SIMULTANEOUSLY LIFTING SETS OF VARIABLES IN BINARY KNAPSACK PROBLEMS

by

KAMANA SHARMA

M.Tech, NITC, India, 2004

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Industrial and Manufacturing Systems Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2007

Approved by:

Major Professor
Dr. Todd Easton

Abstract

Integer programming (IP) has been and continues to be widely used by industries to minimize cost and effectively manage resources. Faster computers and innovative IP techniques have enabled the solution to many large-scale IPs. However, IPs are NP-hard and many IPs require exponential time to solve.

Lifting is one of the most widely used techniques that helps to reduce computational time and is widely applied in today's commercial IP software. Lifting was first introduced by Gomory for bounded integer programs and a theoretical and computationally intractible technique to simultaneously lift sets of variables was introduced by Zemel in 1978.

This thesis presents a new algorithm called the Maximal Simultaneous Lifting Algorithm (MSLA), to simultaneously uplift sets of binary integer variables into a cover inequality. These lifted inequalities result in strong inequalities that are facet defining under fairly moderate assumptions.

A computational study shows that this algorithm can find numerous strong inequalities for random Knapsack (KP) instances. The pre-processing time observed for these instances is less than $\frac{1}{50}^{th}$ of a second, which is negligible. These simultaneously lifted inequalities are easy to find and incorporating these cuts to KP instances reduced the solution time by an average of 41%. Therefore, implementing MSLA should be highly beneficial for large real-world problems.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Integer programming (IP) has been and continues to be used by industries to minimize cost and effectively manage resources. Faster computers and innovative integer programming techniques has enabled the solution to many large-scale integer programs. As a result, integer programs are being used more and more to obtain quality solutions to practical problems.

For instance, $100 million per year is saved by Delta Airlines by using an integer programming model for their fleet assignments. This IP provides assignments to over 450 airplanes and more than 2500 domestic flights every day. Similarly, American Airlines have an annual savings of over $20 million by using an integer program to solve a crew scheduling problem on a monthly basis.

Integer programs are typically used in business and economic situations, but they have also been applied to varieties of other problems. Some industries that frequently use integer programming models include: transportation [16, 17, 69, 74], energy [60, 63, 73], telecommunications [16, 50], and manufacturing [1, 28, 46].

This thesis focuses on a classical IP, called the Knapsack Problem (KP). The name knapsack draws an analogy with the problem faced by a camper who needs to fill her knapsack with objects while being restricted by the overall weight. Among the n objects

the camper can select each object has an associated benefit that the camper gets for taking the object and an associated positive weight. The camper is unable to carry more than a maximum overall weight. Therefore, a choice has to be made between the n objects such that she gets the maximum benefit and obeys the overall weight constraint.

A single constraint from a integer program can be transformed to a knapsack constraint and therefore, most theoretical results for a KP can be extended to IP. Hence, the knapsack problem is widely studied for its theoretical benefits [4, 6, 24, 26, 51, 58, 62, 77]. Additionally, the knapsack problem has been widely studied for its numerous applications. Some common applications include resource allocation [12, 75], cutting stock [55, 29] and capital budgeting [12, 54, 57].

1.1 Integer Programming

Integer programs are optimization problems, which enforce integer constraints on all of the variables. IPs try to get the best outcome (e.g. maximum profit) subject to list of constraints (e.g. budget constraints) using a linear mathematical model. IP problems have the general form:

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{Subject to} \quad & Ax \leq b \\ & x \in \mathbf{Z^n_+}. \end{aligned}$$

where $A \in \mathbf{R^{m \times n}}$, $c \in \mathbf{R^{1 \times n}}$ and $b \in \mathbf{R^{m \times 1}}$. The expression to be maximized or minimized is called the objective function, which is $c^T x$ in this case. The constraints are inequalities given by $Ax \leq b$ and restrict the solution space and the constraint $x \in Z^n_+$ restricts the value of x_i to positive integers.

