EXACT SYNCHRONIZED SIMULTANEOUS UPLIFTING OVER ARBITRARY INITIAL INEQUALITIES FOR THE KNAPSACK POLYTOPE

by

CARRIE AUSTIN BEYER

B.S., Kansas State University, 2011

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

2011

Approved by:

Major Professor

Dr. Todd Easton

ABSTRACT

Integer programs (IPs) are mathematical models that can provide an optimal solution to a variety of different problems. They have been used to reduce costs and optimize organizations. Additionally, IPs are NP-complete resulting in many IPs that cannot be solved. Cutting planes or valid inequalities have been used to decrease the time required to solve IPs.

Lifting is a technique that strengthens existing valid inequalities. Lifting inequalities can result in facet defining inequalities, which are the theoretically strongest valid inequalities. Because of these properties, lifting procedures are used in software to reduce the time required to solve an IP.

The thesis introduces a new algorithm for exact synchronized simultaneous uplifting over an arbitrary initial inequality for knapsack problems. Synchronized Simultaneous Lifting (SSL) is a pseudopolynomial time algorithm requiring $O(nb + n^3)$ effort to solve. It exactly uplifts two sets simultaneously into an initial arbitrary valid inequality and creates multiple inequalities of a particular form. This previously undiscovered class of inequalities generated by SSL can be facet defining.

A small computational study shows that SSL is quick to execute, requiring on average less than a quarter of a second. Additionally, applying SSL inequalities to a knapsack problem enabled commercial software to solve problems that it could not solve without them.

Contents

List of Figures

List of Tables

Dedication

My work is dedicated to my family for their love, support, and understanding.

Acknowledgments

There are many who, through their support and efforts, aided in the success of this thesis.

I would first like to acknowledge Dr. Todd Easton. It is not an understatement to say that without his help, patience, and technical knowledge, this thesis would not exist. Additionally, I would like to recognize the efforts of Dr. John Wu and Dr. Diego Maldonado 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 during my college career.

Chapter 1

Introduction

Integer programming is a tool that can and has been used reduce costs, optimize organizations, and businesses. Integer programs (IPs) are mathematical models that can provide an optimal solution to a variety of different problems and take the form maximize $c^T x$ subject to $Ax \leq b$ and $x \in \mathbb{Z}^n_+$, where $A \in \mathbb{R}^{mxn}$ and $b \in \mathbb{R}^{mx1}$. This thesis presents an algorithm to generate cutting planes using exact synchronized simultaneous uplifting for the knapsack problem. This algorithm can be applied to numerous IPs.

One example of the usefulness of IPs involves The United States Postal Service (USPS). The USPS faces challenges in profitability and quality. To address these issues, a large scale integer programming algorithm was developed to identify cost saving opportunities along delivery routes. The USPS has saved over $5 million annually after implementing an integer program in 2007 [30]. This amounts to a 24% reduction of transportation costs.

USPS is only one example of the significant research in transportation applications of IPs [2, 30, 34, 39]. The airline industry uses IPs for crew assignment [1] and to determine the optimal flight legs [37]. IPs can also solve problems in genetics research [11, 19]. Sports games are frequently scheduled using IPs [16, 40]. These examples are only a small sampling of the variety of IP applications used throughout the world.

Beyond the industry applications, the average person unknowingly attempts to solve IPs. For instance, a child attends the state fair with $40 from her parents and wants to find the combination of rides and snacks that will make her happiest without exceeding her budget. This brief example illustrates a simple IP called the knapsack problem (KP), which is critical to this thesis.

The classic KP example involves a camper preparing for her trip and determining what to bring in her knapsack. She has the option to either select each item, or not. Each item has an associated benefit and weight. She maximizes the benefit, restricted by the amount she can carry.

The most common method to solve IPs is the branch and bound algorithm. It uses the optimal solution from linear relaxations. A linear relaxation (LR) is the IP formulation without the integer requirement. When the LR contains fractional values branch and bound creates two nodes, also referred to as children. One child adds the constraint that a fractional variable is less than or equal to the floor of its value from the LR. The other child adds the constraint that the variable must be greater than or equal to its ceiling. By adding these constraints, the non integer space between some integer

points is no longer valid in either of these problems. Running this process iteratively enumerates all integer points. Eventually branch and bound finds the optimal solution, but it can require exponential time. To reduce the time required to solve IP problems with branch and bound, cutting planes are frequently used.

A cutting plane is a valid inequality that when added to the problem eliminates some fraction of the LR's feasible area. A valid inequality is satisfied by every feasible IP solution. Applying iterations of cutting planes can force the optimal LR solution to become an integer solution, thus the IP is solved. Facet defining cutting planes are the theoretically strongest valid inequalities.

One method to obtain a facet defining inequality is through lifting. Lifting uses the restricted polyhedron which forces some variables in the problem to specific values. Lifting alters the coefficients on the variables of a valid inequality to make it stronger. It is possible to create a facet defining inequality through lifting. This thesis focuses on the development of facet defining inequalities through exact synchronized simultaneous uplifting.

1.1 Research Motivation

Bolton [10] developed an exact synchronized simultaneous uplifting algorithm. This is not a robust lifting algorithm because the restricted polyhedron is the empty set. The goal of this research is to develop an exact synchronized simultaneous uplifting algorithm which lifts two sets into an arbitrary initial valid inequality for the knapsack instance.

Thus presenting a new lifting method with the objective of generating cutting planes to reduce the time to solve IPs.

1.2 Research Contributions

This thesis presents a new exact synchronized simultaneous uplifting (SSL) algorithm for the knapsack polytope. The input to SSL is an initial valid inequality, a knapsack problem, and two sets of mutually exclusive indices. A table is generated based on the indices selected from the initial valid inequality. Feasible points are generated for each table value. These points are used to calculate the exact synchronized simultaneous uplifting coefficients.

The primary contributions of this thesis lie in the creation of exact synchronized simultaneously uplifted variables in an arbitrary inequality for the KP polyhedron. Theoretical results provide the conditions under which the cutting planes generated are facet defining. Additionally SSL runs in pseudopolynomial time, $O(nb + n^3)$.

Results from a small computational study show applying SSL technology enabled CPLEX 10.0 [38], a commercial integer programming software, to solve 29.5% more problems than its traditional methods. Thus, SSL improves upon the solutions available from commercial software. Since SSL creates cutting planes, they can be applied to traditional linear integer programs and even nonlinear integer programs.

1.3 Outline

Chapter 2 contains an overview of integer programming and polyhedral theory providing the background information necessary to understand the research presented in this thesis. Topics covered include: cutting planes and facet defining inequalities, the knapsack problem, covers, and lifting. Formal definitions along with detailed examples aid in the understanding of these complex topics.

Chapter 3 presents SSL. First, notation is defined followed by an overview of the algorithm. Next, the pseudocode provides the details to execute SSL. Proof of correctness and theoretical conditions for facet defining inequalities are presented. Finally, an example demonstrates SSL produces multiple facet defining inequalities.

The results from the computational study are found in Chapter 4. The class of problems generated is described along with data to support the effectiveness of SSL. Data presented includes nodes evaluated, preprocessing times, and time required to solve to optimality. The results show that SSL cuts enabled CPLEX to solve significantly more problems than its traditional methods.

Finally, Chapter 5 provides a conclusion of SSL and its computational results. This chapter also contains ideas and extensions discovered during the development of SSL that can be pursued as future research.

Chapter 2

Background Information

This chapter introduces the integer programming and mathematical background necessary to understand this thesis. Concepts discussed include integer programming, the definition and use of cutting planes, the knapsack problem with an example, covers, and a variety of lifting techniques. Through the discussion in this chapter, a basic understanding of the concepts should lead to an appreciation and understanding of the advancements presented in Chapter 3. Nemhauser and Wolsey [28] is an excellent technical reference for additional integer programming information.

2.1 Integer Programming and Polyhedral Theory

An integer program (IP) has a linear objective equation that is maximized or minimized and subject to a finite set of linear constraints. The decision variables are required to

be integer. Thus, IPs follow the form:

Maximize $z^{IP} = c^T x$

subject to $Ax \leq b$

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

where $c \in \Re^n$, $A \in \Re^{mxn}$ and $b \in \Re^m$.

The feasible space for an IP is defined as $P = \{x \in \mathbb{Z}_+^n : Ax \leq b\}$. The solution space, P contains a set of countable points. Let N be the set of indices of an IP, $N = \{1, ..., n\}$.

A basic concept in many optimization methods is convexity. A set is convex if every point on the line segment connecting any two points from the set, is also in the set. Formally, S is convex if, and only if, $\lambda s_1 + (1 - \lambda)s_2 \in S$ for all $s_1, s_2 \in S$ and $\lambda \in [0, 1]$. The convex hull of a set S, $conv(S)$, is defined as the intersection of all convex sets that contain S. Observe that P is not a continuous region and therefore the solution to an IP is not convex unless there is only zero or one feasible solution.

Algorithms used to solve IPs often rely on solving iterations of the linear relaxation (LR) of an IP. An IP is transformed into a LR by removing the integer restrictions from the problem. Here, the linear relaxation that corresponds to the given problem is referred to as IP^{LR} with the format, maximize $c^T x$ subject to $Ax \leq b, x \in \Re_+^n$. Let P^{LR} be defined as the LR's feasible region, $P^{LR} = \{x \in \Re_+^n : Ax \leq b\}$.

The solution to a single linear inequality is a half space. All $x \in \Re^n$ such that $\sum_{j=1}^n \alpha_j x_j \leq \beta$ is a half space and is clearly convex. A polyhedron is the intersection of a finite number of half spaces. Using these definitions, the feasible region of a linear

program $\{x \in \Re_+^n : Ax \leq b\}$ is a polyhedron and is convex. When a polyhedron is bounded it is known as a polytope. Furthermore, P^{ch} is a polyhedron.

The dimension is an important characteristic of a polyhedron. The dimension of a polyhedron is the number of linearly independent vectors that define the space. Vectors, $v^1, ..., v^q \in \Re^n$ are linearly independent if and only if the unique solution to $\sum_{i=1}^{q} \lambda_i v_i = 0$ is $\lambda_i = 0$ for all $i = 1, ..., q$.

Because P consists of a set of points, affine independence is used to determine $dim(P^{ch})$ points found within P. A set of points $x_1, x_2, x_3, ..., x_r \in \Re_+^n$ is affinely independent if and only if, $\sum_{j=1}^{r} \lambda_j x_j = 0$ and $\sum_{j=1}^{r} \lambda_j = 0$ is uniquely solved by $\lambda_j = 0 \; \forall$ $j = 1, ..., r$.

The dimension of a convex set is the maximum number of affinely independent points minus one. The reduction by one is due to the fact that one of the points represents the origin and linearly independent vectors can then be created by connecting this "origin" and any of the other affinely independent points.

2.2 Cutting Planes and Facets

With a polyhedron and its dimension defined, cutting planes, valid inequalities, faces, and facets can now be introduced and discussed. This section covers these topics and how they apply to integer programming.

In integer programming there are two different polyhedrons to consider. There is P^{ch}

which has integer extreme points and P^{LR} which may not. The relationship between these two polyhedrons and the transformation from P^{LR} to P^{ch} may aid in the ability to solve an IP.

A cutting plane is a valid inequality that is used to eliminate area of P^{LR} without removing any points in P. The inequality $\sum_{j=1}^{n} \alpha_j x_j \leq \beta$ is a valid inequality of P^{ch} if and only if every $x \in P$ satisfies $\sum_{j=1}^{n} \alpha_j x_j \leq \beta$. The idea of a face is used to theoretically judge the strength of a valid inequality.

Every valid inequality induces a face of a polyhedron. The valid inequality, $\sum_{j=1}^{n} \alpha_j x_j \leq \beta$, defines a face $F \subseteq P^{ch}$ of the form $F = \{x \in P^{ch} : \sum_{j=1}^{n} \alpha_j x_j = \beta\}$. If $F \neq \{\emptyset\}$, then F is said to support P^{ch}. Thus, a face consists of all points in the polyhedron that meet the inequality at equality.

A polyhedron can be described as a set of faces. The minimum definition includes only the faces that have the dimension of one less than $dim(P^{ch})$. These are known at facet defining inequalities and are the strongest of all valid inequalities.

Facet defining inequalities are vitally important because they are both necessary and sufficient for the description of P^{ch}. Any inequality that is not a facet defining inequality of P^{ch} is redundant and can be eliminated. Determining P^{ch} is critical to integer programming research because solving a linear program over P^{ch} generates an optimal integer solution. This means that branch and bound is not required and allows the optimal solution to be found at the root node. The following example illustrates these concepts.

