a_4 is selected and $sum = 60 + 33 = 93$, p is set to 3 and q remains at 0. Continuing the process, a_3 is selected and sum is updated to $93 + 33 = 126$, p is set to 4 and q remains at 0. This process is continued until a_6, a_5, a_4, a_3, and a_2 are selected and $sum = 160$ which is larger than $b = 145$. Since $q = 0$, $sum = 160 - 34 = 126$ and $p = 5 - 1 = 4$. At this point, the initialization of the $FeasiblePoints$ subroutine is completed.

The main step of the $FeasiblePoints$ subroutine begins with $sum = 126$ is less than $b = 145$, so $feasE[0] = p = 4$ and $feasF[0] = q = 0$. Then $count$ is incremented to 1 and sum is set to $126 + 9 = 135$ and $q = 1$. Proceeding, sum is still less than 145, so $feasE[1] = 4$, $feasF[1] = 1$, $count = 2$, $sum = 146 > 145$, and $q = 2$. On the next pass, $sum = 135 + 11 = 146$ is greater than 145, so sum is set to $sum - a_{e_{|E|-p+1}}$ which leads to $sum = 146 - 33 = 113$ and $p = 3$. Next, sum is less than 145, so $feasE[2] = 3$, $feasF[2] = 2$, $count = 3$, $sum = 113$, and $q = 3$. Again sum is less than 145, so $feasE[3] = 3$, $feasF[3] = 3$, $count = 4$, $sum = 125$, and $q = 4$. In another pass through the while loop, $sum = 137$ is still less than 145, so $feasE[4] = 3$, $feasF[4] = 4$, $count = 5$, $sum = 150$, and $q = 5$. Clearly, $sum = 150$ is greater than 145, so $sum = 150 - 33 = 117$ and $p = 2$. On the next pass through the while loop, $sum = 117$ is less than 145, so $feasE[5] = 2$, $feasF[5] = 5$, $count = 6$, $sum = 130$, and $q = 6$. This process is repeated until $p < 0$ or $q > |F|$.

Then, $Numfeaspoints$ is set to $count + 1$. In this case, $Numfeaspoints = 12$. The values listed in Table 3.1 are returned from the $FeasiblePoints$ subroutine.

$count$	$feasE[count]$	$feasF[count]$
0	4	0
1	4	1
2	3	2
3	3	3
4	3	4
5	2	5
6	2	6
7	2	7
8	1	8
9	1	9
10	0	10
11	0	11

Table 3.1: Data Reported by $FeasiblePoints$ Subroutine

These are the potential candidates to be extreme points and are used in the main step of SSLA. A graphical representation of these points is shown in Figure 3.1.

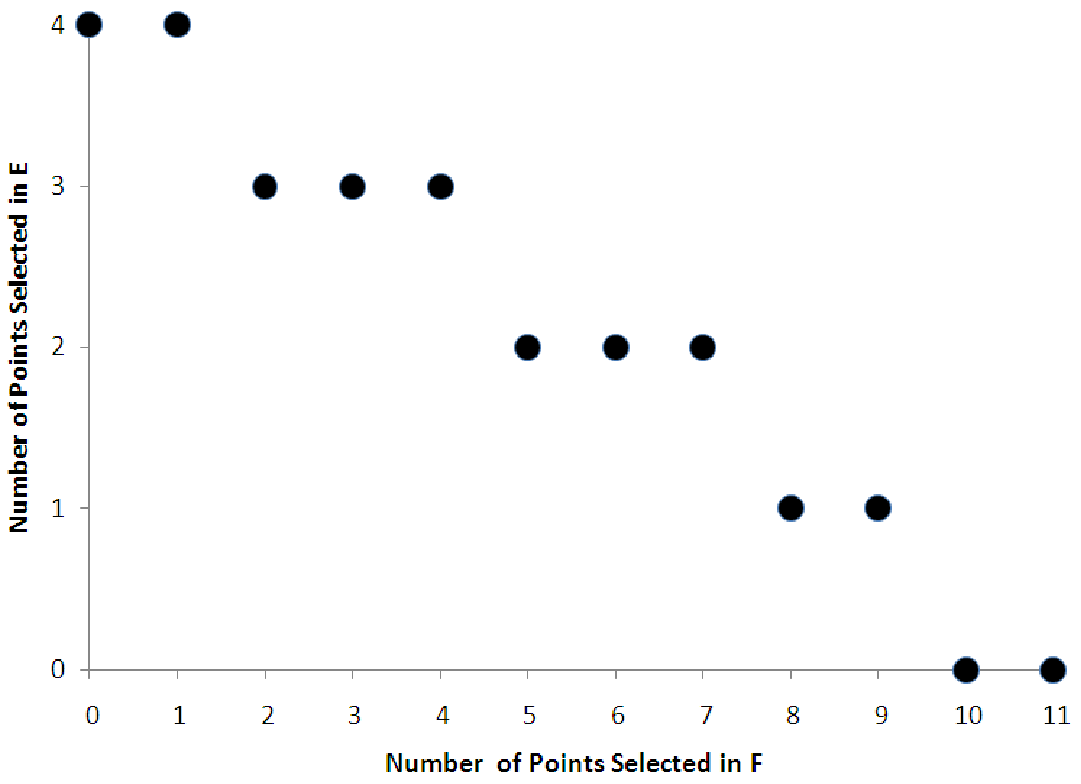

Figure 3.1: Feasible Integer Points Reported from SSLA

From the previous section it is evident that the next step requires finding the convex hull of these points. To begin the main step of the SSLA set $inq = 1$. It follows that $mark = 0$ is less than $Numfeaspoints - 1 = 11$, so $p^* = feasE[0] = 4$ and $q^* = feasF[0] = 0$. Next, $count$ is incremented to 1 and $\alpha^* = \infty$. Since $count = 1$ which is less than $Numfeaspoints - 1 = 11$, $p = feasE[1] = 4$ and $q = feasF[1] = 1$. Using these values, α_E is $\frac{q-q^*}{p*q-q*p} = \frac{1-0}{4*1-0*4} = \frac{1}{4}$ and α_F is $\frac{p^*-p}{p*q-q*p} = \frac{4-4}{4*1-0*4} = 0$. Since $\frac{\alpha_F}{\alpha_E} = 0 < \alpha^*$, α^* is reset to 0. Then $mark = 1$, $\alpha_E^* = \frac{1}{4}$, $\alpha_F^* = 0$, and $count = 2$. Notice that the inequality $\frac{1}{4}\sum_{i \in E} x_i + 0 \sum_{i \in F} x_i \leq 1$ meets the points $(4, 0)$ and $(4, 1)$ at equality.