If only some of the unknown variables are required to be integers, then the problem is called a mixed integer programming (MIP) problem [2, 3, 14, 40, 44, 58, 64]. Traditionally, IPs have linear objective functions and much work done in this area by [14, 15, 22, 31, 52, 53]. In this thesis we concentrate on IPs with linear objective functions and constraints, but the

results can be easily extended to IPs with non-linear objective functions.

1.2 Solving Integer Programs

In contrast to linear programming, which can be solved efficiently in the worst case, integer programming problems are NP-hard [47]. Many integer programs require exponential time to solve and thus, the optimal solutions for many practical problems still cannot be obtained.

The simplest of algorithms to solve an IP is enumeration. In this methodology all the IP solutions for a problem are enumerated and the one that has the best value for the objective function is selected. Consider a IP problem with 50 binary decision variables then the number of possible solutions is $2^{50} = 1.1 * 10^{15}$, which is a very large number and can take years to solve. As the number of decision variables increase the number of possible solutions also become exponentially high. Therefore, enumeration is computationally too expensive to solve large IPs [59].

Due to the drawback in the enumeration technique there have been many algorithms for solving integer linear programs. The two most frequently used ones are branch and bound and cutting-planes, which are discussed in the following sections.

1.2.1 Branch and Bound

Branch and Bound was first proposed by A. H. Land and A. G. Doig [49] in 1960 and is the most widely used method to solve integer programs. Branch and Bound is also an enumeration process, but this method differs from the enumeration method described above in that not all the feasible solutions are typically enumerated.

In simple terms, branch and bound can be outlined as an enumeration tree of linear programs in which each problem has the same constraints and objective except for some additional bounds on certain components of the decision variables x_i. Consider the problem $max\ c^T x$, subject to $Ax \leq b$, $x \in \mathbf{R}^{\mathbf{n}}_{+}$. At the root of the branch and bound tree is the

original problem without the integer constraint $x \in \mathbf{Z}_+^{\mathbf{n}}$ i.e. the linear relaxation (LR) given by $max\ c^T x$, subject to $Ax \leq b$, $x \geq 0$. The solution $Z^{LR\,*}$ to this root problem may not have all integer components and hence, $x^{LR\,*}$ represents the optimal solution of the linear relaxation.

If $x^{LR\,*}$ is not integer then two new child nodes are created by finding an i such that $p < x_i < p+1$ and $p \in \mathbf{Z}_+$. The first child node has the same linear relaxation as its parent node plus the additional constraint $x_i \leq p$; while the second child node has the additional constraint $x_i \geq p+1$. This branching process can be carried out recursively and each of the two new problems may give rise to two more problems when branching is performed on one of the non-integer components of their solutions. It can be seen that the enumeration tree obtained in branch and bound is binary. This branching process continues until all leaf nodes are fathomed. A node can be fathomed if the solution to the LR is integer, infeasible, or if $Z^{LR\,*} < Z^{IP}$ where Z^{IP} is the best integer solution found so far. Once all of the leaf nodes are fathomed, the algorithm returns the best integer solution or that the problem is infeasible.

The efficiency of branch and bound depends critically on the effectiveness of the branching method used. A bad choices could lead to repeated branching without any fathoming. In this case, the method would be similar to enumeration. There are different methods to search the branching tree with the most common ones being depth first search, breadth first search and best child node. The depth first search searches along one branch till a node is fathomed and then back tracks to the parent node and goes down the other branch. The breadth first search goes down the branching tree one level at a time. The best child node always picks the child node with the best $Z^{LR\,*}$.

There are advantages and disadvantages for all the above mentioned methods. Most common IP softwares use best child search strategy as default settings. Though this strategy is good it may still require exponential time to solve the problem. There is no search method that works best for all problems.

1.2.2 Cutting Plane

The cutting plane method was introduced by Gomory [32, 33, 34]. This technique aims at generating valid inequalities or cuts which eliminate some portion of the linear relaxation without cutting off any feasible solutions. A *valid inequality* for an IP is an inequality, $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$, that satisfies every feasible point. The cuts generated by this method are added to the linear relaxation as new constraints and are useful if they eliminate critical portions of the linear relaxation.