Example 2.1

Consider the following integer program:

Maximize $z^{IP} = 2x_1 + 3x_2$

Subject to $3x_1 + 2x_2 \leq 12$

$x_1 + 2x_2 \leq 6$

$x_1,\ x_2\ \in \mathbb{Z}_+$.

Figure 2.1 provides a graphical view of this IP. The first constraint, $3x_1 + 2x_2 \leq 12$, passes through points $(0, 6)$, C, and D. The second constraint, $x_1 + 2x_2 \leq 6$, passes through the points A, B, C, and $(6, 0)$. Clearly P^{LR} is defined by these two constraints and the x_1 and x_2 axes. The solution to the LR is the point $(3, 1.5)$ giving a z value of 10.5. The large circles represent P, the feasible integer points.

The inequality, $x_1 + x_2 \leq 4$, removes no integer points and therefore is valid as shown by the dashed line in Figure 2.2. If it is applied to the problem as a cutting plane, it removes the "BCD" triangle. The new LR solution is the point (2, 2) with a z^{LR} value of 9. Since the LR solution is integer, the IP solution is also the point (2, 2) with a z^{IP} value of 9.

To show that $x_1 + x_2 \leq 4$ induces a facet, it must be valid and its face must have a dimension of one less than P^{ch}. The dimension of P^{ch} can be found by bounding it from above and below. For this example, the points $(0, 0)$, $(0, 1)$ and $(1, 0)$ are all affinely independent and feasible. Therefore, the $dim(P^{ch}) \geq 3 - 1$. Because there are

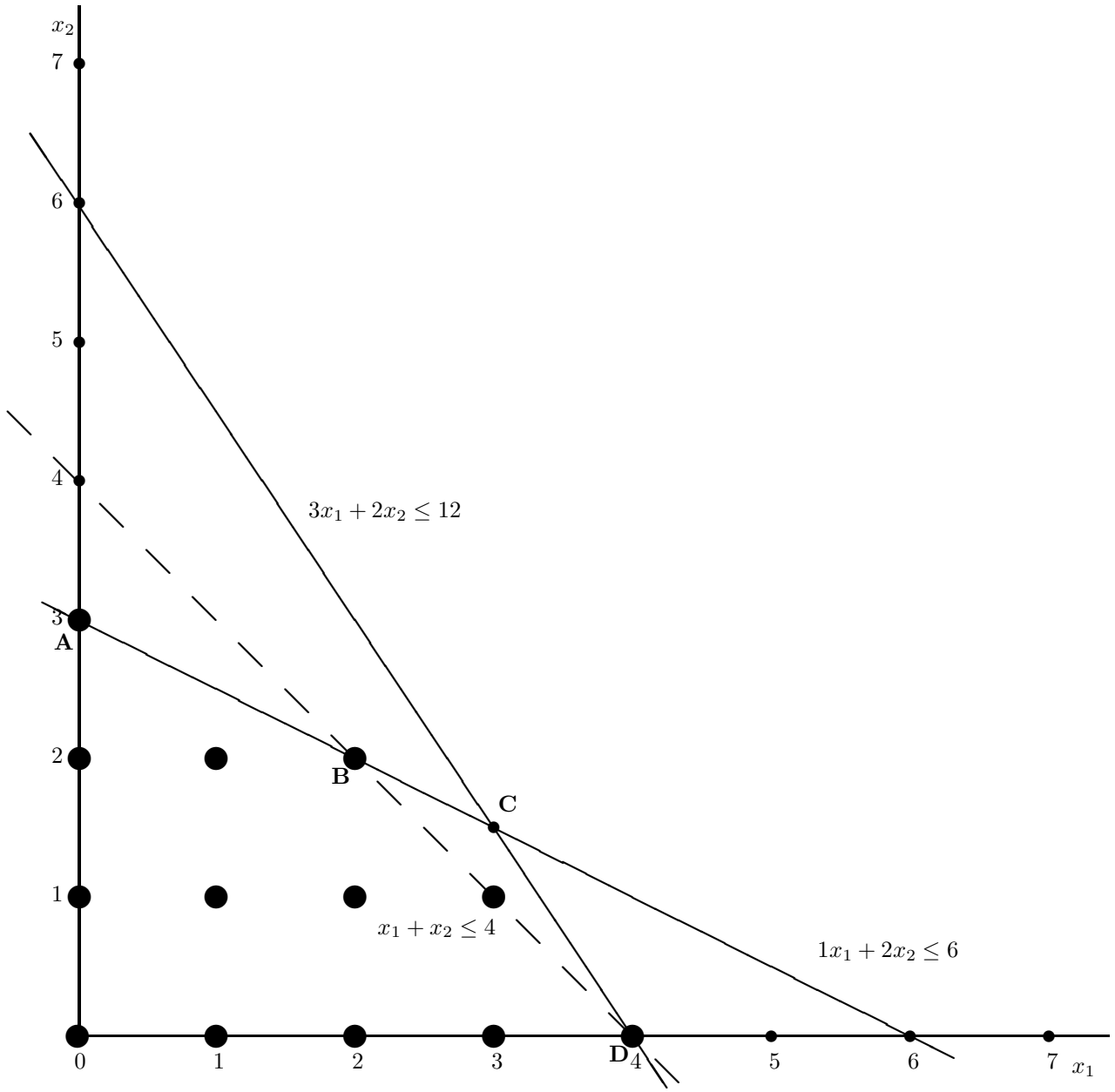

Figure 2.1: Cutting Plane Method in Example 2.1

two variables, $dim(P^{ch}) \leq 2$. Therefore, the dimension of P^{ch} is two.

The inequality, $x_1+x_2 \leq 4$ is valid as seen in Figure 2.1 because it does not violate any integer points. The dimension of the face of the inequality is at least one, with affinely independent points $(1,3)$ and $(4,0)$. These points are feasible and satisfy $x_1 + x_2 = 4$. Additionally, the point $(0,0)$ is feasible and does not satisfy $x_1 + x_2 = 4$, thus the face is not the entire P^{ch}. Therefore the face's dimension is one. Consequently, $x_1 + x_2 \leq 4$ is a facet defining inequality.

The other facet defining inequalities for P^{ch} in this example are $x_1+2x_2 \leq 6$, $x_1 \geq 0$, and $x_2 \geq 0$. With the inclusion of all facets, P^{ch} is fully defined and all corner points are integer.

This example shows how valid inequalities are used to help solve IPs. The number of variables and complexity are directly related. As the number of variables increases, the difficulty of finding valid and facet defining inequalities also increase. Knapsack problems are the simplest form of IPs with n variables and are the focus for the next section and also this thesis.

2.3 The Knapsack Problem

An important area of integer programming research is the knapsack problem (KP), which is a particular class of integer programs. The name is derived from an example of a camper packing a knapsack with items for a trip. There are n items available to

bring each with a benefit, c_j, and weight, a_j. The camper looks to bring the maximum benefit while being restricted by the total amount of weight that she can carry, b.

The binary KP formulation begins by setting $x_j = 1$, if that item is selected; and $x_j = 0$, if not. The IP is , maximize $\sum_{j=1}^{n} c_j x_j$, subject to $\sum_{j=1}^{n} a_j x_j \leq b, x_j \in \mathbb{B}$ for all $j = 1, 2, ..., n$ where $a_j \geq 0 \; \forall \; j = 1, ..., n$. Let P_{KP} represents the set of feasible solutions, $P_{KP} = \{x \in \mathbb{B}^n : \sum_{i=1}^{n} a_i x_i \leq b\}$. The convex hull is then denoted as $P_{KP}^{ch} = conv(P_{KP})$.

There exist numerous methods to solve a knapsack problem. Integer programming, dynamic programming and shortest path models are among the most common and frequently taught in college courses. [32] provides an excellent background information on other methods used to solve the knapsack problem.

A reason that KPs are so highly studied is in their value to general integer programming. Any binary integer programming constraint can be converted into a knapsack through a simple transformation. If the constraint is an equality, then two constraints are formed, a '$\leq$' and a '$\geq$'. Each '$\geq$' constraint is multiplied through by a -1. If there is an $a_i < 0$, then x_i is replaced with $1 - x_i'$. With this transformation, knapsack cuts can be applied to any single binary integer programming constraint.

In a general KP, it can be assumed that the a_i's are sorted in descending order; if $i, j \in N$ and $i < j$, then $a_i \geq a_j$. Furthermore, if $a_1 \geq b$, then $x_1 = 0$ for all feasible solutions and x_1 can be eliminated from the problem. Thus there are always $n + 1$ an affinely independent points when, only one item is included, for all items, and when no items are included. Consequently, P_{KP}^{ch} is fully dimensional,$dim(P_{KP}^{ch}) = n$. The

following example illustrates a basic KP.

Example 2.2

Consider a camper deciding between 25 items to take on a trip. The hiker can carry 379 ounces, or approximately 24 pounds. The benefit and weight in ounces of each item can be found in Table 2.1, followed by the problem's IP formulation.

Item#	1	2	3	4	5	6	7	8	9	10	11	12	13
Benefit	20	7	46	79	9	84	42	34	91	107	117	3	39
Weight	105	93	92	90	74	72	72	71	70	65	64	62	61
Item#	14	15	16	17	18	19	20	21	22	23	24	25	
Benefit	12	60	87	90	54	2	22	99	46	17	5	27	
Weight	60	44	44	44	43	43	42	42	41	41	40	40	

Table 2.1: Benefit and weight of items that may be taken in the knapsack

Maximize

$$20x_1 + 7x_2 + 46x_3 + 79x_4 + 9x_5 + 84x_6 + 42x_7 + 34x_8 + 91x_9 + 107x_{10} + 117x_{11} +$$

$$3x_{12} + 39x_{13} + 12x_{14} + 60x_{15} + 87x_{16} + 90x_{17} + 54x_{18} + 2x_{19} + 22x_{20} + 99x_{21} +$$

$$46x_{22} + 17x_{23} + 5x_{24} + 27x_{25}$$

Subject to

$$105x_1 + 93x_2 + 92x_3 + 90x_4 + 74x_5 + 72x_6 + 72x_7 + 71x_8 + 70x_9 + 65x_{10} + 64x_{11} +$$

$$62x_{12} + 61x_{13} + 60x_{14} + 44x_{15} + 44x_{16} + 44x_{17} + 43x_{18} + 43x_{19} + 42x_{20} + 42x_{21} +.$$

$$41x_{22} + 41x_{23} + 40x_{24} + 40x_{25} \leq 379$$

$$x_j \in \{0, 1\}, j \in \{1, ..., 25\}.$$

The optimum solution to this problem is to select items $\{9, 10, 11, 15, 16, 17, 21\}$ leading to an objective benefit of 651 units while carrying 373 ounces.

2.4 Covers

As previously discussed, integer programs and their facet defining inequalities are fairly simple to determine in two dimensions. In higher dimensions, the difficulty greatly increases. Covers can assist with this challenge. A cover is a set of indices from a binary knapsack constraint when $x_j = 1 \ \forall \ j = 1, ..., n$ the inequality is infeasible. Covers are often used to find a valid starting inequality that can be strengthened by lifting. Thus, cover inequalities are very important in KP when looking for facet defining inequalities.

Formally, $C \subset N$ is a cover if and only if $\sum_{j \in C} a_j > b$. A minimal cover occurs when one indice is removed from the set and it is no longer a cover. Formally, $\sum_{j \in C \setminus \{k\}} a_j \leq b$ for each $k \in C$.

Every cover defines a cover inequality. These are always valid and follow the form $\sum_{j \in C} x_j \leq |C| - 1$. It is valid because the sum of all the coefficients is greater than the maximum allowed by the constraint. Therefore at least one item must be left at zero in each feasible solution.

In addition to minimal covers, there exist extended covers. They are defined as $E(C) = C \cup \{i \in N : a_i \geq a_j \text{ for all } j \in C\}$ and follow the form $\sum_{j \in E(C)} x_j \leq |C| - 1$. The example below depicts these concepts.

Example 2.3

Looking back at the KP in Example 2.1, the constraint is

$$105x_1 + 93x_2 + 92x_3 + 90x_4 + 74x_5 + 72x_6 + 72x_7 + 71x_8 + 70x_9 + 65x_{10} + 64x_{11} +$$

$$62x_{12} + 61x_{13} + 60x_{14} + 44x_{15} + 44x_{16} + 44x_{17} + 43x_{18} + 43x_{19} + 42x_{20} + 42x_{21} +.$$

$$41x_{22} + 41x_{23} + 40x_{24} + 40x_{25} \leq 379$$

Based on this constraint the set {15, 16, 17, 18, 19, 20, 21, 22, 23, 24} is a minimal cover represented by the inequality $x_{15}+x_{16}+x_{17}+x_{18}+x_{19}+x_{20}+x_{21}+x_{22}+x_{23}+x_{24} \leq 9$. An extended cover is {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24}. Its inequality is $\sum_{i=1}^{24} x_i \leq 9$.