Since $count = 2$ is still less than $Numfeaspoints - 1$, $p = feasE[2] = 3$, $q =$

$feasF[2] = 2$, $\alpha_E = \frac{2-0}{4*2-0*3} = \frac{2}{8}$, $\alpha_F = \frac{4-3}{4*2-0*3} = \frac{1}{8}$. It follows that $\frac{\alpha_F}{\alpha_E} = \frac{1}{2}$ which is more than $\alpha^* = 0$, so $\alpha^* = 0$ and $mark = 1$ and $count$ is incremented to 3. Again note that $\frac{2}{8}\sum_{i\in E} x_i + \frac{1}{8}\sum_{i\in F} x_i \leq 1$ meets $(4,0)$ and $(3,2)$ at equality.

As stated above, $count = 3$ which is less than $Numfeaspoints - 1$ so $p = feasE[3] = 3$, $q = feasF[3] = 3$, $\alpha_E = \frac{1}{4}$, $\alpha_F = \frac{1}{12}$. It follows that $\frac{\alpha_F}{\alpha_E} = \frac{1}{3}$ which is more than $\alpha^* = 0$, so $\alpha^* = 0$ and $mark = 1$ and $count = 3 + 1 = 4$.

It follows that p, q and $count$ continue to be incremented. For each change in $count$, α_E, α_F, and $\frac{\alpha_F}{\alpha_E}$ are calculated and $\frac{\alpha_F}{\alpha_E}$ is compared to α^* in an attempt to find the minimum $\frac{\alpha_F}{\alpha_E}$. This process yields the values listed in Table 3.2.

p	q	α_E	α_F	$\frac{\alpha_F}{\alpha_E}$	mark
4	**1**	$\mathbf{\frac{1}{4}}$	**0**	**0**	**1**
3	2	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{2}$	1
3	3	$\frac{1}{4}$	$\frac{1}{12}$	$\frac{1}{3}$	1
3	4	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{1}{4}$	1
2	5	$\frac{1}{4}$	$\frac{1}{10}$	$\frac{2}{5}$	1
2	6	$\frac{1}{4}$	$\frac{1}{12}$	$\frac{1}{3}$	1
2	7	$\frac{1}{4}$	$\frac{1}{14}$	$\frac{2}{7}$	1
1	8	$\frac{1}{4}$	$\frac{3}{32}$	$\frac{3}{8}$	1
1	9	$\frac{1}{4}$	$\frac{1}{12}$	$\frac{1}{3}$	1
0	10	$\frac{1}{4}$	$\frac{1}{10}$	$\frac{2}{5}$	1
0	11	$\frac{1}{4}$	$\frac{1}{11}$	$\frac{4}{11}$	1

Table 3.2: Values for the First SSLA Inequality

As is seen in Table 3.2, the minimum $\frac{\alpha_F}{\alpha_E} = 0$ and comes from $p = 4$, $q = 1$. That means for this iteration, α^* remains at 0 and *mark* remains at 1. Thus, α_E^{*1} and α_F^{*1} are reported as $\frac{1}{4}$ and 0, respectively. Thus, the first SSL inequality is $\frac{1}{4}\sum_{i\in E} x_i + 0\sum_{i\in F} x_i \leq 1$.

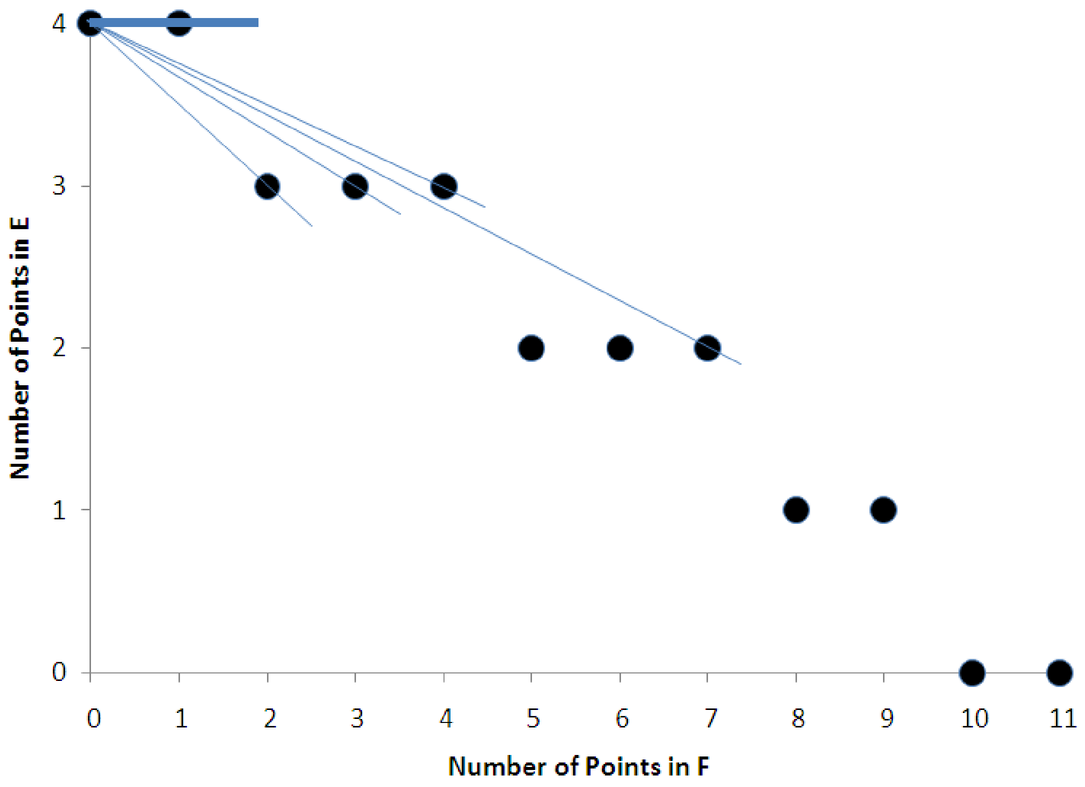

Figure 3.2: Feasible Points with First Cutting Plane

Figure 3.2 graphically depicts this process. The algorithm determines α_E and α_F such that the $\frac{\alpha_F}{\alpha_E}$ is the slope of the line between $(4,0)$ and a particular feasible point generated by the $FeasiblePoints$ subroutine. For clarity, this figure shows 5 such lines. Recognize that the algorithm would have evaluated 11 such lines. The most extreme line occurs at the minimum slope, $\frac{\alpha_F}{\alpha_E} = 0$. This line is indicated in bold and is the reported inequality.