The cutting plane method is widely used as it can reduce the computational time for solving the problem. A lot of research has been done in this area some of which are [10, 13, 21, 32, 33, 34, 35, 37, 38]. Not all cuts developed are important or useful. The strongest valid inequalities generated are facet defining inequalities. A more detailed discussion on this topic is presented in Chapter 2.

Branch and cut is typically the most commonly used method to solve integer programs. This is a hybrid of branch and bound method and the cutting plane method discussed above. Branch and cut method first solves the linear relaxation of the IP to obtain the optimal solution $Z^{LR\,*}$ and $x^{LR\,*}$. Cutting plane algorithms are then used to find a few cutting planes. This process is repeated until either an integer solution is found or until no more cutting planes are found. Finally, if some of the variables $x^{LR\,*}$ are still not integer then branch and bound is performed.

1.2.3 Lifting

Lifting is one of the most widely used techniques to generate strong cutting planes. Lifting was first introduced by Gomory [36] and has been an area of research ever since. Lifting is a useful technique used to strengthen valid inequalities. Lifting starts with a valid inequality and makes it stronger by introducing more variables or changing the coefficients of existing indices. In certain instances the resulting lifted inequality could be facet defining inequalities. These lifted inequalities can be used as a cutting plane and should help to reduce the

computational time for the IP.

There are numerous different types of lifting. They includes up, down and middle lifting. Each of these types of lifting can be performed sequentially and probably simultaneously. This thesis deals mainly with simultaneous lifting and a more detailed discussion is presented in the next few chapters.

1.3 Research Motivations

The technique of simultaneous lifting sets of integer variables was introduced by Zemel [76] in 1978. This method was restricted to binary variables and furthermore, his method was based on finding the extreme points from the solutions of exponentially many IPs and found all simultaneously lifted facets. A few years ago, Easton and Hooker [26] improved this result and found a polynomial time algorithm to simultaneously lift a single set of binary variables into a cover inequality for a KP. However, their research left the open question of how to choose the lifting set. The motivation for this research is to determine the appropriate simultaneous lifting sets.

1.4 Research Contributions

This thesis presents a new algorithm, MSLA, to simultaneously uplift sets of binary variables in a KP instance. These uplifted inequalities result in cutting planes for KP. Furthermore, these inequalities result in strong inequalities that are facet defining under fairly moderate assumptions.

An additional advancement of this thesis is that this algorithm runs in $O(|E||C| + |C|log|C| + |E|log|E|)$ effort. This running time is theoretically faster than iteratively applying the result from Easton and Hooker [26].

A computational study shows that this algorithm can find numerous strong inequalities

for random KP instances. These simultaneously lifted inequalities are easy to find and by including these cuts to a KP instance, the solution time decreased by an average of 41%. Therefore, I postulate that implementing MSLA would be highly beneficial for large real-world problems.

1.5 Thesis Outline

The basic fundamentals of integer programming that are required to understand this thesis are presented in Chapter 2. A brief introduction to the knapsack problem, covers and cover inequalities, is given. Also, this chapter discusses some established lifting techniques.

Chapter 3 gives a formal presentation of the simultaneous lifting technique proposed in this thesis. It gives the details of the new maximal simultaneous lifting algorithm and discusses the main theoretical benefits. The chapter walks through an example to help illustrate the algorithm and its critical steps and benefits.

A computational study is performed to clearly establish the benefits of the algorithm proposed in this thesis. Chapter 4 presents the results of this study that strongly show the computational efficiency of this algorithm. Finally, Chapter 5 contains the concluding remarks. A few challenging questions are raised that could be future research topics.