2.5 Lifting

Lifting is a tool used in integer programming to create stronger inequalities. Gomory [20] first developed lifting to strengthen inequalities by increasing the dimension of the cutting plane. Lifting has been studied by a variety of sources including [3, 5, 6, 7, 8, 9, 12, 13, 14, 15, 18, 21, 22, 26, 29, 31, 33, 43, 44]. A more specific breakdown of references is provided in a later section.

Lifting takes a valid inequality of a restricted space and changes some of the coefficients and possibly the right hand side to make the inequality valid over the entire polyhedron. It is also used to find cutting planes that can be facet defining.

A restricted space is vital to lifting and takes a subset of variables, $E \subset N$, and forces

them to a fixed value, k_i for each $i \in E$. Formally let $P^{ch}_{E,K} = conv\{x \in P : x_i = k_i$ for all $i \in E\}$ where $k_i \in \mathbb{Z}$ and $K = (k_1, k_2, ..., k_{|E|})$. Thus, the restricted space only looks at a specific subset of variables, instead of the entire polyhedron.

Let $\sum_{i \in E} \alpha_i x_i + \sum_{i \in N \setminus E} \alpha_i x_i \leq \beta$ be a valid inequality of $P^{ch}_{E,K}$ where $E \subset N$. Lifting attempts to create a valid inequality of P^{ch} that takes the form $\sum_{i \in E} \alpha'_i x_i + \sum_{i \in N \setminus E} \alpha_i x_i \leq \beta'$.

There are multiple types of lifting that can be used: up, down and middle lifting, exact and approximate lifting, single and synchronized lifting, and sequential and simultaneous lifting. This leads to 24 different combinations. One example is exact single sequential uplifting.

2.5.1 Up, Down, and Middle Lifting

There are three ways of lifting in terms of assuming the initial coefficients for variables to be lifted. Uplifting is the most common lifting technique and assumes the initial $\alpha_i = 0$ for all $i \in E$, or $K = (0, 0, ..., 0)$. The opposite is down lifting which assumes that all k_i's are set to the upper bound of x_i for all $i \in E$. Finally there is middle lifting [41] where $0 < k_i < u_i \; \forall \; i \in E$. In this thesis, uplifting is used to create a new class of valid inequalities.

2.5.2 Exact and Approximate Lifting

Exact and approximate lifting differ in the known strength of the inequality. With exact lifting, the strongest possible inequality is formed. Thus, the goal is to increase α' and/or decrease β' as far as possible while maintaining validity. Typically an integer program is solved to determine these values. With approximate lifting, the α' and β' may be able to be increased or decreased to strengthen the inequality. Thus, most approximate lifting techniques do not need to solve an integer program but merely approximate the solution.

2.5.3 Single and Synchronized Lifting

Single and synchronized lifting refer to the number of inequalities generated by the lifting algorithm. Single lifting is the most commonly studied and generates a single inequality. Synchronized lifting creates many inequalities that take a particular form.

2.5.4 Sequential and Simultaneous Lifting

Sequential and simultaneous lifting differentiates between the number of variables lifted at a time. Sequential lifting requires the size of E to be one. Simultaneous lifting requires $|E| \geq 2$.

2.5.5 Prior Research on Lifting

With these nine broad classes of lifting, prior research methods can be classified according to the type of lifting. One of the most common lifting techniques is exact single sequential uplifting [41], which is described in the following example.

Example 2.4

A general method to perform exact single sequential uplifting of a binary variable, x_1, is as follows. Begin with $\sum_{j=2}^{n} \alpha_j x_j \leq \beta$ that is valid for $P^{ch}_{\{1\}}$. To determine α_1, solve the following integer program

$$z^{*IP} = \text{maximize } \sum_{j=2}^{n} \alpha_j x_j$$

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

$$x_1 = 1$$

$$x_1 \in \{0,1\}, x_2, ..., x_n \in \mathbb{Z}^n$$

and let $\alpha_1 = \beta - z^{*IP}$.

Returning to Example 2.1, consider the cover inequality $x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} + x_{22} + x_{23} + x_{24} \leq 9$. The following steps lift x_1. Solve

$$z^{*IP} = \text{maximize } x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} + x_{22} + x_{23} + x_{24}$$

$$\text{subject to } 105x_1 + 93x_2 + 92x_3 + 90x_4 + 74x_5 + 72x_6 + 72x_7 + 71x_8 + 70x_9 +$$

$$65x_{10} + 64x_{11} + 62x_{12} + 61x_{13} + 60x_{14} + 44x_{15} + 44x_{16} + 44x_{17} +$$

$$43x_{18} + 43x_{19} + 42x_{20} + 42x_{21} + 41x_{22} + 41x_{23} + 40x_{24} + 40x_{25} \leq 379$$

$$x_1 = 1$$

$$x_i \in \{0, 1\} \text{ for } i = 1, ..., 25$$

It is possible to let $x_1 = 1$ and include six other indices $x_{19}, x_{20}, x_{21}, x_{22}, x_{23}, x_{24}$ while remaining valid. Therefore $z^* = 6$. Since $\alpha_1 = \beta - z^*$, $\alpha_1 = 9 - 6 = 3$. The up lifted valid inequality is $3x_1 + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} + x_{22} + x_{23} + x_{24} \leq 9$.

Next, uplifting x_2 one obtains $z^* = 7$, and $\alpha_2 = 9 - 7 = 2$. The uplifted valid inequality becomes $3x_1 + 2x_2 + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} + x_{22} + x_{23} + x_{24} \leq 9$.

This continues until all variables are lifted. The results are summarized in Table 2.2.

Lifting Var	z^*	α
x_1	6	3
x_2	7	2
x_3	7	2
x_4	7	2
x_5	7	2
x_6	8	1
x_7	8	1
x_8	8	1
x_9	8	1
x_{10}	8	1
x_{11}	8	1
x_{12}	8	1
x_{13}	8	1
x_{14}	8	1
x_{25}	8	1

Table 2.2: Sequential Example

The final exact single sequentially up lifted inequality is $3x_1 + 2x_2 + 2x_3 + 2x_4 + 2x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} + x_{20} + x_{21} + x_{22} + x_{23} + x_{24} + x_{25} \leq 9$, which is facet defining as shown by the points in Table 2.3.

0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
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	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	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	0	0	0	0	1	1	1	1	1	1	1	1	1	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	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	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	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	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	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	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	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	0	0	0	0	1
0	1	1	1	1	1	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	1	1	1	1	1	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	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	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	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	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 2.3: Affinely independent points for exact single sequential uplifting example

The first nine affinely independent points are from the minimal cover inequality. The remaining 16 points are generated by the lifted variables, $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$, $x_{11}, x_{12}, x_{13}, x_{14}, x_{25}$. Each variable lifted into the inequality increases the dimension by one. As mentioned before, when x_1 is lifted in six other variables $x_{19}, x_{20}, x_{21}, x_{22}, x_{23}$,and x_{24} are set equal to one. This is shown in the 10^{th} point.

This is one simple example of exact lifting. Notice that an IP is solved each time a new variable is lifted. This can cause an increase in run time so the problem with lifting

takes longer to solve than the original problem. In 2010, Harris [25] formally introduced synchronized and single lifting as a new class of lifting. Therefore, the classification of prior researchers lifting techniques into this new lifting method should be thoroughly discussed.

A primary lifting result is due to Balas [5], who provides tight bounds for approximate synchronized sequential uplifting into a cover inequality from a knapsack constraint. Gutierrez developed a method to perform both exact single sequential and simultaneous uplifting for a generic IP, it requires solving a single branching tree.

The "sequence independent lifting" results of [4, 23, 36, 42] should be considered approximate simultaneous single uplifting. Balas and Zemel provided an exponential algorithm to perform exact synchronized simultaneous uplifting [6] into a cover inequality for a knapsack constraint. Recently there have been polynomial time algorithms developed to perform exact simultaneous single uplifting on the knapsack polytope [17, 35]. In 2009 Kubik [27] developed a pseudo-polynomial time algorithm for multiple simultaneously lifted sets into a valid inequality for P^{ch}_{KP}.

Bolton [10] introduced the original exact synchronized simultaneous uplifting method for the knapsack polyhedron. Realistically, her method could not lift into any inequality and her results should have been viewed as two set inequalities. Recently Harris expanded upon this idea and developed a three set lifting inequality in cubic time for the knapsack polyhedron.

There still exist numerous other papers that involve lifting, but the majority of these

papers are only applicable for certain problems. The reader should now be able to classify these lifting procedures. The next section describes in detail Bolton's exact synchronized simultaneous uplifting and helps to provide necessary background for this thesis.

2.5.6 Synchronized Simultaneous Lifting

The input for Bolton's algorithm is a knapsack constraint and two mutually exclusive lifting sets E_1 and E_2 and the inequality $0x \leq 0$. The outputs from this are valid inequalities of the form $\alpha_{E_1} \sum_{i \in E_1} x_i + \alpha_{E_2} \sum_{i \in E_2} x_i \leq 1$.

When applying Bolton's algorithm, the number of variables included from each set, E_1 and E_2, are organized as an ordered pair. If three variables were included from E_1 and seven from E_2, the point is $(3, 7)$ where E_1 is the x axis and E_2 is the y axis.

Once all feasible combinations of variables have been determined the slope, $\frac{\alpha_{E_2}}{\alpha_{E_1}}$, of the lines from the first point to all other points is found. The line that eliminates no other feasible points is selected. The next extreme point can be determined by the minimum slope. If a tie occurs, the point furthest along the line is selected. This extreme point is now identified and the slope of the lines to all subsequent points is found. This process is repeated until the final extreme point is located on the opposite axis. Returning to the previous problem, cutting planes are generated in the following example by exact synchronized simultaneous uplifting.

Example 2.5

Consider a portion of the KP from Example 2.2 given the constraint, $105x_1 + 93x_2 + 92x_3 + 90x_4 + 74x_5 + 72x_6 + 72x_7 + 71x_8 + 70x_9 \leq 379$. Let E_1 =$\{1,2,3,4\}$ and E_2=$\{5,6,7,8,9\}$. The output is valid inequalities of the form $\alpha_1 \sum_{i \in E_1} x_i + \alpha_2 \sum_{i \in E_2} x_i \leq \beta$.

First the feasible combinations are found and shown in Table 2.4. These are found in linear time using the process introduced by Easton and Hooker [17]. It begins by taking all elements in E_1 and decrementing until a feasible point is found. Then, the number of elements from E_2 are increased until the constraint becomes invalid which requires E_1 to then decrease.

E_1	E_2	F/I
4	0	I
3	0	F
3	1	F
3	2	I
2	2	F
2	3	I
1	3	F
1	4	F
1	5	I
0	5	F

Table 2.4: Candidate point list

With the possible points determined, the infeasible points can be eliminated and the α values can be calculated based on the first two feasible points, (3,0) and (3,1). Both α_1 and α_2 are found by solving the following system of equations

$\alpha_1(3) + \alpha_2(0) = 1$

$\alpha_1(3) + \alpha_2(1) = 1.$

This determines that $\alpha_1 = \frac{1}{3}$ and $\alpha_2 = 0$ The same process was repeated for all feasible points with the results in Table 2.5. Thus, the minimum α is clearly 0 between points $(3,0)$ and $(3,1)$. From there, the first set of α values are identified as $(\frac{1}{3},0)$ leading to the inequality $\frac{1}{3}\sum_{i\in E_1} x_i + 0\sum_{i\in E_2} x_i \leq 1$, also written as $\sum_{i\in E_1} x_i + 0\sum_{i\in E_2} x_i \leq 3$. Next, take the point, $(3,1)$, and conduct the same procedure.

E_1	E_2	α_1	α_2	$\frac{\alpha_2}{\alpha_1}$
3	0	-	-	-
3	1	$\frac{1}{3}$	0	0
2	2	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{2}$
1	3	$\frac{1}{3}$	$\frac{2}{9}$	$\frac{2}{3}$
1	4	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{2}$
0	5	$\frac{1}{3}$	$\frac{1}{5}$	$\frac{3}{5}$

Table 2.5: Alphas - 1

The minimum α fraction is $\frac{2}{3}$ at the point $(1,4)$ is seen in Table 2.6. This leads to the inequality $\frac{3}{11}\sum_{i\in E_1} x_i + \frac{2}{11}\sum_{i\in E_2} x_i \leq 1$. This can also be written as $3\sum_{i\in E_1} x_i + 2\sum_{i\in E_2} x_i \leq 11$.

Shown in Table 2.7, the same procedure is completed using $(1,4)$ as the anchor point to the final point $(5,0)$. The final inequality is $\sum_{i\in E_1} x_i + \sum_{i\in E_2} x_i \leq 5$.