For the next iteration, since $mark = 1$ is less than $Numfeaspoints - 1 = 11$ then $p^* = feasE[1] = 4$ and $q^* = feasF[1] = 1$. Additionally, $count = 1 + 1 = 2$, and $\alpha^* = \infty$. It is apparent that $count = 2$ is less than $Numfeaspoints - 1 = 11$, so $\alpha_E = \frac{2-1}{4*2-1*3} = \frac{1}{5}$, $\alpha_F = \frac{4-3}{4*2-1*3} = \frac{1}{5}$, and $\frac{\alpha_F}{\alpha_E} = 1$. Since $\frac{\alpha_F}{\alpha_E} < \alpha^*$, $\alpha^* = 1$, $mark = 2$,

$\alpha_E^* = \frac{1}{5}$, $\alpha_F^* = \frac{1}{5}$, and $count = 3$. Next $count = 3$ is less than $Numfeaspoints - 1$, so $\alpha_E = \frac{2}{9}$, $\alpha_F = \frac{1}{9}$, and $\frac{\alpha_F}{\alpha_E} = \frac{1}{2}$. Since $\frac{\alpha_F}{\alpha_E} < \alpha^*$, $\alpha^* = \frac{1}{2}$, $mark = 3$, $\alpha_E^* = \frac{2}{9}$, $\alpha_F^* = \frac{1}{9}$, and $count = 4$.

Again, $count = 4$ is less than $Numfeaspoints - 1$, so $\alpha_E = \frac{3}{13}$, $\alpha_F = \frac{1}{13}$, and $\frac{\alpha_F}{\alpha_E} = \frac{1}{3}$. Since $\frac{\alpha_F}{\alpha_E} < \alpha^*$, $\alpha^* = \frac{1}{3}$, $mark = 4$, $\alpha_E^* = \frac{3}{13}$, $\alpha_F^* = \frac{1}{13}$, and $count = 5$.

This process is repeated until $count \geq 11$. This iteration yields the results listed in Table 3.3.

p	q	α_E	α_F	$\frac{\alpha_E}{\alpha_F}$	mark
3	2	$\frac{1}{5}$	$\frac{1}{5}$	1	2
3	3	$\frac{2}{9}$	$\frac{1}{9}$	$\frac{1}{2}$	3
3	4	$\frac{3}{13}$	$\frac{1}{13}$	$\frac{1}{3}$	4
2	5	$\frac{2}{9}$	$\frac{1}{9}$	$\frac{1}{2}$	4
2	6	$\frac{5}{22}$	$\frac{1}{11}$	$\frac{2}{5}$	4
2	**7**	$\mathbf{\frac{3}{13}}$	$\mathbf{\frac{1}{13}}$	$\mathbf{\frac{1}{3}}$	**7**
1	8	$\frac{7}{31}$	$\frac{3}{31}$	$\frac{3}{7}$	7
1	9	$\frac{8}{35}$	$\frac{3}{35}$	$\frac{3}{8}$	7
0	10	$\frac{9}{40}$	$\frac{1}{10}$	$\frac{4}{9}$	7
0	11	$\frac{5}{22}$	$\frac{1}{11}$	$\frac{2}{5}$	7

Table 3.3: Values for Second SSL Inequality

Notice that there is a tie for the minimum $\frac{\alpha_F}{\alpha_E} = \frac{1}{3}$. In this case, the algorithm chooses the value that is furthest down the list. This means $\alpha^* = \frac{1}{3}$, $mark = 7$, and $\alpha_E^{*2} = \frac{3}{13}$ and $\alpha_F^* = \frac{1}{13}$ are reported. This line is highlighted in Table 3.3. Thus the second SSL inequality is $\frac{3}{13}\sum_{i\in E} x_i + \frac{1}{13}\sum_{i\in F} x_i \leq 1$.