Chapter 2

Background Information

For a better understanding of the core concepts of this research, a brief introduction of the various fundamentals of integer programming is presented in this chapter. Consider the integer program (IP), maximize $c^T x$, subject to $Ax \leq b$, $x \in \mathbf{Z}^{\mathbf{n}}_{+}$ where $A \in \mathbf{R}$ is an $m \times n$ matrix and b is an $m \times 1$ matrix . The feasible region, P, is the set of integer points that satisfy all the constraints, $P = \{x \in \mathbf{Z}^{\mathbf{n}}_{+} | Ax \leq b\}$. Let $N = \{1, 2, 3, ..., n\}$, represent the indices.

Integer programming problems are NP-hard [47] in general. So all known algorithms require exponential time to solve. Thus, many practical IP problems have unknown optimal solutions since they are currently not solvable even on today's fastest machines.

There have been many techniques used to solve integer programs. These techniques fundamentally rely on the *linear relaxation* of the integer program. The linear relaxation takes the form $max\ c^T x$, subject to $Ax \leq b$, $x \geq 0$, which is the integer program without the integer constraint. Let P^{LR} represent the feasible region for the linear relaxation, $P^{LR} = \{x \in \mathbf{R}^{\mathbf{n}}_{+} | Ax \leq b\}$. One of the most widely used methods to solve integer programs is branch and bound, which is a glorified enumeration algorithm and is discussed in Section 1.2.1. Branch and bound has proved an effective tool to solve integer programs. However, this method may solve exponentially many linear relaxations.

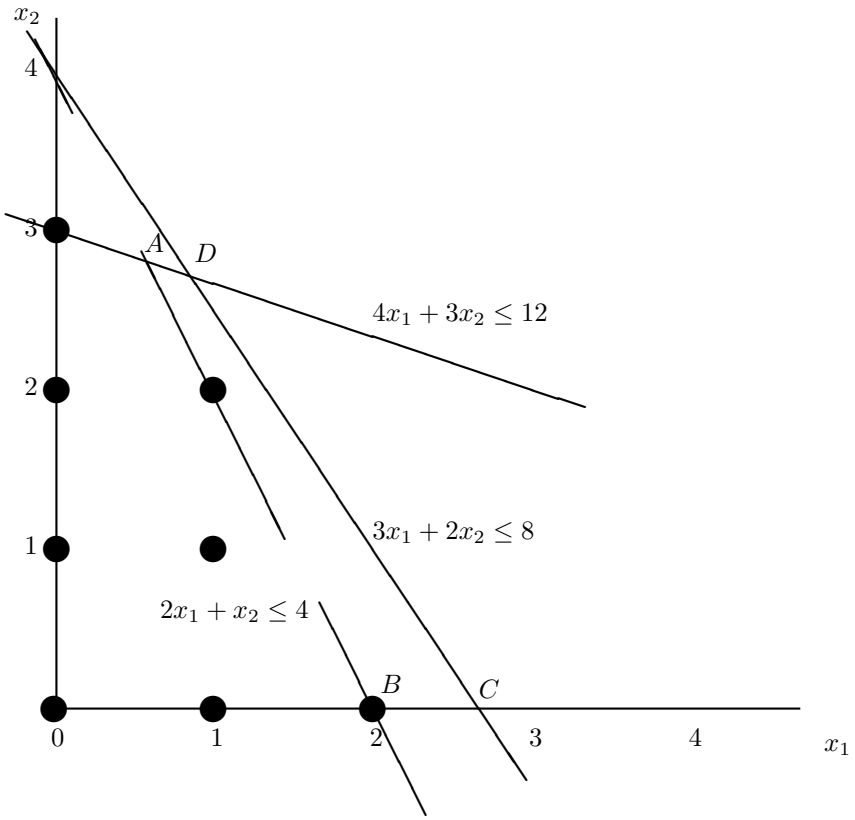

Figure 2.1: *Cutting Plane Method*

Another frequently used technique to solve integer programs is the use of *valid inequalities* discussed in Section 1.2.2, which was introduced by Gomory [32, 33, 34]. A *valid inequality* for an IP is an inequality, $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$, that satisfies every point in P. Valid inequalities are also known as *cuts* or *cutting planes*. This technique aims at generating cuts which eliminate some portion of the linear relaxation without cutting off any feasible solutions. These generated cuts are added to the integer program as new constraints.