This completes the exact synchronized simultaneous uplifting and generates three valid inequalities. Of these three inequalities, the first two are facet defining. For $\sum_{i\in E_1} x_i + 0\sum_{i\in E_2} x_i \leq 3$, the nine affinely points are shown in Table 2.8 and prove that

E_1	E_2	α_1	α_2	$\frac{\alpha_2}{\alpha_1}$
3	1	-	-	-
2	2	$\frac{1}{4}$	$\frac{1}{4}$	1
1	3	$\frac{1}{4}$	$\frac{1}{4}$	1
1	4	$\frac{3}{11}$	$\frac{2}{11}$	$\frac{2}{3}$
0	5	$\frac{4}{15}$	$\frac{1}{5}$	$\frac{3}{4}$

Table 2.6: Alphas - 2

E_1	E_2	α_1	α_2	$\frac{\alpha_2}{\alpha_1}$
1	4	-	-	-
0	5	$\frac{1}{5}$	$\frac{1}{5}$	1

Table 2.7: Alphas - 3

this inequality is facet defining. For $3\sum_{i\in E_1} x_i + 2\sum_{i\in E_2} x_i \leq 11$, the nine points shown in Table 2.9 are all affinely independent.

This illustrates that facet defining inequalities can be generated using exact synchronized simultaneous uplifting where $E = \emptyset$. Chapter 3 describes SSL and how it is applied to an arbitrary initial valid inequality.

0	1	1	1	0	0	0	0	0
1	0	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1
0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	1

Table 2.8: Affinely independent points for $\sum_{i \in E_1} x_i + 0 \sum_{i \in E_2} x_i \leq 3$

0	1	1	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0
1	1	0	1	0	0	0	0	0
1	1	1	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1
0	0	0	0	1	0	1	1	1
0	0	0	0	1	1	0	1	1
0	0	0	0	1	1	1	0	1
1	1	1	1	1	1	1	1	0

Table 2.9: Affinely independent points for $3 \sum_{i \in E_1} x_i + 2 \sum_{i \in E_2} x_i \leq 11$

Chapter 3

Synchronized Simultaneous uplifting over arbitrary initial inequalities for the knapsack polytope

This chapter presents the SSL algorithm developed in this research. SSL uplifts two sets into an arbitrary inequality simultaneously and generates multiple inequalities of the same form. Discussed in this chapter is an overview of SSL introducing the notation. This is followed by the pseudocode for SSL and the theoretical argument of correctness and facet defining conditions. Finally, an example illustrates SSL and shows that it creates a new class of facet defining inequalities.

3.1 SSL

The input for SSL is a knapsack constraint $\sum_{i=1}^{n} a_i x_i \leq b$, disjoint sets $E,\ E_1,\ E_2 \subset N$ and a valid inequality $\sum_{i \in E} \alpha_i x_i \leq \beta$ of $PKP^{ch}_{E_1 \cup E_2}$. Without loss of generality, $E_1 = \{i^1_1, i^1_2, ..., i^1_{e_1}\}$ and $E_2 = \{i^2_1, i^2_2, ..., i^2_{e_2}\}$ where $e_1 = |E_1|$ and $e_2 = |E_2|$. SSL outputs valid inequalities of the form $\sum_{i \in E} \alpha_i x_i + \sum_{i \in E_1} \alpha_1 x_i + \sum_{i \in E_2} \alpha_2 x_i \leq \beta$.

The goal for this algorithm is to generate multiple inequalities by uplifting two sets simultaneously into a given inequality. This process begins with the creation of the table to identify feasible set combinations. The table has a place for each possible value of b. Stored in the table, T_j, is the largest amount from the left hand side of the valid initial inequality that can be taken at that b_j value. For convenience define $T_j = -\infty$ if $j < 0$.

Feasible combinations from the sets are then found for each table value, T_j for $j =$ 0 to b. These j represent the weight of objects from each set that can be selected while remaining feasible. This step also finds Bolton's feasible points to the knapsack constraint with a right hand side of $b - j$. These points are called the candidate extreme points for j and denoted as EP_j. The value j is the third dimension of elements in EP_j. Thus a $q \in EP_j$ takes the form (e_{1_q}, e_{2_q}, j_q). Clearly, $j_q = j$ in this situation. Let the set of all candidate extreme points be $EP = \bigcup_{j=0}^{b} EP_j$. Next the points in EP are sorted by angle from smallest to largest where zero degrees occurs on the E_1 axis.

With the points generated and sorted, the α values are then calculated. This begins with all points forming a 0 angle, thus with $e_{2_q} = 0$. For each of these points, α_1 is calculated by taking the inequality right hand side, β, minus the table value and

dividing the difference by the number of objects taken from E_1, $\frac{\beta - T_{j_q}}{e_{1_q}}$. The point with the minimum α_1 is the most extreme point, it is now called p and selected as the first fixed point.

With this first extreme point determined, α_2 is calculated by solving a system of equations using the first point, q in EP where $e_{2_q} \neq 0$. This leads to the first test inequality, which is obtained by solving for α_1 and α_2 in the following two equations: $\alpha_1 e_{1_p} + \alpha_2 e_{2_p} = \beta - T_{j_p}$ and $\alpha_1 e_{1_q} + \alpha_2 e_{2_q} = \beta - T_{j_q}$.

Every point q' remaining in the sorted angle list is checked to see if the test inequality is valid (if $\alpha_1 e_{1_{q'}} + \alpha_2 e_{2_{q'}} < \beta - T_{j_{q'}}$). If this point does not strictly satisfy the inequality, then test point, q', takes q's place (a new inequality is found between point p and q' and the q is updated). This continues until the points reach the E_2 axis. At this point, the inequality with α_1 and α_2 are reported and can be used to generate a valid inequality.

Next q becomes the next fixed point, p, and the process continues. Inequalities will be generated until the fixed point reaches the E_2 axis. The result of SSL is multiple valid inequalities of the form $\sum_{i \in E} \alpha_i x_i + \sum_{i \in E_1} \alpha_1 x_i + \sum_{i \in E_2} \alpha_2 x_i \leq \beta$. SSL is formally defined by the pseudocode in the following section.

Synchronized Simultaneous Lifting Algorithm

Initialization

Sort E_1 and E_2 so that their knapsack coefficients are in descending order.

CreateTable Subroutine

For ($j = 0$ to b) begin

Find EP_j using the FeasiblePoint Subroutine

End For

Let $EP := \bigcup_{j=0}^{b} EP_j$ sorted from smallest to largest according to the angle.

Let $\alpha_1 := \infty$.

For Each ($q \in EP$) begin

If $e_{2_q} = 0$, then

If $\frac{\beta - T_{j_q}}{e_{1_q}} \leq \alpha_1$, then

$\alpha_1 := \frac{\beta - T_{j_q}}{e_{1_q}}$.

$pos := q$.

End If

End If

End For

Let $ext := pos$.

Let $pos := ext + 1$.

Let $p := pos$.

$\alpha_2 := \frac{1}{e_{1_q} * e_{2_p} - e_{2_q} * e_{1_p}} * [e_{1_q} * (\beta - T_{j_p}) - e_{1_p} * (\beta - T_{j_q})]$

End Initialization

Main Step

While (q's angle is less than 90 degrees) begin

For ($p = pos$ and $e_{1_p} \neq 0$)

If ($\alpha_1 e_{1_p} + \alpha_2 e_{2_p} > \beta - T_{j_p}$), then

$p := p + 1$.

$\alpha_1 := \frac{1}{e_{1_q} * e_{2_p} - e_{2_q} * e_{1_p}} * [e_{2_p} * (\beta - T_{j_q}) - e_{2_q} * (\beta - T_{j_p})]$

$\alpha_2 := \frac{1}{e_{1_q} * e_{2_p} - e_{2_q} * e_{1_p}} * [e_{1_q} * (\beta - T_{j_p}) - (e_{1_p} * (\beta - T_{j_q})]$

$pos := p$.

$p := p + 1$.

End For

$ext := pos$.

Report $\alpha_1 \sum_{i \in E_1} x_i + \alpha_2 \sum_{i \in E_2} x_i \leq \beta$ as a valid inequality.

End While

CreateTable Subroutine

Initialization

For ($j = 0$ to b)

$T_j := 0$.

End For

Main Step

For Each ($i \in E$) begin

For ($j = 0$ to b) begin

If $T_{b-j-a_i} + \alpha_i > T_{b-j}$, then

$T_{b-j} := T_{b-j-a_i} + \alpha_i$.

End For

End For Each

Feasible Point Subroutine for EP_j

Initialization

$sum := 0, p := 0, q := 0, count := 0$

While($sum \leq b - j$)

If ($p \leq |E_1| - 1$), then $sum := sum + a_{i^1_{e_1-p}}$ and $p := p + 1$.

Else $sum := sum + a_{e^2_{e_2-q}}$ and $q := q + 1$.

End While

If $q = 0$, then $p := p - 1$, $sum := sum - a_{i^1_{e_1-p}}$.

If $q = 1$, then $q := q - 1$, $sum := sum - a_{i^2_{e_2-q}}$.

If $q \geq 2$, then

$EP_j[count] := (p, 0)$ and $count := count + 1$.

$q := q - 1$ and $sum := sum - a_{i^2_{e_2-q}}$.

Main Step

$i := 0$

While ($p \geq 0$ and $q \leq e_2$)

If ($sum > b_i$, then $sum := sum - a_{i^1_{e_1-p+1}}$ and $p := p - 1$.

Else

$EP_j[count] := (p, q)$, $count := count + 1$

$sum := sum + a_{i^2_{e_2-q}}$ and $q := q + 1$.

End Else

$numpoints_i := count + 1$

End While

Return $feasE_1$, $feasE_2$ and $numpoints_i$.

With SSL fully detailed by the psuedo code, it is now necessary to prove its theoretical value. This is done through proof of validity, running time, and facet defining conditions.

Theorem 3.1.1 *Given a knapsack constraint* $\sum_{i \in N} a_i x_i \leq b$ *and a valid inequality* $\sum_{i \in E} \alpha_i x_i \leq \beta$ *of* $P^{ch}_{KP_{E_1 \cup E_2}}$, *then the SSL algorithm returns a valid inequality of* P_{KP} *of the form* $\sum_{i \in E} \alpha_i x_i + \sum_{i \in E_1} \alpha_1 x_i + \sum_{i \in E_2} \alpha_2 x_i \leq \beta$.

Proof: For contradiction, assume that there exists an $x \in P_{KP}$ such that $\sum_{i \in E} \alpha_i x_i + \sum_{i \in E_1} \alpha_1 x_i + \sum_{i \in E_2} \alpha_2 x_i > \beta$. Furthermore, let this x be the first such x in the sorted list. Now, let the SSL inequality be generated from fixed point $p = (e_{1_p}, e_{2_p}, j_p)$ and the next fixed point $q = (e_{1_q}, e_{2_q}, j_q)$. Let $j = \sum_{i \in E} a_i x_i \leq b$ due to the validity of $\sum_{i \in E} \alpha_i x_i \leq \beta$. Clearly, $T_j = \sum_{i \in E} \alpha_i x_i$ due to the obvious correctness of the dynamic programming table. Let $e_{1_x} = |\{i \in E_1 : x_i = 1\}|$ and $e_{2_x} = |\{i \in E_2 : x_i = 1\}|$.

Assume that x occurs in the sorted angle list after the fixed point that created the SSL inequality. Since $x \in P_{KP}$ and due to the sorted order of the knapsack constraint, EP_j contains (e_{1_x}, e_{2_x}, j_x) or a point that is larger in one or both of the first two dimensions. However, SSL checks to verify that each point in EP_j does not violate the inequality. Since this point is in EP_j, α_1 and α_2 would have changed to different values, a contradiction.

Assume that x occurs in the sorted angle list prior to the fixed point that created the SSL inequality. Thus, x, p and q are not on the same line and they create a triangle in the e_1 and e_2 plane. Since this is the first point that violates an inequality, x must be on the interior or line segment from p to the prior fixed point. But this violates the convexity of P^{ch}_{KP} and the result follows.

□

In addition to being valid, the inequalities produced by SSL require pseudopolynomial effort to generate. A large portion of this time comes in the initialization, rather than the main step.

Theorem 3.1.2 *Given a knapsack constraint* $\sum_{i \in N} a_i x_i \leq b$ *and a valid inequality* $\sum_{i \in E} \alpha_i x_i \leq \beta$ *of* $P^{ch}_{KP_{E_1 \cup E_2}}$, *then the SSL requires* $O(nb + n^3)$.