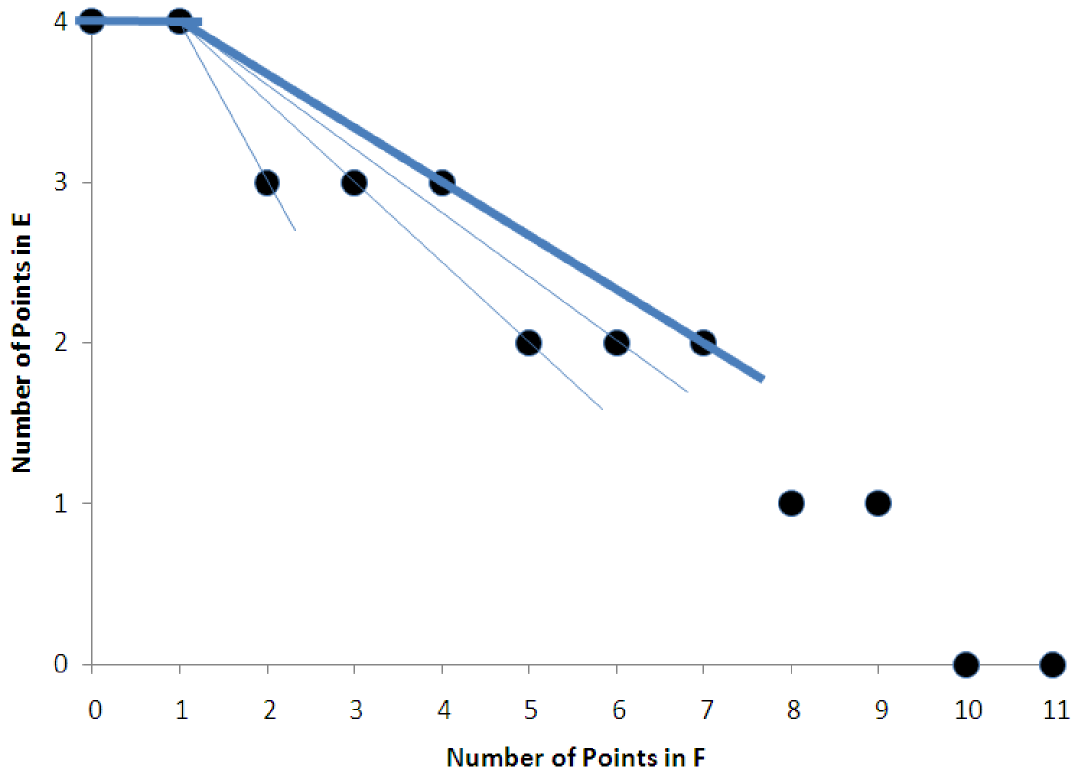

Figure 3.3: Feasible Points with First and Second Cutting Planes

Similar to Figure 3.2, Figure 3.3 graphically depicts this process. Again, the algorithm determines α_E and α_F such that the $\frac{\alpha_F}{\alpha_E}$ is the slope of the line between $(4, 1)$ and a particular feasible point generated by the $FeasiblePoints$ subroutine. In this case, the most extreme line occurs at $\frac{\alpha_F}{\alpha_E} = \frac{1}{3}$. Again, this line is indicated in bold and is the reported inequality.

Notice that if the point $(3, 4)$ had been selected, the same inequality would have resulted on this iteration. However, on the next iteration $(2, 7)$ would have been selected and would have generated the same inequality as was generated on this iteration. Hence, SSLA selects the point that is furthest down the list or furthest from the fixed point with the minimum $\frac{\alpha_F}{\alpha_E}$ ratio.

The next iteration begins with $inq = 3$, $p^* = 2$, $q^* = 7$, $count = 8$, and $\alpha^* = \infty$. The second while loop is used again to find α_E^{*3} and α_F^{*3}. The values listed in Table 3.4 are the result of the third iteration.

p	q	α_E	α_F	$\frac{\alpha_E}{\alpha_F}$	mark
1	8	$\frac{1}{9}$	$\frac{1}{9}$	1	8
1	9	$\frac{2}{11}$	$\frac{1}{11}$	$\frac{1}{2}$	9
0	10	$\frac{3}{20}$	$\frac{1}{10}$	$\frac{2}{3}$	9
0	**11**	$\mathbf{\frac{2}{11}}$	$\mathbf{\frac{1}{11}}$	$\mathbf{\frac{1}{2}}$	**11**

Table 3.4: Values for Third SSL Inequality

Again, a there is a tie for the minimum $\frac{\alpha_F}{\alpha_E} = \frac{1}{2}$ and the algorithm will select $\alpha^* = \frac{1}{2}$, $mark = 11$, $\alpha_E^{*3} = \frac{2}{11}$, and $\alpha_F^{*3} = \frac{1}{11}$. Since $mark = 11$, the while loop is not entered and the algorithm has concluded. The final inequality is $\frac{2}{11}\sum_{i\in E} x_i + \frac{1}{11}\sum_{i\in F} x_i \leq 1$. All of the cutting planes generated by SSLA in this example are shown in Figure 3.4.

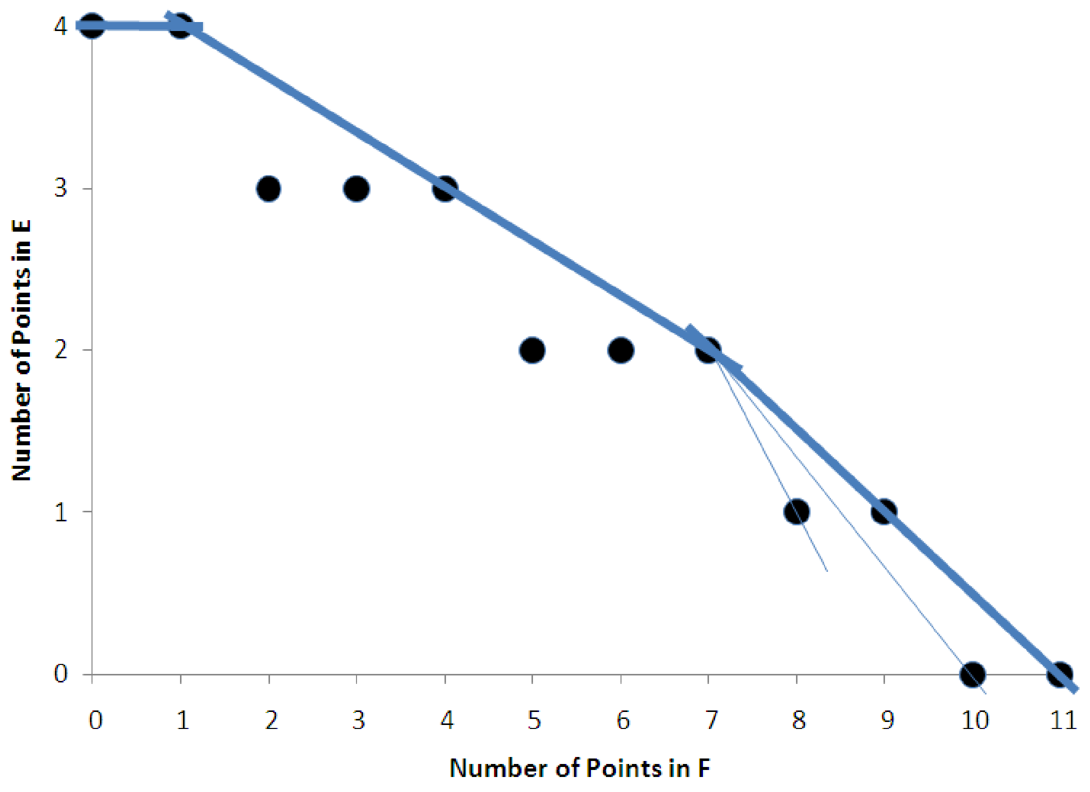

Figure 3.4: Feasible Points with First, Second and Third Cutting Planes

As is shown above, SSLA yields the following valid inequalities. Furthermore, each of these inequalities is facet defining.