Example 2.1 Consider the following integer problem

$$\begin{aligned} \text{Maximize} \quad & x_1 + 2x_2 \\ \text{Subject to} \quad & 4x_1 + 3x_2 \leq 12 \\ & 3x_1 + 2x_2 \leq 8 \\ & x_1, x_2 \in \mathbb{Z}_+. \end{aligned}$$

Figure 2.1 provides a graphical view of this problem. The first constraint $4x_1 + 3x_2 \leq 12$

passes through the points (0,3), A and D. The second constraint $3x_1 + 2x_2 \leq 8$ passes through the points (0,4), D and C. The linear relaxation is defined by the constraints $4x_1 + 3x_2 \leq 12$, $3x_1 + 2x_2 \leq 8$, the axis x_1 and x_2. Within this space the solid circles show the feasible integer solutions to the problem.

The aim of a cutting plane is to eliminate some portion of the linear relaxation without cutting off any integer solution. The dashed line represents a valid inequality $2x_1 + x_2 \leq 4$ and passes through the points (2,0) and (1,2). It can be seen that the inequality cuts off the region $ABCD$ of the linear relaxation without eliminating any feasible integer point. Clearly, this is a strong cut and this cut is further discussed in Example 2.3.

2.1 Polyhedral Theory

Polyhedral theory provides the basic and fundamental concepts of many optimization topics. It deals with the feasible sets of linear programming problems, which are called polyhedra, convex sets and convex analysis. A more detailed discussion on polyhedral theory and its core concepts is presented in this section and a more extensive coverage can be found in [56].

A set S is called a *convex set* if for any two points p and q in S, $\lambda p + (1 - \lambda)q \in S$ for each $\lambda \in [0, 1]$. It can be noted that $\lambda p + (1 - \lambda)q$ represents a point on the line segment joining p and q. Any point of the form $\lambda p + (1 - \lambda)q$ where $0 \leq \lambda \leq 1$ is the weighted average of p and q. Thus, a set S is called a convex set if for any two points p, q in S the straight line segment between p and q is entirely contained in S.

A *halfspace* is the solution space for a single linear inequality, i.e. all $x \in \mathbf{R^n}$ such that $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$. A polyhedron is the intersection of a finite number of halfspaces and hence the feasible region of a linear program is a *polyhedron* and is also convex. A bounded polyhedron is called a *polytope*.

The solutions to an integer program are neither convex nor a polyhedron, which requires

the introduction of a *convex hull*. A convex hull of a set S is defined as the intersection of all convex sets that contain S. Let P again be the set of feasible solutions of an integer program. Define the convex hull of P to be P^{conv}. The P^{conv} is critical in integer programming research, because solving a linear program on P^{conv} results in the optimal integer solution and therefore there is no need to perform branch and bound. Furthermore, it is well known that P^{conv} is a polyhedron.

The dimension of a polyhedron plays a critical role in determining useful cutting planes and P^{conv}. Typically, dimension is defined as the number of linearly independent vectors contained in the polyhedron. However, since the desired polyhedron is derived from a collection of points, affine independence is commonly used. A collection of points $x_1, x_2, x_3, ..., x_r \in \mathbf{R}^n_+$ are affinely independent if and only if $\sum_{i=1}^{r} \lambda_i x_i = 0$ and $\sum_{i=1}^{r} \lambda_i = 0$ is uniquely solved by $\lambda_i = 0$ for all $i = 1, ..., r$.

The dimension of a convex set is defined as the maximum number of linearly independent vectors. Alternately, the dimension of a convex set is the maximum number of affinely independent points minus one. Conventionally, an empty set has dimension equal to -1. With dimension defined, one can now formally describe which cutting planes are necessary to describe P^{conv} by examining faces. Every valid inequality of P^{conv} induces a face of P^{conv}. Given a valid inequality, $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$, its induced face is $\{x \in P^{conv} : \sum_{i=1}^{n} \alpha_i x_i \leq \beta\}$.