Proof: Consider SSL step by step. First in the initialization, the E_1 and E_2 sets are sorted by their a coefficients. This requires $O(nlog(n))$ effort using bisection sort. Next the table is created. For each element in E all b elements in the table are evaluated, which requires $O(nb)$ effort. It requires linear, $O(nb)$, effort to find EP_j using the

FeasiblePoints subroutine. There are at most $e_1 * e_2$ feasible points. By compacting the table, there only exist at most $O(n^2)$ points that are not dominated by either a higher j value or a higher e_{p_1} or e_{p_2}. Such a compaction would require $O(nb)$. The angles can then be sorted and obtained in $O(n^2 log(n))$. The final step in initialization, to find the first fixed point, which requires $O(n)$ as there can only be e_1 points on the axis.

Each inequality that the main step evaluates requires examining at most each point in this compacted list. Thus, each inequality requires $O(n^2)$. There can be at most $O(e_1 + e_2)$ inequalities. Thus, the main step requires $O(n^3)$. Reporting is $O(n^2)$. Thus, SSL requires $O(nb + n^3)$.

□

With the running time determined, it is then useful to show that SSL can produce facet defining inequalities. First define ξ_i to be the identity point in the i^{th} dimension.

Any inequality generated from SSL is created from two points. It is a simple task to modify this algorithm to record these two points, which are now called (e_{1_p}, e_{2_p}, j_p) and (e_{1_q}, e_{2_q}, j_q). For brevity, SSL does not force the points that generated the inequalities in any particular order. Thus, if one has a p and q that does not meet the following theorem. The reader should reorder the p and q to see if the opposite order satisfies the conditions. It is also known that points beyond $i^1_{e_1}$ or $i^2_{e_2}$ do not exist.

Theorem 3.1.3 *Given a knapsack constraint $\sum_{i \in N} a_i x_i \leq b$ and a valid inequality $\sum_{i \in E} \alpha_i x_i \leq \beta \neq 0$ that is facet defining for $P^{ch}_{KP_{E_1 \cup E_2}}$ and the inequality $\sum_{i \in E} \alpha_i x_i + \sum_{i \in E_1} \alpha_1 x_i + \sum_{i \in E_2} \alpha_2 x_i \leq \beta$ from the SSL algorithm that is generated from the two fixed*

points p and q corresponding to the table. If any of the following conditions are met:

i) If $e_{1_p} = e_1 - 1$ *and* $e_{2_q} = e_2 - 1$, *then* $\sum_{i \in \{i^1_1, \ldots, i^1_{e_1-1}, i^2_{e_2-e_{2_p}+1}, \ldots, i^2_{e_2}\}} a_i \leq b - j^1$ *and* $\sum_{i \in \{i^1_{e_1-e_{1_q}+1}, \ldots, i^1_{e_1}, i^2_1, \ldots, i^2_{e_2-1}\}} a_i \leq b - j^2$,

ii) If $1 \leq e_{1_p} \leq e_1 - 2$ *and* $e_{2_q} = e_2 - 1$, *then* $\sum_{i \in \{i^1_1, i^1_{e_1-e_{1_p}+2}, \ldots, i^1_{e_1}, i^2_{e_2-e_{2_p}+1}, \ldots, i^2_{e_2}\}} a_i \leq b - j^1$, $\sum_{i \in \{i^1_{e_1-e_{1_p}}, \ldots, i^1_{e_1-1}, i^2_{e_2-e_{2_p}+1}, \ldots, i^2_{e_2}\}} a_i \leq b - j^1$ *and* $\sum_{i \in \{i^1_{e_1-e_{1_q}+1}, \ldots, i^1_{e_1}, i^2_1, \ldots, i^2_{e_2-1}\}} a_i \leq b - j^2$,

iii) If $e_{1_p} = e_1 - 1$ *and* $1 \leq e_{2_q} \leq e_2 - 2$, *then* $\sum_{i \in \{i^1_1, \ldots, i^1_{e_1-1}, i^2_{e_2-e_{2_p}+1}, \ldots, i^2_{e_2}\}} a_i \leq b - j^1$, $\sum_{i \in \{i^1_{e_1-e_{1_q}+1}, \ldots, i^1_{e_1}, i^2_1, i^2_{e_2-e_{2_q}+2}, \ldots, i^2_{e_2}\}} a_i \leq b - j^2$ *and* $\sum_{i \in \{i^1_{e_1-e_{1_q}+1}, \ldots, i^1_{e_1}, i^2_{e_2-e_{2_q}}, \ldots, i^2_{e_2-1}\}} a_i \leq b - j^2$,

iv) If $1 \leq e_{1_p} \leq e_1 - 2$ *and* $1 \leq e_{2_q} \leq e_2 - 2$, *then* $\sum_{i \in \{i^1_1, i^1_{e_1-e_{1_p}+2}, \ldots, i^1_{e_1}, i^2_{e_2-e_{2_p}+1}, \ldots, i^2_{e_2}\}} a_i \leq b - j^1$, $\sum_{i \in \{i^1_{e_1-e_{1_p}}, \ldots, i^1_{e_1-1}, i^2_{e_2-e_{2_p}+1}, \ldots, i^2_{e_2}\}} a_i \leq b - j^1$, $\sum_{i \in \{i^1_{e_1-e_{1_q}+1}, \ldots, i^1_{e_1}, i^2_1, i^2_{e_2-e_{2_q}+2}, \ldots, i^2_{e_2}\}} a_i \leq b - j^2$ *and* $\sum_{i \in \{i^1_{e_1-e_{1_q}+1}, \ldots, i^1_{e_1}, i^2_{e_2-e_{2_q}}, \ldots, i^2_{e_2-1}\}} a_i \leq b - j^2$,

then the SSL inequality is facet defining for P^{ch}_{KP}.

Proof: Cases i), ii) and iii) are a subset of case iv); therefore, it suffices to prove this case. The reader can easily eliminate portions of this proof to obtain the other three cases.

Given a knapsack constraint $\sum_{i \in N} a_i x_i \leq b$, a valid inequality $\sum_{i \in E} \alpha_i x_i \leq \beta \neq 0$ that is facet defining for $P^{ch}_{KP_{E_1 \cup E_2}}$ and the inequality $\sum_{i \in E} \alpha_i x_i + \sum_{i \in E_1} \alpha_1 x_i + \sum_{i \in E_2} \alpha_2 x_i \leq \beta$ generated from the SSL algorithm, assume that all of the necessary assumptions are true for case iv).

By Theorem 3.1.1, the SSL inequality is valid. Furthermore, the origin does not meet the SSL inequality at equality, thus the face of the SSL inequality has dimension at most $n-1$. Now it suffices to find $n = |E| + |E_1| + |E_2|$ affinely independent points.

Due to assumption, $\sum_{i \in \{i^1_1, i^1_{e_1 - e_{1_p}+2}, \ldots, i^1_{e_1}, i^2_{e_2 - e_{2_p}+1}, \ldots, i^2_{e_2}\}} a_i \leq b - j^1$. Therefore, the points $\sum_{i \in \{i^1_{e_1 - e_{1_p}+2}, \ldots, i^1_{e_1}, i^2_{e_2 - e_{2_p}+1}, \ldots, i^2_{e_2}\}} \xi_i + \xi_l$ are feasible in the knapsack constraint with b replaced by $b - j_1$ for all $l = \{i^1_1, \ldots, i^1_{e_1 - e_{1_p}-1}\}$ due to the sorted order of the a coefficients in the knapsack constraint. Furthermore, by assumption $\sum_{i \in \{i^1_{e_1 - e_{1_p}}, \ldots, i^1_{e_1-1}, i^2_{e_2 - e_{2_p}+1}, \ldots, i^2_{e_2}\}} a_i \leq b - j^1$, the points $\sum_{i \in \{i^1_{e_1 - e_{1_p}}, \ldots, i^1_{e_1}, i^2_{e_2 - e_{2_p}+1}, \ldots, i^2_{e_2}\}} \xi_i - \xi_l$ are feasible in the knapsack constraint with b replaced by $b - j_1$ for each $l \in \{e_2 - e_{2_p}, \ldots, e_2\}$ due to the sorted order of a.

Since each of these e_1 points requires at most $b - j_1$ units from the knapsack constraint, there exists a point containing only $x_i = 1$ with $i \in E$ such that $\sum_{i \in E} \alpha_i x_i = T_{j_1}$ and this point requires at most j_1 units from the knapsack constraint. To each of the previously mentioned e_1 points add this point. Call these points Q.

Each point in Q is clearly feasible as the sum of the respective a coefficients is less than or equal to $b - j_1 + j_1 = b$. Furthermore, each point has exactly e_{1_p} variables equal to one with indices in E_1 and e_{2_p} variables equal to one with indices in E_2. Evaluating any point in Q in the SSL inequality therefore results in $T_{j_1} + \alpha_1 e_{1_p} + \alpha_2 e_{2_p} = \beta$ due to the method that SSL generates α_1 and α_2. Thus, each point in Q is in the SSL inequality's face.

To create e_2 more points, observe that $\sum_{i \in \{i^1_{e_1 - e_{1_q}+1}, \ldots, i^1_{e_1}, i^2_1, i^2_{e_2 - e_{2_q}+2}, \ldots, i^2_{e_2}\}} a_i \leq b - j^2$ due

to assumption. Therefore, the points $\sum_{i\in\{i^1_{e_1-e_{1_q}+1},\ldots,i^1_{e_1},i^2_{e_2-e_{2_q}+2},\ldots,i^2_{e_2}\}}\xi_i+\xi_l$ are feasible in the knapsack constraint with b replaced by $b-j_2$ for all $l=\{i^2_1,\ldots,i^2_{e_2-e_{2_q}-1}\}$ due to the sorted order of the a coefficients in the knapsack constraint. Furthermore, by assumption $\sum_{i\in\{i^1_{e_1-e_{1_q}+1},\ldots,i^1_{e_1},i^2_{e_2-e_{2_q}},\ldots,i^2_{e_2-1}\}}a_i\leq b-j^2$, the points $\sum_{i\in\{i^1_{e_1-e_{1_q}+1},\ldots,i^1_{e_1},i^2_{e_2-e_{2_q}},\ldots,i^2_{e_2}\}}\xi_i-\xi_l$ are feasible in the knapsack constraint with b replaced by $b-j_2$ for each $l\in\{e_2-e_{2_q},\ldots,e_2\}$ due to the sorted order of a.

Since each of these e_2 points requires at most $b-j_2$ units from the knapsack constraint, there exists a point containing only $x_i=1$ with $i\in E$ such that $\sum_{i\in E}\alpha_i x_i=T_{j_2}$ and this point requires at most j_2 units from the knapsack constraint. To each of the previously mentioned e_2 points include this point. Call these points Q'.

Each point in Q' is clearly feasible as the sum of the respective a coefficients is less than or equal to $b-j_2+j_2=b$. Furthermore, each point has exactly e_{1_q} variables equal to 1 with indices in E_1 and e_{2_q} variables equal to 1 with indices in E_2. Evaluating any point in Q' in the SSL inequality therefore results in $T_{j_2}+\alpha_1 e_{1_q}+\alpha_2 e_{2_q}=\beta$ due to the method that SSL generates α_1 and α_2. Thus, each point in Q' is in the SSL inequality's face.

Since $\sum_{i\in E}\alpha_i x_i\leq\beta\neq 0$ that is facet defining for $P^{ch}_{KP_{E_1\cup E_2}}$, there exist $|E|$ affinely independent points in P_{KP} that satisfy $\sum_{i\in E}\alpha_i x_i+\sum_{i\in E_1}\alpha_1 x_i+\sum_{i\in E_2}\alpha_2 x_i=\beta$ with the property that $x_i=0$ for all $i\in E_1\cup E_2$. Call these points Q''.

Consider the matrix obtained by including the points from Q, Q' and Q''. There are $|E|+|E_1|+|E_2|=n$ points and each point meets the SSL inequality at equality. The

points in Q'' are clearly affinely independent from the points in Q and Q' due to the block diagonal construction. The points in Q', Q'' are clearly affinely independent from the other points in Q', Q'', respectively, as shown in the sequential lifting example in Chapter 2. Since SSL obtained α_1 and α_2, the point with (e_{1_p}, e_{2_p}) and (e_{1_q}, e_{2_q}) are also affinely independent. Thus, these points are n affinely independent points and the SSL inequality is facet defining.

□

With the detailed procedure for SSL described and the theoretical strength shown, it is useful to demonstrate SSL with an example. The following section will illustrate the key aspects in executing SSL and determining the facet defining conditions.

3.2 SSL Examples

The following examples will demonstrate SSL, the facet defining conditions, and show that is creates a new class of inequalities.