$\frac{1}{4}\sum_{i\in E} x_i + 0\sum_{i\in F} x_i \leq 1$

$\frac{3}{13}\sum_{i\in E} x_i + \frac{1}{13}\sum_{i\in F} x_i \leq 1$

$\frac{2}{11}\sum_{i\in E} x_i + \frac{1}{11}\sum_{i\in F} x_i \leq 1.$

To prove that $\frac{1}{4}\sum_{i\in E} x_i + 0\sum_{i\in F} x_i \leq 1$ is facet defining, 19 affinely independent points that are feasible and meet this inequality at equality must be found. The two feasible points that generated this inequality are $(4, 0)$ and $(4, 1)$. Using Theorem 3.2.2 we must determine whether or not $(p', q') = (4, 0)$ or $(p', q') = (4, 1)$. Obviously the point (p'', q'') is the other point. In this case, $p^{'} = 4$, $q^{'} = 0$, $p^{''} = 4$, and $q^{''} = 1$.

As is shown in Figure 3.5, $p^{'}$ is permuted in the upper left portion of the matrix while $q^{'}$ is held constant and $p^{''}$ is held constant while $q^{''}$ is permuted in the bottom right portion of the matrix. Due to the sorted order of the knapsack instance, it is sufficient to ensure the following sets are not covers to prove the inequality is facet defining: $\{1, 4, 5, 6\}$, $\{2, 3, 4, 5\}$, and $\{3, 4, 5, 6, 7\}$. Hence, the following affinely independent points show that $\sum_{i \in E} \frac{1}{4} x_i \leq 1$ is facet defining.

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

Figure 3.5: Affinely Independent Points for $\sum_{i \in E} \frac{1}{4} x_i + \sum_{i \in F} 0 x_i \leq 1$

To prove that $\sum_{i \in E} \frac{3}{13} x_i + \sum_{i \in F} \frac{1}{13} x_i \leq 1$ is facet defining, again 19 affinely independant points must be found. To begin, $p^{'} = 2$, $q^{'} = 7$, $p^{''} = 4$, and $q^{''} = 1$. As is shown in Figure 3.6, $p^{'}$ is permuted in the upper left portion of the matrix while $q^{'}$ is held constant and $p^{''}$ is held constant while $q^{''}$ is permuted in the bottom right portion of the matrix. Due to the sorted order of the knapsack instance, it is sufficient to ensure the following sets are not covers to prove the inequality is facet defining: $\{1, 6, 13, 14, 15, 16, 17, 18, 19\}$, $\{4, 5, 13, 14, 15, 16, 17, 18, 19\}$, and $\{3, 4, 5, 6, 7\}$. Hence, the following affinely independent points show that $\sum_{i \in E} \frac{3}{13} x_i + \sum_{i \in F} \frac{1}{13} x_i \leq 1$ is facet defining.

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

Figure 3.6: Affinely Independent Points for $\sum_{i \in E} \frac{3}{13} x_i + \sum_{i \in F} \frac{1}{13} x_i \leq 1$

Finally, to prove that $\sum_{i \in E} \frac{2}{11} x_i + \sum_{i \in F} \frac{1}{11} x_i \leq 1$ is facet defining, 19 affinely independant points are found with $p^{'} = 2$, $q^{'} = 7$, $p^{''} = 1$, and $q^{''} = 9$. Notice that $(p^{''}, q^{''}) = (1, 9)$ is not an extreme point. This shows that the inequality can be proven to be facet defining without only using the extreme points of the polyhedron. In other words, the conditions of Theorem 3.2.2 are sufficient but not necessary.

Again, as is shown in Figure 3.7, $p^{'}$ is permuted in the upper left portion of the matrix while $q^{'}$ is held constant and $p^{''}$ is held constant while $q^{''}$ is permuted in the bottom right portion of the matrix. Due to the sorted order of the knapsack instance, it is sufficient to ensure the following sets are not covers to prove the inequality is facet defining: $\{1, 6, 13, 14, 15, 16, 17, 18, 19\}$, $\{4, 5, 13, 14, 15, 16, 17, 18, 19\}$,

$\{6,7,12,13,14,15,16,17,18,19\}$, and $\{6,10,11,12,13,14,15,16,17,18,19\}$. Hence, the following affinely independent points show that $\sum_{i\in E}\frac{2}{11}x_i + \sum_{i\in F}\frac{1}{11}x_i \leq 1$ is facet defining.

$$\begin{array}{ccccccccccccccccccc}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&1&0&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&1&1&1&1&1&0&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0
\end{array}$$

Figure 3.7: Affinely Independent Points for $\sum_{i\in E}\frac{2}{11}x_i + \sum_{i\in F}\frac{1}{11}x_i \leq 1$

One of the most exciting aspects of this research and this example is the generation of the inequality $\frac{3}{13}\sum_{i\in E}x_i + \frac{1}{13}\sum_{i\in F}x_i \leq 1$. This inequality could be found using exact simultaneous up lifting. However, to successfully create this inequality, the starting inequality would have had to have been $3\sum_{i\in E}x_i \leq 13$ or $\sum_{i\in F}x_i \leq 13$.

Consider starting with the inequality $3\sum_{i\in E}x_i \leq 13$. The dimension of the face of this inequality is -1 as no feasible point meets this inequality at equality. Thus, the natural instinct is to change the right hand side to 12. Dividing by 3 changes the inequality to an extended cover inequality. Therefore, no one would select the starting inequality $3\sum_{i\in E} x_i \leq 13$ without divine intervention. Thus, the need to consult an oracle.

A similar argument holds for $\sum_{i\in F} x_i \leq 13$. Again, the dimension of the face is -1 and decreasing the right hand side to 11 creates another extended cover inequality. Thus an oracle must be consulted in order to utilize simultaneous lifting to create this inequality.

Since no one would naturally guess either of these inequalities as a starting inequality to perform lifting, an oracle must be consulted to begin with $3\sum_{i\in E} x_i \leq 13$ or $\sum_{i\in F} x_i \leq 13$ valid inequalities as input to a simultaneous lifting algorithm. Consequently, we view SSL inequalities as capable of generating an entirely new class of cutting planes for the knapsack polyhedron.