One can determine which inequalities are necessary in the description of P^{conv} through the dimension of a face. If $\{x \in P^{conv} : \sum_{i=1}^{n} \alpha_i x_i = \beta\} = \emptyset$, then the face is called a trivial face and the inequality is not necessary in the description of P^{conv}. Furthermore, any inequality that does not define a face that has dimension exactly one less than the dimension of P^{conv} is not necessary in the description of P^{conv}. Inequalities that induce a face of $dim(P^{conv}) - 1$ are called facet defining inequalities. Each facet defining inequality could be represented by infinitely many inequalities, but including one such inequality for each facet will result in P^{conv}, which has integer corner points and so there is no need to perform branch and bound. Facet defining inequalities have been an important part of IP research over many years. A few of these references are [2, 4, 5, 7, 20, 24, 25, 39, 44, 48, 58].

A commonly used theorem to check if an inequality is facet defining is as follows:

Theorem 2.2. *[56] Let $\alpha^T x \leq \beta$ be a valid inequality, then if there exists a point $x \in P^{conv}$ such that $\alpha^T x < \beta$, then the $dim(\{x \in P^{conv} : \alpha^T x = \beta\}) \leq dim(P^{conv}) - 1$.*

Proof: For proof refer [56].

Example 2.3 Revisiting the previous Example 2.1, the solid circles in Figure 2.1 represent the feasible integer solutions. The $dim(P^{conv}) \leq 2$ since there are 2 variables. The three points (0,0), (1, 0) and (0, 1) lie in P and are affinely independent. Therefore, the dim(P^{conv}) $\geq$ 3 - 1 = 2. From the 2 arguments we can see that the $dim(P^{conv}) = 2$.

By inspection it can be clearly said that the cut $2x_1 + x_2 \leq 4$ is valid. It can be noted that the inequality cuts off some of the linear relaxation space enclosed by $ABCD$ without eliminating any integer points. We can see that the cut passes through the points (1, 2) and (2,0) which satisfy the inequality at equality and so the dimension of the face is at least 2 - 1 = 1. The dimension of the face $\leq dim(P)^{conv} \leq$ 2 - 1 = 1 by Theorem 2.2 because (0,0) is feasible and $2(0) + 0 < 4$. Therefore, the above inequality induces a face that has dimension equal to 1 and it is a facet defining inequality.

2.2 Introduction to Knapsack Problem

As discussed in Chapter 1, the Knapsack Problem (KP) is a special class of integer programs. The name knapsack draws an analogy with the problem faced by a camper who needs to fill her knapsack with objects while being restricted by the overall weight. Recall that there are n objects the camper can take and each object has an associated benefit, c_i, that the camper gets for taking the object. On the other hand, each object also has an associated positive weight, a_i, and she is unable to carry more than an overall weight of b. Therefore, a choice has to be made between the n objects such that she gets the maximum benefit and obeys the constraint of the maximum overall weight.

Knapsack Problems are NP-Hard [47] in general and may require exponential time to solve. Dynamic programming can be used to solve KPs with peudo-polynomial time algorithms [11, 29, 30, 67]. Mathematically, a knapsack problem can be solved as an IP.