Example 3.1

Reconsider the example from Chapter 2 with the knapsack constraint

$$105x_1 + 93x_2 + 92x_3 + 90x_4 + 74x_5 + 72x_6 + 72x_7 + 71x_8 + 70x_9 + 65x_{10} + 64x_{11} +$$

$$62x_{12} + 61x_{13} + 60x_{14} + 44x_{15} + 44x_{16} + 44x_{17} + 43x_{18} + 43x_{19} + 42x_{20} + 42x_{21} +.$$

$$41x_{22} + 41x_{23} + 40x_{24} + 40x_{25} \leq 379.$$

The inequality, $3x_1 + 3x_2 + 3x_3 + 3x_4 + 2x_5 + 2x_6 + 2x_7 + 2x_8 + 2x_9 \leq 11$, generated

in Example 2.4 is the initial valid inequality. Also, let set $E_1 = \{10, 11, 12, 13, 14\}$ and $E_2 = \{15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25\}$.

With this information, a table is generated to find the maximum left hand side of the valid inequality that can be included for each possible b. The table for this example is condensed to show only the critical numbers, where $T_j \neq T_{j+1}$ for $j = 0, ..., b$. Table generation begins by creating the array T which contains spaces for all values from 0 to b. All spaces initially contain a zero. They are then updated using dynamic programming. The table is created in iterations, based on the elements in E. Starting at the end for a given $i \in E$, the dynamic programming formula is, if T_j is less than $T_{j-a_i} + \alpha_i$, then $T_j := T_{j-\alpha_i} + \alpha_i$ for $j = b, ..., 0$. This is done for each i in E.

For this example, begin at T_{379} and consider x_1 where a_1 is 105 and α_1 is 3. Initially both T_{379} and T_{274} are zero. Since T_{379} is less than $0 + 3$ it is assigned a value of 3. This continues for all T_j. After considering x_1 the table created is shown in Table 3.1. For simplicity, the range of j with the same T_j have been condensed.

j	0-104	105-379
T_j	0	3

Table 3.1: Table including only x_1

This same process is completed for x_2 then all indices in the initial valid inequality. The final table, Table 3.2, contains the maximum left hand side of the valid inequality that can be included at each table location.

The points are then generated using a portion the Easton and Hooker's algorithm [17]

j	0-69	70-89	90-140	141-159	160-181	182-230
T_j	0	2	3	4	5	6
j	231-251	252-274	275-322	323-344	345-379	
T_j	7	8	9	10	11	

Table 3.2: Table for example

where the right hand side of the inequality is evaluated as $\beta - T_j$. For each "dominate table" location, the extreme feasible point candidates are listed in Table 3.3.

j	0	70	90	141	160	182	231	252	275	323	345
T_j	0	2	3	4	5	6	7	8	9	10	11
	5 0	4 0	4 0	3 0	3 0	3 0	2 0	2 0	1 0	0 1	0 0
	5 1	4 1	4 1	3 1	2 2	2 1	1 2	1 1	1 1		
	4 3	3 3	3 2	2 2	1 3	1 3	0 3	0 3	0 2		
	3 4	2 4	2 4	1 4	0 5	0 4					
	2 6	1 6	1 5	0 5							
	1 7	0 7	0 7								
	0 9										

Table 3.3: Initial generated Table

For simplicity, non-extreme points are eliminated using Bolton's algorithm. This is not stated in the algorithm and is unnecessary. However it does reduce a few steps in the example. The remaining points are in Table 3.4 and illustrated graphically in Figure 3.1.

The next step generates EP by combining EP_j for all j in the table. The angle for each point is calculated. The sorted angle list for the dominate table locations is shown in Table 3.5.

The initial candidate point lies on the E_1 axis. The most extreme point has the minimum α_1 value. For point q where $e_{2_q} = 0$, let α_{1_q} be $\frac{11-T_{j_q}}{e_{1_q}}$. This calculation is

j	0	70	90	141	160	182	231	252	275	323	345
T_j	0	2	3	4	5	6	7	8	9	10	11
	5 0	4 0	4 0	3 0	3 0	3 0	2 0	2 0	1 0	0 1	0 0
	5 1	4 1	4 1	3 1	2 2	2 1	1 2	0 3	1 1		
	4 3	3 3	2 4	1 4	0 5	1 3	0 3		0 2		
	3 4	1 6	0 7	0 5		0 4					
	1 7	0 7									
	0 9										

Table 3.4: Generated Table after Bolton's

shown for all possible candidate points in Table 3.6. The minimum α_1 value is 1.5 from point $(2, 0, 8)$, as calculated by $\frac{11-8}{2}$. This point becomes the initial candidate point, or the fixed point, is $e_{1_8} = 2$, $e_{2_8} = 0$, and $j_8 = 8$.

The next step is to calculate an initial α_1 and α_2 from the candidate point, q, and the first point off the E_1 axis, at $p = 10$, $(5, 1, 0)$. These values are found by solving this system of equations.

$2\alpha_1 + 0\alpha_2 = (11 - 8)$

$5\alpha_1 + \alpha_2 = (11 - 0)$

This yields $\alpha_1 = \frac{3}{2}$ and $\alpha_2 = \frac{7}{2}$. All points with larger angles are then tested for validity in the equation $\frac{3}{2}\sum_{i\in E_1} x_i + \frac{7}{2}\sum_{i\in E_2} x_i \leq 11 - T_j$. Applying the next point (4,1,2) results in $\frac{3}{2}(4) + \frac{7}{2}(1) \leq 11 - 2$, which simplifies to $9.5 > 9$ and the test fails. Thus, (4,1,2) is combined with (2, 0, 8) (the fixed point) to create new α values. The following equations are solved,

$2\alpha_1 + 0\alpha_2 = (11 - 8)$

$4\alpha_1 + \alpha_2 = (11 - 9)$.

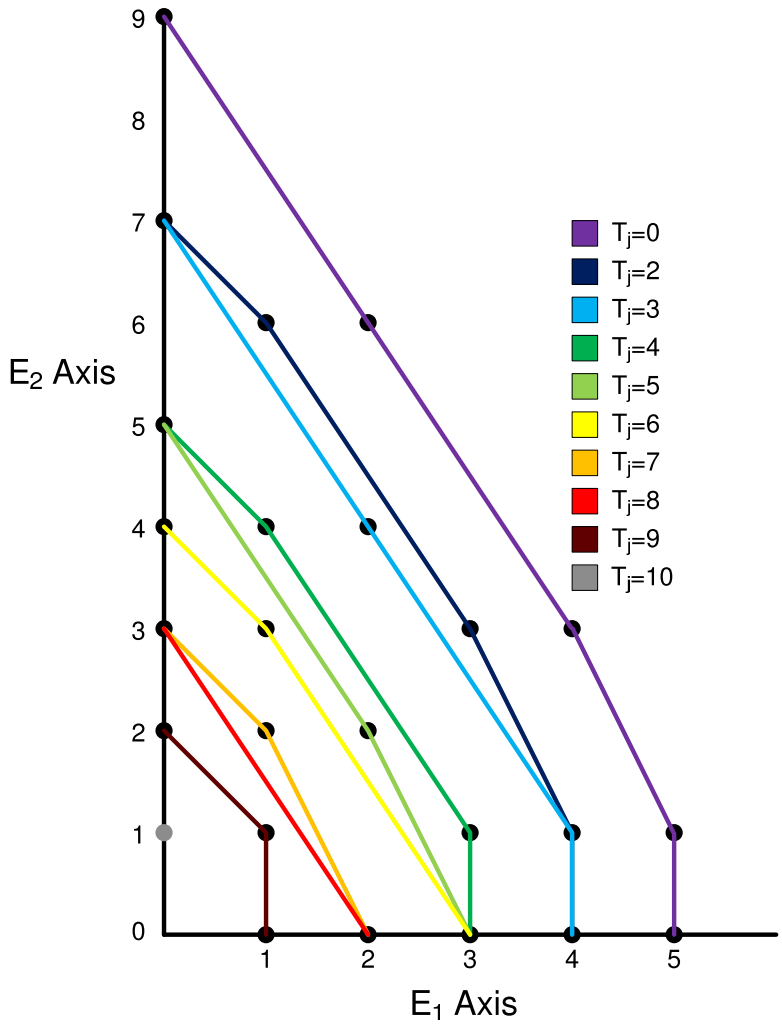

Figure 3.1: Graphic of Points

Solving the above equations generates the new test inequality $\frac{3}{2}\sum_{i\in E_1} x_i + 3\sum_{i\in E_2} x_i \leq 11 - T_j$. The next point (4,1,3) fails. Generating the next α_1 and α_2 provides the new test inequality $\frac{3}{2}\sum_{i\in E_1} x_i + 2\sum_{i\in E_2} x_i \leq 11 - T_j$. Several points satisfy this inequality, but point 14 (3,2,3) fails. This leads to a new inequality, which fails at (4,3,0). The next test inequality is $\frac{3}{2}\sum_{i\in E_1} x_i + \frac{5}{3}\sum_{i\in E_2} x_i \leq 11$ which fails at point (3,3,2).

This continues until the point (1,1,9) forms the equation $\frac{3}{2}\sum_{i\in E_1} x_i + \frac{1}{2}\sum_{i\in E_2} x_i \leq 11 - T_j$. All remaining points satisfy this equation. Thus, $3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i +$

Pt#	E_1	E_2	T_j	Angle	Pt#	E_1	E_2	T_j	Angle
1	5	0	0	0	18	2	4	3	63.43
2	4	0	2	0	19	1	2	7	63.43
3	4	0	3	0	20	1	3	6	71.57
4	3	0	4	0	21	1	4	4	75.96
5	3	0	5	0	22	1	6	2	80.54
6	3	0	6	0	23	1	7	0	81.87
7	2	0	7	0	24	0	9	0	90
8	**2**	**0**	**8**	**0**	25	0	7	2	90
9	1	0	9	0	26	0	7	3	90
10	5	1	0	11.31	27	0	5	4	90
11	4	1	2	14.04	28	0	5	5	90
12	4	1	3	14.04	29	0	4	6	90
13	3	1	4	18.43	30	0	3	7	90
14	3	3	2	45	31	0	3	8	90
15	2	2	5	45	32	0	2	9	90
16	1	1	9	45	33	0	1	10	90
17	3	4	0	53.13	34	0	0	11	DNE

Table 3.5: Ordered EP by Angle

$\frac{3}{2}\sum_{i=10}^{1} 4x_i + \frac{1}{2}\sum_{i=15}^{25} x_i \leq 11$ is a valid inequality and the point (1,1,9) is considered the next fixed point.

The new fixed point, (1,1,9), continues the process with the next point in line (3,4,0). The points that fail and the α values that they form with (1,1,9) are shown in Table 3.7. The inequality created with points (1,1,9) and (0,3,8) generates an inequality that is satisfied by all points and $3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i + \sum_{i=10}^{14} x_i + \sum_{i=15}^{25} x_i \leq 11$ is valid. Furthermore, E_2 axis has been reached and SSL terminates.

For this problem and inequality, SSL generates two inequalities,

$3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i + \frac{3}{2}\sum_{i=10}^{14} x_i + \frac{1}{2}\sum_{i=15}^{25} x_i \leq 11$ and

$3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i + \sum_{i=10}^{14} x_i + \sum_{i=15}^{25} x_i \leq 11.$

Pt#	E_1	E_2	T_j	α_1
1	5	0	0	2.20
2	4	0	2	2.25
3	4	0	3	2.00
4	3	0	4	2.33
5	3	0	5	2.00
6	3	0	6	1.67
7	2	0	7	2.00
8	**2**	**0**	**8**	**1.50**
9	1	0	9	2.00

Table 3.6: Initial extreme points

These inequality are valid by Theorem 3.1.1. Furthermore, these inequalities satisfy Theorem 3.1.3 and so they are facet defining inequalities. For $3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i + \frac{3}{2}\sum_{i=10}^{14} x_i + \frac{1}{2}\sum_{i=15}^{25} x_i \leq 11$ the 25 points that are generated from this theorem are presented in Table 3.8. The first nine points come from the initial valid inequality $3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i \leq 11$. For convenience, lines section together the sets E, E_1, and E_2 across the top, and the indices in the sets along the side. The two points that create this SSL inequality are point $8 = p$, $(2, 0, 8)$, and point $16 = q$, $(1, 1, 9)$. Since $1 \leq e_{1_p} = 2 \leq e_1 - 2 = 3$ and $1 \leq e_{2_q} = 1 \leq e_2 - 2 = 9$, case iv) must be verified.