Chapter 4

Computational Results

As stated previously, the main advancement this research provides is the introduction of a new class of lifted inequalities, SSL inequalities, and an algorithm to find SSL inequalities in binary knapsack problems with quadratic running time. The purpose of this section is to provide computational results to help determine whether or not these inequalities are useful. The computational results show that SSL inequalities allow CPLEX, an advanced optimization software, to solve problems that it otherwise would not have sufficient memory to solve.

This short computational study is conducted on a Pentium IV PC with 2.0 GB of RAM. Twenty problems of 2 different classes are solved using CPLEX with SSL cuts and CPLEX alone so the results could be compared.

Many classes of problems were attempted before a reasonable class of randomly generated problems was obtained. The goal was to find problems that CPLEX was unable to solve, thus the randomly generated problems could not be trivial. Unfortunately, random problems are often times trivial or nearly impossible to solve which is the reason it took a significant amount of effort to find reasonable classes of problems. An additional complication was the number of parameters that could be changed in each problem class.

It is important to note that it is possible for a knapsack problem to have more than one constraint. This special type of knapsack problem is referred to as a multiple knapsack problem. The type of multiple knapsack problem (MKP) that was used in this class of problems is represented as Maximize $\sum_{i \in N} c_i x_i$ subject to $\sum_{i \in N} a_{ji} x_i \leq b_j$ for $j = 1, ..., r$ and $x_i \in \{0, 1\}$ $\forall$ $i \in N$ where r is the number of rows and $a_{ji} \geq 0$ for all $i \in N$ and $j \in \{1, ..., r\}$.

The MK instances take the same general form. The constraint coefficients, a_{ji}, are randomly distributed integers between 0 and 1000 that follow a uniform distribution. The objective coefficients, c_i are calculated by summing the column coefficients and adding on a random integer between 0 and 5. In other words, $c_i = \sum_{j=1}^{r} a_{ji} + u$ where u is a number taken from a $U(0, 5)$ for all $j \in N$. The right hand side of each constraint is the sum of all a_i, in that constraint divided by one tenth of the number of variables and rounded down. Formally, $b_j = \lfloor \frac{\sum_{i=1}^{n} a_{ji}}{\frac{n}{10}} \rfloor$.

The first class of problems contains 100 variables and 3 constraints and the results are found in Table 4. The second class of problems contains 50 variables and 4 constraints and the results are found in Table 4. These instances were run with CPLEX at it default setting and also run with CPLEX at its default settings including the inequalities generated from SSLA.

Twenty problems of each class were randomly generated and solved using CPELX alone and CPLEX with the SSL cuts. The results of the trials are recorded in Tables 4 and 4. The first three columns of each table refer to the trials using CPLEX alone. The data listed in these columns are whether the problem could be solved, the amount of time it took CPLEX to solve, and the number of nodes generated. The last six columns of each table refer to the trials that were conducted using CPLEX with the SSL inequalities. The data listed in these columns are whether the problem could be solved, the amount of time it took CPLEX with the SSL inequalities to solve, the number of nodes generated, the size of E, the size of F, and the number of SSL inequalities generated.

It is important to note that if there was not enough memory for the problem to be solved, the times listed are the times before the maximum amount of memory was used. Additionally, all times are reported in seconds and the final row of the table is bolded and represents the average of each column. The average time reported is computed using only the instances where both CPLEX alone and CPLEX with the SSL inequalities were able to solve the problem to optimality.

In this computational study, SSLA is applied to each constraint. Thus, the 100 variable instances have 3 implementations of SSLA and the 50 variable instances have 4 such implementations. The sets E and F partition the set N and the elements of E are chosen to be the indices with the largest coefficients in the particular constraint. The set E has between one fifth to one half of the indices and it is selected so that the difference between the coefficients with indices in E and the coefficients with indices in F are as large as possible.

CPLEX			SSLA					
Solved	*Time*	*Nodes*	*Solved*	*Time*	*Nodes*	$\|E\|$	$\|F\|$	*Ineq*
No	538.16	3630000	Yes	2418.45	40400000	38	62	11
No	925.73	5009232	Yes	1303.42	19670000	33	67	12
Yes	2449.88	38980000	Yes	2941.76	47010000	44	56	10
Yes	527.19	10180000	Yes	1916.08	27900000	51	49	13
No	586.11	9260000	Yes	954.77	14750000	32	68	11
Yes	681.59	119400000	Yes	1117.58	13600000	33	67	14
No	649.31	10540000	Yes	3140.13	52030000	51	49	13
No	657.34	10660000	No	2543.72	37300000	48	52	12
No	594.61	9530000	No	1728.7	21970000	50	50	12
Yes	235.55	4650000	Yes	267.56	5080000	38	62	14
No	557.92	8820000	Yes	2260.17	30300000	30	70	14
Yes	1766.91	29160000	Yes	2281.49	26850000	39	61	12
Yes	517.17	7910000	Yes	808.13	8760000	36	68	13
Yes	502.13	7678324	Yes	201.45	2950000	40	60	12
No	612.89	9740000	Yes	1266.80	16949187	41	59	12
No	540.50	8420000	No	1077.50	17000000	31	69	11
Yes	1175.70	18500000	Yes	739.47	9610000	45	55	12
Yes	761.63	12960000	No	1394.06	20610000	31	69	11
Yes	291.20	5500000	Yes	666.81	9360000	49	51	15
Yes	526.89	9060000	No	325.75	430000	37	63	12
11/20	**905.26**	**15083009**	**15/20**	**1215.59**	**15020264**	**39.85**	**60.35**	**12.3**