The knapsack can be modeled by letting x_i be 1, if the object is selected; and 0, if the object is not selected. An IP formulation for a KP is as follows:

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{i=1}^{n} c_i x_i \\ & \textstyle\sum_{i=1}^{n} a_i x_i \leq b \\ & x_i = \mathbf{Z}_+ \ \forall i = 1, 2, ...n. \end{aligned}$$

Let the feasible region of a KP be $P_{KP} = \{x \in \mathbb{B}^n : \sum_{i=1}^{n} a_i x_i \leq b\}$ and $P_{KP}^{conv} = conv(P_{KP})$. Without loss of generality assume that the variables a_i are sorted descending order. This means $a_i \geq a_j$ where $i < j$ and $i, j \in N$. It is assumed that $a_i \geq 0 \ \forall \ i \in N$, if this is not true, then if $a_i = 0$, then x_i can be easily eliminated from the problem. On the other hand, if $a_i < 0$ then x_i can be replaced by $1 - x_i$ and $a_i > 0$. Now if $a_1 \geq b$, then $x_1 = 0$ in all feasible solutions and x_1 can be eliminated from the problem. With these assumptions we can clearly see that P_{KP}^{conv} is full dimensional with the affinely independent points 0 and $e_i \ \forall \ i = 1, 2, ...n$ where e_i is the i^{th} identity vector. The following example presents a knapsack instance.

Example 2.4 Consider the following knapsack problem. Going back to the analogy of the camper in this problem there are a total of 11 objects. The associated benefits and the weight for each object is given in Table 2.4. Also, the camper is constrained by the maximum weight of 90. The main objective of the problem is to maximize the benefit while satisfying the constraint. The IP can be written as:

objects	1	2	3	4	5	6	7	8	9	10	11
benefit	10	7	9	4	10	2	7	8	6	5	2
weight	29	29	27	25	14	14	12	11	11	10	10

Table 2.1: *Associated Weight and Benefit*

Maximize

$$10x_1 + 7x_2 + 9x_3 + 4x_4 + 10x_5 + 2x_6 + 7x_7 + 8x_8 + 6x_9 + 5x_{10} + 2x_{11}$$

Subject to

$$29x_1 + 29x_2 + 27x_3 + 25x_4 + 14x_5 + 14x_6 + 12x_7 + 11x_8 + 11x_9 + 10x_{10} + 10x_{11} \leq 90$$

$$x_i = \{0, 1\} \ \forall \ i = 1, 2, ..., 11.$$

The solution to the above problem is (1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0) with the camper having a benefit given by the objective value $z^* = 46$. From the solution we can see that the camper has selected objects $1, 5, 7, 8, 9$ and 10 which have an overall weight = 29 + 14 + 12 + 11 + 11 + 10 = 87. The above problem will be used throughout this thesis to illustrate the research results.

Knapsack problems is an area of particular interest to researchers. This is because any IP constraint can be transformed into a KP. The transformations are as follows: if the IP constraint is of type ' = ' then it is transformed into two sets of constraints one with ' $\leq$ ' and other with ' $\geq$ '. Greater than or equal to constraints are multiplied by -1 and become less than or equal to constraints. If any $a_i < 0$, then x_i is substituted with $x_i' = 1 - x_i$ and the equivalent problem now has $a_i > 0$. Therefore, any cutting of a KP can be applied to any single IP constraint. Thus, finding stronger cutting planes to KP is of critical importance to integer programming research.

2.3 Cover and Cover Inequalities

As mentioned in Section 1.2.2, cutting planes are a very useful technique for solving IPs. Commonly, used cut for a KP instance is a cover. A cover is a set of indices, which if the corresponding variables are selected together would violate the constraint inequality. The cover can include any number of variables such that together they would be infeasible. There can be exponential number of covers for a given problem. A cover is called a minimal cover if the removal of one variable from the set makes it feasible. Covers have been the focus of research for many years, a few references are [5, 40, 41, 58, 62].

Formally, a *cover*, $C \subseteq N$, is a set of indices where $\Sigma_{i \epsilon C}\ a_i > b$. Covers are very important and induce a cover inequality, which is valid and given by $\Sigma_{i \epsilon C}\ x_i \leq |C| - 1$. A cover is called minimal if $\Sigma_{i \epsilon C - \{j\}}\ a_i \leq b$ for each $j \in C$. Minimal covers generate inequalities of dimension at least $|C| - 1$.