To verify the first condition, observe that $a_{10} + a_{14} = 65 + 60 = 125 \leq 379 - 252$ where the 252 is the smallest j value where $T_j = 8$. This then enables the points in 10^{th} and 11^{th} columns to be feasible in Table 3.8. To verify the second condition, observe that $a_{12} + a_{13} = 62 + 61 = 123 \leq 379 - 252$. This then enables the points in 12^{th}, 13^{th} and 14^{th} columns to be feasible in Table 3.8. It is trivial to verify that these points satisfy the inequality at equality. The points for x_1 to x_9 contribute 8 to the left hand side of the inequality. Since $\alpha_1 = \frac{3}{2}$ and there are always 2 variables between x_{10} to x_{14}

E_1	E_2	Layer	α_1	α_2
1	1	9		
3	4	0	-3	5
2	4	2	$-\frac{1}{2}$	$-\frac{5}{2}$
2	4	3	0	2
2	6	0	$\frac{1}{4}$	$\frac{7}{4}$
1	3	6	$\frac{1}{2}$	$\frac{3}{2}$
1	6	2	$\frac{3}{5}$	$\frac{7}{5}$
0	9	0	$\frac{7}{9}$	$\frac{11}{9}$
0	7	3	$\frac{6}{7}$	$\frac{8}{7}$
0	3	8	1	1

Table 3.7: Trial α values with candidate point (1,1,9)

set to one.

To verify the third condition, observe that $a_{14} + a_{15} = 60 + 44 = 104 \leq 379 - 275$ where the 275 is the smallest j value where $T_j = 9$. This then enables the points in 15^{th} to the 25^{th} columns to be feasible in Table 3.8. Observe that the fourth condition does not exist due to the fact that $e_{2_q} = 1$. It is trivial to verify that these points satisfy the inequality at equality. The points for x_1 to x_9 contribute 9 to the left hand side of the inequality. Since $x_{14} = 1$, another $\frac{3}{2}$ is contributed. Finally, there is always 1 variables between x_{15} to x_{25} set to one, which contributes and additional $\frac{1}{2}$.

For $3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i + \sum_{i=10}^{14} x_i + \sum_{i=15}^{25} x_i \leq 11$ the 25 points in Table 3.9 meet the inequality at equality and are affinely independent. As with the first inequality, the

x_i	E									E_1					E_2										
1	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
2	1	1	1	1	1	0	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
10	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
11	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
12	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
13	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
14	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
15	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
16	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
17	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
18	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
19	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
20	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
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 3.8: Affinely independent points for $3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i + \frac{3}{2}\sum_{i=10}^{14} x_i + \frac{1}{2}\sum_{i=15}^{25} x_i \leq 11$

first nine points come from the initial valid inequality, $3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i \leq 11$, and the remaining 16 points were generated using Theorem 3.1.3. The fixed points used to generate the inequality are $16 = p$, $(1, 1, 9)$ and $31 = q$, $(0, 3, 8)$. Since $1 \leq e_{1_p} = 1 \leq e_1 - 2 = 3$ and $1 \leq e_{2_q} = 3 \leq e_2 - 2 = 9$, case iv) must be verified.

To verify the first condition, observe that $a_{10} + a_{25} = 65 + 40 = 105 \leq 379 - 275$. This then enables the points in 10^{th} through 14^{th} columns to be feasible in Table 3.9. The

second conditions does not exist due to fact that $e_{1_p} = 1$. To verify that these points satisfy the inequality at equality, the points for x_1 to x_9 contribute 9 to the left hand side of the inequality, and there are always three variables between x_{15} and x_{25} equal to one which contributes a total of three.

x_i	E									E_1					E_2										
1	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
2	1	1	1	1	1	0	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
3	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
10	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
11	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
12	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
13	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
14	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
15	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
16	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
17	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
18	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
19	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
20	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
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1
25	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 3.9: Affinely independent points for $3\sum_{i=1}^{4} x_i + 2\sum_{i=5}^{9} x_i + \sum_{i=10}^{14} x_i + \sum_{i=15}^{25} x_i \leq 11$

To verify the third condition, observe that $a_{15} + a_{24} + a_{25} = 44 + 40 + 40 = 124 \leq 379 - 252$. This allows the 15^{th} through the 21^{st} points to be feasible. To verify the final condition, $a_{22} + a_{23} + a_{24} = 122$ which allows the 22^{th} through 25^{th} points to be

clearly feasible. These points meet the inequality at equality. The points for x_1 to x_9 contribute 8 to the left hand side of the inequality, there is always a variable between x_{10} and x_{14} equal to one which contributes one, and finally x_{25} contributes one.

One of the most exciting results is that these SSL inequalities are a new class of facet defining inequalities for the knapsack polytope. To show that this new class of inequalities cannot be generated by other lifting techniques, consider the following example.

Example 3.2

Let the knapsack constraint be

$148x_1 + 148x_2 + 74x_3 + 74x_4 + 74x_5 + 19x_6 + 19x_7 + 19x_8 + 19x_9 + 13x_{10} + 13x_{11} + 13x_{12} + 13x_{13} + 13x_{14} \leq 222.$

Executing Bolton's algorithm [10] with

$E_1 = \{1, 2\}$ and $E_2 = \{3, 4, 5\}$

provides three feasible extreme points (1,0), (1,1) and (0,3). This provides the inequalities

$x_1 + x_2 \leq 1$ and

$\frac{2}{3}x_1 + \frac{2}{3}x_2 + \frac{1}{3}x_3 + \frac{1}{3}x_4 + \frac{1}{3}x_5 \leq 1.$

This second inequality is far more interesting and is facet-defining over the reduced space and so it is used for this example. To reduce the amount of fractional calculations, consider the equivalent valid inequality $2x_1 + 2x_2 + x_3 + x_4 + x_5 \leq 3$.

To this initial inequality, apply SSL with sets

$E_1 = \{6, 7, 8, 9\}$ and

$E_2 = \{10, 11, 12, 13, 14\}$.

The dynamic programming table with the feasible extreme points is shown in Table 3.10. The angles associated with the points are then calculated and the list is sorted, seen in Table 3.11.

j	0	74	148	222
T_j	0	1	2	3
	4 0	4 0	3 0	0 0
	4 5	4 5	3 1	
	0 5	0 5	2 2	
			1 4	
			0 5	

Table 3.10: Dynamic programming table and feasible extreme points

Pt#	E_1	E_2	T_j	Angle
1	4	0	0	0
2	4	0	1	0
3	3	0	2	0
4	3	1	2	18.43
5	2	2	2	45
6	4	5	0	51.34
7	4	5	1	51.34
8	1	4	2	75.96
9	0	5	0	90
10	0	5	1	90
11	0	5	2	90

Table 3.11: Ordered EP by Angle

The minimum α_1 occurs at point (3,0,2) with $\alpha_1 = \frac{1}{3}$. Following SSL, the points (3,0,2) and (3,1,2) form the first lifted inequality, with $\alpha_1 = \frac{1}{3}$ and $\alpha_2 = 0$. This inequality is not violated by any other points, so it is valid and (3,1,2) becomes the next

fixed point.

From (3,1,2), the point (4,5,1) creates the inequality with $\alpha_1 = \frac{3}{11}$ and $\alpha_2 = 211$, that is not violated by any of the other feasible points. However, (1,4,2) meets this inequality at equality and it becomes the next fixed point. The final inequality uses (1,4,2) and (0,5,1) and assigns $\alpha_1 = \frac{1}{5}$ and $\alpha_2 = \frac{1}{5}$. Thus, SSL creates the following three inequalities.

$2\sum_{i=1}^{2} x_i + \sum_{i=3}^{5} x_i + \frac{1}{3}\sum_{i=6}^{9} x_i + 0\sum_{i=10}^{14} x_i \leq 3$

$2\sum_{i=1}^{2} x_i + \sum_{i=3}^{5} x_i + \frac{3}{11}\sum_{i=6}^{9} x_i + \frac{2}{11}\sum_{i=10}^{14} x_i \leq 3$ and

$2\sum_{i=1}^{2} x_i + \sum_{i=3}^{5} x_i + \frac{1}{5}\sum_{i=6}^{9} x_i + \frac{1}{5}\sum_{i=10}^{14} x_i \leq 3.$

Consider the second inequality, $2\sum_{i=1}^{2} x_i + \sum_{i=3}^{5} x_i + \frac{3}{11}\sum_{i=6}^{9} x_i + \frac{2}{11}\sum_{i=10}^{14} x_i \leq 3$. First observe that this inequality is not based upon a cover inequality. Therefore, none of the results on lifting covers can find this inequality. Furthermore, it has four distinct coefficients, which set it apart from both Bolton's [10] and Harris's [25] algorithms. Applying sequential lifting, will generate integer coefficients, so it generate this inequality either. Applying simultaneous lifting E_1 [24] results in a coefficient of $\alpha_1 = \frac{1}{3}$ and then $\alpha_2 = 0$. In the opposite order, simultaneous lifting generates an $\alpha_2 = \frac{1}{5}$ and $\alpha_1 = \frac{1}{5}$. Consequently, SSL generates previously undiscovered valid inequalities of the P_{KP}^{ch}.

To show that SSL generates undiscovered facet defining inequalities, it is necessary to determine the affinely independent points utilizing Theorem 3.1.3. Since $e_{1_p} = 3 = e_1 - 1 = 4 - 1$ and $e_{2_q} = 4 = e_2 - 1 = 5 - 1$, the conditions for facet defining are outlined by case i). The affinely independent points are shown in Table 3.12.

x_i	E					E_1				E_2				
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	1	1	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	1	0	0	0	0	0	0	0	0	0
4	0	0	1	0	1	1	1	1	1	1	1	1	1	1
5	1	0	0	1	1	1	1	1	1	1	1	1	1	1
6	0	0	0	0	0	0	1	1	1	0	0	0	0	0
7	0	0	0	0	0	1	0	1	1	0	0	0	0	0
8	0	0	0	0	0	1	1	0	1	0	0	0	0	0
9	0	0	0	0	0	1	1	1	0	1	1	1	1	1
10	0	0	0	0	0	0	0	0	0	0	1	1	1	1
11	0	0	0	0	0	0	0	0	0	1	0	1	1	1
12	0	0	0	0	0	0	0	0	0	1	1	0	1	1
13	0	0	0	0	0	0	0	0	0	1	1	1	0	1
14	0	0	0	0	0	1	1	1	1	1	1	1	1	0

Table 3.12: Affinely independent points for $2\sum_{i=1}^{2} x_i + \sum_{i=3}^{5} x_i + \frac{3}{11}\sum_{i=6}^{9} x_i + \frac{2}{11}\sum_{i=10}^{14} x_i \leq 3$

The first five points are facet defining by the original lifting inequality. To verify the first condition, observe that $a_6 + a_7 + a_8 + a_{14} = 19 + 19 + 19 + 13 = 71 \leq 222 - T_2 = 74$. This allows the 6^{th} through 9^{th} points to be feasible. To verify the second condition observe that $a_9 + a_{10} + a_{11} + a_{12} + a_{13} = 19 + 13 + 13 + 13 + 13 = 71 \leq 222 - T_2 = 74$. This allows the 10^{th} through 14^{th} points to be feasible.

It is clear to see that these points meet the lifted inequality at equality and are affinely independent. Thus, SSL can generate previously undiscovered facet defining inequalities. Furthermore, if more variables are included in this problem than any of the lifted facet defining inequalities off of this new inequality would also be previously undiscovered classes.

Chapter 4

Computational Results

One contribution of this research is in the application of exact synchronized simultaneous uplifting over an arbitrary initial valid inequality and the development of an algorithm to iteratively create valid inequalities of this new class. This section describes the quantitative advancements of SSL through its implementation into commercial integer programming software, CPLEX[38]. These results show that SSL inequalities allow CPLEX to solve problems that it could not solve before due to insufficient memory. Additionally, SSL reduces the time to solve some IPs that CPLEX alone could also solve.

The results of this computational study were obtained through the use of an Intel (R) Core i7 computer with a 1.58 GHz processor and 3.0 GB of RAM. A commercial optimization software, IBM's CPLEX 10.0, was used to compare the results with and without the SSL cuts. All results are reported in seconds. The SSL implementation is created in C++.

An appropriate class of problems has to solve within a general time interval. Problems solving too quickly are unlikely to show either positive or negative effects for SSL.

Additionally, problems requiring days to solve are also impractical for a small computational study.

This study is performed on randomly generated multiple knapsack problems (MKP). An MKP is a KP with more than one constraint and is represented as maximize $\sum_{i \in N} c_i x_i$ subject to $\sum_{i \in N} a_{ij} x_i \leq b_j$ for $j = 1, ..., r$ and $x_i \in \{0, 1\}$ for all $i \in N$ where r is the number of rows and $a_{ij} \geq 0$ for all $i \in N$ and $j \in \{1, ..., r\}$.