Table 4.1: Data Reported for Randomly Generated Problems with 100 Variables

CPLEX			SSLA					
Solved	*Time*	*Nodes*	*Solved*	*Time*	*Nodes*	$\lvert E\rvert$	$\lvert F\rvert$	*Ineq*
No	1316.53	20800000	Yes	3140.72	47160000	15	35	17
Yes	874.95	1591000	Yes	1314.13	1530000	13	37	15
Yes	2097.27	36770000	Yes	2586.16	39480000	16	34	15
No	1624.41	26020000	Yes	3751.42	56900000	20	30	17
No	1603.11	25550000	Yes	4046.36	61250000	22	28	15
No	775.78	12230000	No	2423.75	33250000	18	32	15
No	930.56	14870000	No	3225.3	45340000	13	37	15
Yes	1927.05	33580000	Yes	2696.80	34180000	12	38	15
No	918.13	14990000	No	3064.95	41950000	16	34	16
No	994.39	16290000	Yes	5316.27	84220000	26	24	13
No	877.19	14170000	Yes	4840.70	76650000	22	28	16
Yes	1984.73	34530000	Yes	2687.19	38620000	24	26	16
No	887.09	14490000	Yes	4006.8	59850000	11	39	17
No	1008.36	16680000	Yes	4737.63	72820000	12	38	14
No	1078.72	17560000	Yes	5920.39	85880000	17	33	17
Yes	1013.27	16280000	Yes	2595.77	37790000	22	28	17
No	1568.68	13790000	Yes	5845.17	85730000	17	33	14
No	895.99	14440000	No	3550.44	44650000	26	24	20
No	922.38	14640000	No	3224.28	47220000	11	39	12
Yes	3390.11	55140000	Yes	5577.53	88640000	17	33	16
6/20	**1881.23**	**36478231**	**15/20**	**2909.6**	**38217718**	**17.5**	**32.5**	**15.6**

Table 4.2: Data Reported for Randomly Generated Problems with 50 Variables

One of the most exciting results is the number of distinct SSL inequalities. For these problems, there was an average of 12 SSL inequalities for the 100 variable instances and 15 for the 50 variable instances. Furthermore, these inequalities are not simply cover inequalities. One inequality that was generated in the above trials is $7\sum_{i\in E} x_i + \sum_{i\in F} x_i \leq 57$ where $|E| = 17$ and $|F| = 33$. This was created when 7 elements from E are taken and 8 elements from F or 6 elements from E and 15 elements from F. Another such inequality is $2\sum_{i\in E} x_i + 3\sum_{i\in F} x_i \leq 42$ where again $|E| = 17$ and $|F| = 33$.

Importantly, SSLA results in very quick preprocessing times. In all trials, the preprocessing time was less than $\frac{1}{1000}$ of a second and is not reported in the table due to this short processing time. The fact that an average of 15 cuts can be generated in less than one thousandth of a second indicates that SSLA can be used as a preprocessing step for commercial IP solvers.

The most significant result is that SSL cuts allowed CPLEX to solve problems to optimality that it otherwise would not have been able to solve. Of the 40 instance CPLEX alone only solved 17 to optimality while CPLEX with SSL cuts solved 30 to optimality. Thus SSL cuts allowed a 76 percent improvement over CPLEX alone.

A primary reason for these computational advancements is that the SSL cuts are strong. The number of noninteger points in unfathomed nodes for CPLEX alone is consistently the number of constraints. In contrast, once SSL cuts are implemented, the unfathomed nodes frequently had an extra one or two noninteger points. Thus, these SSL constraints are still active. This important result even occured deep in the tree after millions of nodes had been fathomed. In other words, many times the SSL inequalities were very strong inequalities.

Observe that implementing the SSL cuts increased the amount of average time required to solve the problem to optimality (when both CPLEX and CPLEX with SSL cuts solved the problems). This result is surprising and may be because the SSL cuts dramatically increase the number of constraints, 4 to 19 or 3 to 15. Thus, the basis that is stored is larger for each node and one expects the time to evaluate each node would increase slightly.

Although there are some minor tradeoffs for implementing SSLA, the theoretical and computational results obtained in this research are very promising because they suggest this research can be useful in lifting. Even more important than these results themselves is the future work that this research might lead to. There are many ways this researh can be adapted to different types of problems to be of use in future lifting research. Some of these possibilities are listed in Chapter 5.

Chapter 5

Conclusion

This research has provided a significant advancement to the area of lifting in integer programming. It has introduced Synchronized Simultaneous Lifting (SSL) as a new type of lifting. SSL finds classes of valid inequalities that could not be found using any of the previously discovered lifting methods without the consultation of an oracle. The main advantages of SSL inequalities are that no starting valid inequality is required, arbitrary sets may be used to begin and many inequalities are generated using the method. Additionally, this thesis provided further conditions to ensure the inequalities are facet defining.

This thesis also introduced a quadratic time algorithm that can be used in binary knapsack problems to find SSL inequalities. These inequalities can significantly decrease the amount of memory required to solve computationally difficult, randomly generated problems. The improvement can be so significant that the inequalities sometimes allow

CPLEX to solve problems that it otherwise would not have been able to solve.

5.1 Future Research

While the results listed in Chapters 3 and 4 show SSL inequalities can be useful in binary knapsack problems and can generate facet defining inequalities, much computational work remains to be done. It would be very helpful to find more classes of randomly generated problems that easily lend themselves to the implementation of the SSLA. Additionally, more work could be done in implementing these problems in various benchmark problems so that the results may be compared.

Nearly as important as the computational results themselves are the branches of research that can be spun off of these results. Another area of opportunity for research is to expand this theory into new types of problems. An obvious expansion is problems with multiple constraints. SSLA works with one knapsack constraint but could likely be expanded using the same ratio idea that inspired the SSLA.

Another interesting opportunity that would likely yield useful results would be to expand SSLA to include more than two sets. The SSLA uses α_E, α_F, and $\frac{\alpha_F}{\alpha_E}$ to determine the inequalities. This could easily be expanded to include an α_G and corresponding ratios.

Additionally, SSL could be expanded to other classes of integer programs. Examples of these would be nonbinary problems, general mixed integer programs,and multiple constraint knapsack problems.

Finally, opportunities exist to implement the SSLA (or another algorithm that finds SSL inequalities) sequentially to allow for the discovery of even more inequalities. This would allow more than 2 α values to be included in a SSL inequality.

Bibliography

[1] Anbil R., Gelman E., Patty B., and Tanga R. (1991), "Recent advances in crew pairing optimization at American airlines," *Interfaces* **21**, 62-74.

[2] Arunapuram, S., K. Mathur and D. Solow (2003). "Vehicle routing and scheduling with full truckloads," *Transportation Science*, **37**, n 2, May 2003, 170-82.

[3] Atamtürk, A. (2004). "Sequence independent lifting for mixed-integer programming," *Operations Research*, **52** (3), 487-491.

[4] Balas, E., (1975). "Facets of the knapsack polytope", *Mathematical Programming*, **8**, 146-164.

[5] Balas, E and E. Zemel (1978). "Facets of the knapsack polytope from minimal covers," *SIAM Journal of Applied Mathematics*, **34**, 119-148.