The constraint coefficients, a_{ij}, are random integers taken from a uniform distribution, $\{0, ..., 1000\}$ for $i \in \{1, ..., r\}$ and $j \in \{1, ..., n\}$. Each objective coefficient, c_{ij}, is calculated by adding the column coefficients and a uniform random integer between 0 and 100, $c_{ij} = a_{1j} + ... + a_{rj} + u$ where u is a uniformly distributed integer between 0 and 100. The right hand side of each constraint is one fifth of the sum of the row's coefficients rounded down to the nearest integer, $\lfloor \sum_{j=1}^{n} \frac{a_{ij}}{5} \rfloor = b_i$ for $i = 1, ..., r$.

The computational study was conducted using n variables, with n equal to 50, 60, and 70. The number of rows varies such that r equals three and four. To ensure randomness and avoid anomalies, 20 instances of each size problem are generated and the averages reported.

For this study, the initial valid inequality is $0 \leq 0$. The first iteration of SSL creates several unique inequalities. Each of these inequalities are then considered an initial valid inequality for SSL, each of which generated more inequalities. Each of those were all considered a new initial valid inequality and continued SSL iterations until all variables were included.

A key component of SSL is the quality of the sets E_1 and E_2. For the computational study, the initial sets were selected based on the variables with a reduced cost greater than -10. These random indices were sorted based on a and split into two sets at the largest gap in between coefficients, where E_1 contains the variables with the largest a coefficients.

After creating the initial valid inequalities, the sets were selected based on size. The number of indices in the problem but not in the inequality were determined. This number was divided by four and determined to be the number of variables selected for each set. The variables in E_1 were the largest variables based on a and E_2 were the smallest. Thus, of the remaining points not in the inequality, the quarter of points with the largest a value were placed in E_1 and the quarter with the smallest in E_2.

To reduce the number of points that must be generated, SSL simplifies the table into layers. A layer exists for every $T_j \neq T_{j+1}$. This has no effect on the validity of the results because only the most "dominate" T_j is selected.

All inequalities generated by SSL were lifted into the inequality. This includes inequalities generated in an early iterations of SSL that will have more variables lifted in before SSL terminates.

The results from the study are summarized in the tables below. Table 4.1 details the number of problems that were solved with and without the addition of SSL cuts. There are three problem classes where SSL implementation allowed more problems to be solved to optimality than CPLEX alone. No instances where encountered where CPLEX

solved and CPLEX with SSL did not. Of these 120 problems CPLEX solved, 79 while implementing SSL and 61 without. This amounts to a 29.5% increase in the number of problems solved to optimality.

	$n = 50$			$n = 60$			$n = 70$		
	SSL	CPLEX	Δ	SSL	CPLEX	Δ	SSL	CPLEX	Δ
$r = 3$	20	20	0	20	20	0	17	14	3
$r = 4$	20	6	14	1	0	1	1	1	0

Table 4.1: Number of problems solved with SSL versus CPLEX

In situations where the problem failed, to solve the number of nodes and time required were recorded and summarized in Table 4.2 below. When CPLEX failed to optimally solve the problem, the average node size was about 13 million nodes, but the minimum and maximum vary greatly. Additionally, the time before CPLEX declared that the problem could not solve to optimality took between just over a minute to more than 25 minutes.

	Nodes	Time
Average	9,705,000	523.26
Minimum	1,370,000	81.00
Maximum	34,460,000	1553.00

Table 4.2: Nodes and time to fail to solve to optimality

To be effective, SSL inequalities must be generated and applied quickly. The average preprocessing times, are shown in Table 4.3. It never requires more than one second for SSL to execute. On average, it clearly requires less than quarter of a second. Also in Table 4.3 are the average number of SSL cuts generated and added to the problem. There were at most 17 cuts with a minimum of 9. Thus SSL generates many cuts quickly.

	$n = 50$		$n = 60$		$n = 70$	
	cuts	time	cuts	time	cuts	time
$r = 3$	12.65	0.15	12.0	0.20	11.44	0.11
$r = 4$	13.45	0.20	13.2	0.03	14.55	0.01

Table 4.3: Preprocessing time for SSL

When CPLEX solves a problem to optimality both with and without SSL implementation, the time and percent difference recorded and shown in Table 4.4. All times are averages in terms of seconds where a positive percent change represents SSL solving faster than CPLEX.

	$n = 50$			$n = 60$			$n = 70$		
	SSL	CPLEX	%Δ	SSL	CPLEX	%Δ	SSL	CPLEX	%Δ
$r = 3$	185.00	269.15	31.27%	615.80	605.5	-1.7%	670.40	548.4	-22.25%
$r = 4$	1001.50	880.33	-13.76%	DNE	DNE	DNE	1529.00	1103.00	-39.62%

Table 4.4: Time to solve with SSL versus CPLEX

While the time to solve with SSL inequalities is numerically larger, on average, Table 4.5 below shows p-values that resulted from the statistical analysis. A paired t-test compared the time to solve with an α value of 0.05. Of the four situations where there was enough data to compare, three found no significant difference between means time to solve with and without SSL. The only situation with significant difference, signified by a p-value$= 0.01 < \alpha = 0.05$, is when $n = 50$ and $r = 3$ which is the problem class that SSL solved faster than CPLEX. Thus for this instance, SSL solved statistically significantly faster than CPLEX.

The research motivation for this thesis began with Bolton's [10] algorithm. Table

	$n = 50$	$n = 60$	$n = 70$
$r = 3$	0.01	0.866	0.463
$r = 4$	0.552	DNE	DNE

Table 4.5: P-values from paired t-test on time to solve

4.6 shows a comparison between Bolton's method and SSL. SSL was able to solve 5.33% more problems to optimality than Bolton's algorithm. Thus, synchronized simultaneous uplifting to Bolton's inequalities is beneficial.

	$n = 50$		$n = 60$		$n = 70$	
	SSL	Bolton	SSL	Bolton	SSL	Bolton
$r = 3$	20	20	20	18	17	11
$r = 4$	20	20	1	5	1	1

Table 4.6: Problems solved SSL vs. Bolton

Overall, adding SSL inequalities provided a significant improvement over traditional CPLEX resulting in approximately one-third more problems solved to optimality. This computational study shows that SSL inequalities are fast to generate, do not require significantly more time to solve, and enable CPLEX to solve problems that it could not before. Further computational studies are encouraged to understand the precise computational benefits of this thesis and which sets should be used.

Chapter 5

Conclusion

The goal of this thesis was to develop a method to perform exact synchronized simultaneous uplifting of two sets into an arbitrary valid initial inequality. Another objective is to generate cutting planes that perform better than traditional CPLEX. The SSL algorithm presented in this thesis achieves these goals. Furthermore, the inequalities generated by SSL form a new class of valid and at times facet defining inequalities.

SSL requires $O(nb+n^3)$ effort. The inequalities generated are valid and when specific conditions are met, outlined in Theorem 3.1.3, they are facet defining. This is illustrated by an example which demonstrates all critical aspects of the algorithm. After executing SSL, two inequalities are generated. Both of these inequalities are shown to be facet defining.

The computational study presented in Chapter 4 shows that adding SSL inequalities to CPLEX enabled the software to solve 29.5% more problems to optimality where it was unable to before. No instances were encountered that CPLEX could solve while CPLEX with SSL cuts would not. SSL inequalities are created very quickly, requiring less than a second to generate.

5.1 Future Research

At present, exact synchronized simultaneous uplifting into an arbitrary inequality only allows for two sets. Harris [25] developed an algorithm to exact synchronized simultaneous up lift three sets into the zero inequality. There is opportunity for research which allows three or more sets to be lifted into an arbitrary initial valid inequality.

Additionally, SSL is currently restricted to uplifting. There are possible applications in down and middle lifting. SSL is also limited to a single constraint KP. Applications to multiple knapsack instances are still undiscovered.

Beyond the knapsack problem, there are areas of study in different types of problems. Extending the SSL algorithm to handle node packing or edge covering problems is a new application area. Other applications include mixed integer programs and non-binary instances. Further research exists in the extensions of these synchronized simultaneous lifting concepts for general integer programs with negative constraints.

This thesis provided a small computational study. Further research into set selection and variety of problem classes is necessary. Additionally, a larger sample size would allow for greater statistical analysis of the strength and quality of SSL application as compared to CPLEX.

Overall, opportunities for research within polyhedral theory and combinatorial optimization are vast, and only a few are discussed in this chapter. When considering lifting techniques, the goal is to achieve facet defining valid inequality. Unfortunately, a

common challenge of researchers is to find these inequalities in a reasonable amount of time. Thus, there are extensive areas for future development.

Bibliography

[1] Anbil R., E. Gelman , B. Patty , and R. Tanga (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, 170-82.

[3] Atamtürk, A. (2003). "On the facets of the mixed-integer knapsack polyhedron," *Mathematical Programming*, **98** (1-3), 145-175.

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

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

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

[7] Balas, E. and E. Zemel (1984). "Lifting and complementing yields all the facets of positive zero-one programming polytopes," *Mathematical Programming, Proceed-*

ings of the International Conference on Mathematical Programming, R.W. Cottle et al., eds., 13-24.

[8] Balas, E. and S. Ng (1989). "On the set covering polytope. II, Lifting the facets with coefficients in 0,1,2," *Mathematical Programming*, **45** (1), 1-20.

[9] Balas, E. and M. Fishetti (1993). "Lifting procedure for the asymmetric traveling salesman polytope and a large new class of facets," *Mathematical Programming*, **58** (3), 325-352.

[10] Bolton, J. (2009). "Synchronized simultaneous lifting in binary knapsack polyhedra," *MS Thesis*, Department of Industrial and Manufacturing Systems Engineering, Kansas State University.

[11] 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.

[12] Cho C., D. Padberg, and M. Rao (1983). "On the uncapacitated plant location problem. II. Facets and lifting theorems," *Mathematics of Operations Research*, **8** (4), 590-612.

[13] Dahan, X., M. Maza, E. Schost. W. Wu, and Y. Xie (2005). "Lifting techniques for triangular decompositions," *Proceedings of the 2005 International Symposium on Symbolic and Algebraic Computation ISSA '05.*

[14] De Farias Jr, I., E. Johnson, and G. Nemhouser (2002). "Facets of the complimentarity knapsack polytope," *Mathematics of Operations Research*, **27** (1), 210-227.

[15] De Simone, C., (1990). "Lifting facets of the cut polytope," *Operations Research Letters* **9** (5), 341-344.

[16] 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),100-109.

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

[18] Felici, G., and C. Gentile (2003). "Zero-lifting for integer blocks structured problems," *Journal of Combinatorial Optimization*,**7** (2), 161-167.

[19] Ferreira, C., C. de Souza and Y. Wakabayashi (2002). "Rearrangement of DNA fragments: A branch-and-cut algorithm," *Discrete Applied Mathematics*, **116**, (1-2), 161-177.

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

[21] Gu, Z., G. Nemhauser, and M. Savelsbergh (1998). "Lifted cover inequalities for 0-1 integer programs: computation," *INFORMS Journal on Computing*, **10** (4), 427-437.

[22] Gu, Z., G. Nemhauser, and M. Savelsbergh (1999). "Lifted cover inequalities for 0-1 integer programs: complexity," *INFORMS Journal on Computing*, **11** (1), 117-123.

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

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

[25] Harris, A. (2010) "Generating an original cutting-plane algorithm in three sets (GO CATS)," *MS Thesis*, Department of Industrial and Manufacturing Systems Engineering, Kansas State University.

[26] Koster, A., S. Hoesel, and A. Kolen (1998). "The partial constraint satisfaction problem: facets and lifting," *Operations Research Letters*, **23** (3), 89-98.

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

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

[29] Nemhauser, G. and P. Vance (1994). "Lifted cover facets of the 0-1 knapsack polytope with GUB constraints," *Operations Research Letters*,**16** (5), 255-263.

[30] Pajunas, A., E. Matto, M. Trick, and L. Zuluaga (2007). "Optimizing Highway Transportation at the United States Postal Service," *Interfaces*, **37**(6), 515525.

[31] Park, K. (1997). "Lifting cover inequalities for the precedence-constrained knapsack problem," *Discrete Applied Mathematics*, **72** (3), 219-241.

[32] Pisinger,D. (1995) "Algorithms for knapsack problems," *Ph.D. Thesis*, Department of Computer Science, University of Copenhagen.

[33] Richard J., I. De Farias, and G. Nemhauser (2003). "Lifted inequalities for 0-1 mixed integer programming: Superlinear lifting," *Integer Programming*, **98** (1-3), 115-143.

[34] 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), 593-606.

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

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

[37] Subramanian, R., R. Scheff , J. Quillinan , S. Wiper, and R. Marsten (1994). "Coldstart: fleet assignment at delta air lines." *INTERFACES*, **24**, 104-120.

[38] The CPLEX Solver on ILOG's Home Page, http://www.ilog.com/.

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

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

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

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

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

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