[6] Balas, E. and E. Zemel, (1984). "Lifting and complementing yields all the facets of positive zero-one programming polytopes," in *Mathematical Programming, Proceedings of the International Conference on Mathematical Programming*, R.W. Cottle et al., eds., 13-24.

[7] Bean J., C. Noon, and G. Salton.(1987), "Asset Divestiture at Homart Development Company," *Interfaces* **17**, 48-64.

[8] Bertsimas, D., C. Darnell and R. Soucy (1999). “Portfolio construction through mixed-integer programming at Grantham, Mayo, Van Otterloo and Company,” *Interfaces*, **29**, 1, Jan.-Feb. 1999, 49-66.

[9] Bitran G., and A. Hax (1981), "Disaggregation and Resource Allocation Using Convex Knapsack Problems with Bounded Variables," *Management Science* **27** (4), 431-441 .

[10] Brown, D. and I. Harrower (2004). “A new integer programming formulation for the pure parsimony problem in haplotype analysis,” *Algorithms in Bioinformatics. 4th International Workshop, WABI 2004. Proceedings Lecture Notes in Bioinformatics* **3240**, 254-65.

[11] 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.

[12] Easton, T. and K. Hooker, “Simultaneously Lifting Sets of Binary Variables into Cover Inequalities for Knapsack Polytopes,” *Discrete Optimization, Special Issue: In Memory of George B. Dantzig*, **5**(2) May 2008, 254-261.

[13] Ferreira, C., C. de Souza and Y. Wakabayashi (2002). "Rearrangement of DNA fragments: A branch-and-cut algorithm," *Discrete Applied Mathematics*, **116**, (1-2), 15 Jan. 2002, 161-77.

[14] Gomory, R. E., (1958). "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, **64**, 275-278.

[15] Gomory, R. (1969). "Some polyhedra related to combinatorial problems," *Linear Algebra and its Applications*, **2**, 451-558.

[16] Gu, Z., G. Nemhauser, and M. Savelsbergh (2000). "Sequence independent lifting in mixed integer programming," *Journal of Combinatorial Optimization*, **4**, 109-129.

[17] Gutierrez, Talia, (2007) "Lifting general integer variables," *MS Thesis*, Department of Industrial and Manufacturing Systems Engineering, Kansas State University.

[18] Hammer, P., E. Johnson, and U. Peled, (1975). "Facets of regular 0-1 polytopes," *Mathematical Programming*, **8**, 179-206.

[19] Karp, R. (1972). "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York 85-103.

[20] Kaufman, D., J. Nonis, and R. Smith (1998). "A mixed integer linear programming model for dynamic route guidance," *Transportation Research, Part B (Methodological)*, **32B** (6), Aug. 1998, p 431-40.

[21] Kolliopoulos S. and P. Ilissia.(2007), "Partially ordered knapsack and applications to scheduling ," *Discrete Applied Mathematics archive* **155** (8), 889-897.

[22] Kubik, Lauren, (2009) "Simultaneously lifting multiple sets in binary knapsack integer programs," *MS Thesis*, Department of Industrial and Manufacturing Systems Engineering, Kansas State University.

[23] Land, A.H. and A.G. Doig (1960). "An automatic method for solving discrete programming problems," *Econometrica*, **28**, 497-520.

[24] Lee, E. and M. Zaider, (2000). "Mixed integer programming approaches to treatment planning for brachytherapy – Application to permanent prostate implants," *Annals of Operations Research, Optimization in Medicine*, 2000, 147-163.

[25] Lee, E., T. Fox and I. Crocker, (2003) "Integer programming applied to intensity-modulated radiation treatment planning optimization," *Annals of Operations Research, Optimization in Medicine*, 2003, 119: 165-181.

[26] Moore E. Jr., J. Warmke, and L. Gorban.(1991), "The Indispensable Role of Management Science in Centralizing Freight Operations at Reynolds Metals Company," *Interfaces* **21**, 107-129.

[27] Nemhauser, G. and L. Wolsey, (1988). *Integer and combinatorial optimization*, John Wiley and Sons, New York.

[28] Padberg, M. (1973). "On the facial structure of set packing polyhedra," *Mathematical Programming*, **5**, 199-215.

[29] Pinto, R. and B. Rustem (1998) "Solving a Mixed-Integer Multiobjective Bond Portfolio Model Involving Logical Conditions," *Annals of Operations Research*, **81**, 1998, 497-513.

[30] Ruiz, R., C. Maroto and J. Alcaraz (2004). "A decision support system for a real vehicle routing problem," *European Journal of Operational Research*, **153** (3), 16 March 2004, 593-606.

[31] Sharma, Kamana, (2007) "Simultaneously lifting sets of variables in binary knapsack problems," *MS Thesis*, Department of Industrial and Manufacturing Systems Engineering, Kansas State University.

[32] Shebalov, S. and D. Klabjan, (2006) "Sequence independent lifting for mixed integer programs with variable upper bounds," *Mathematical Programming*, **105** (2-3), 523-561.

[33] Subramanian, R., Scheff R., Quillinan J., Wiper D. S., and Marsten R. (1994) "Coldstart: Fleet Assignment at Delta Air Lines." *INTERFACES*, **24**, January-February 1994, 104-120.

[34] Toth, P. (1997). "An exact algorithm for the vehicle routing problem with backhauls," *Transportation Science*, **31** (4), Nov. 1997, 372-85.

[35] Urban, T. (2003). "Scheduling sports competitions on multiple venues," *European Journal of Operational Research*, **148** (2), 16 July 2003, 302-11.

[36] Wolsey, L.A. (1975). “Faces for a linear inequality in 0-1 variables,” *Mathematical Programming*, **8**, 165-178.

[37] Wolsey, L.A. (1976). “Facets and strong valid inequalities for integer programs,” *Operations Research*, **24**, 367-372.

[38] Wolsey, L.A. (1977). “Valid inequalities and superadditivity of 0/1 integer programs,” *Mathematics of Operations Research*, **2**, 66-77.

[39] Zemel, E. (1978). “Lifting the facets of 0-1 polytopes” *Mathematical Programming*, **15**, 268-277.

[40] Zemel, E. (1989). “Easily computable facets of the knapsack polytope,” *Mathematics of Operations Research*, **14**, 760-764.