Theorem 2.5. *[56] If C is a minimal cover, then*

$$\sum_{i \in C} x_i \leq |C| - 1 \qquad (I)$$

is a valid inequality for P_{KP}^{conv}.

Proof: For proof refer [56].

Example 2.6 Reconsider the constraint of the knapsack problem in Example 2.4.

$29x_1 + 29x_2 + 27x_3 + 25x_4 + 14x_5 + 14x_6 + 12x_7 + 11x_8 + 11x_9 + 10x_{10} + 10x_{11} \leq 90.$

Clearly, a cover C is $\{2, 3, 4, 5\}$ because the sum of the coefficients $29 + 27 + 25 + 14 = 95$ which is greater than the right hand side of the above constraint inequality (90). This is also a minimal cover since if any variable from the cover is removed the sum is less than 90. Clearly, the cover inequality is $x_2 + x_3 + x_4 + x_5 \leq 3$ and is valid.

A set $E(C) = C \cup \{j \in N - C : a_j > a_i\ \forall\ i \in C\}$ is called an *extended cover* of C or an extension of C. $E(C)$ can be used to generate valid inequality, called an

extended cover inequality (ECI), which takes the form, $\sum_{i \in E(C)} x_i \leq |C| - 1$. The following theorem provides the necessary and sufficient conditions for an extended cover inequality $E(C)$ to be facet defining.

Theorem 2.7. *[56] Let $C = \{i_1, i_2, ..., i_r\}$ be a minimum cover with $i_1 < i_2 < ... < i_r$. If any of the following conditions hold then the extended cover inequality is a facet defining inequality of P^{conv}.*

a. $C = N$.

b. $E(C) = N$ and (i) $(C \backslash \{i_1, i_2\}) \cup \{1\}$ is not a cover.

c. $C = E(C)$ and (ii) $(C \backslash \{i_1\}) \cup \{p\}$ is not a cover, where $p = min\{i : i \in N \backslash E(C)\}$.

d. $C \subset E(C) \subset N$ and (i) and (ii).

Proof: For proof refer [56].

In the above example we can extend the cover C to $E(C)$ by including the variable x_1 and $E(C) = \{1, 2, 3, 4, 5\}$ and the extended cover inequality is $x_1 + x_2 + x_3 + x_4 + x_5 \leq 3$. Clearly, $C \subset E(C) \subset N$ and this satisfies item d of the above Theorem 2.7. Since $(C \backslash \{i_1\}) \cup p = \{3, 4, 5, 6\}$ is not a cover and $(C \backslash \{i_1, i_2\}) \cup \{1\} = \{1, 4, 5\}$ is not a cover; therefore, the extended cover inequality $x_2 + x_3 + x_4 + x_5 \leq 3$ is facet defining.

Minimal cover inequalities describe faces of dimension at least $|C| - 1$. These inequalities are facet defining if one restricts P^{conv} to small dimensions. Formally, define $P_{x_1=0} = \{x \in \mathbf{Z} : Ax \leq b, x_1 = 0\}$ with its convex hull denoted by $P^{conv}_{x_1=0}$. This definition can be extended to any number of equalities or inequalities in the obvious fashion. A common result is that a minimal cover C induces a facet defining inequality over the convex hull of C defined as $P_C^{KP\ conv} = P^{KP\ conv}_{x_j=0\ \forall j \in N \backslash C}$. Frequently, variables in $N \setminus C$ are lifted so that the resulting inequality, called *Lifted Cover Inequality* (LCI), is facet defining for P^{conv}.

2.4 Lifting Variables

Even though an extended cover inequality can be facet defining, lifting provides facet defining inequalities in more circumstances. Lifting is one of the most widely used techniques to generate strong cutting planes and is widely applied in today's commercial integer programming software. Lifting was first introduced by Gomory [36] for bounded integer programs. Since that time, lifting has been an active area of research with substantial contributions being made by [2, 7, 8, 9, 18, 20, 23, 24, 25, 27, 43, 40, 41, 48, 58, 62, 64, 77].

There are numerous different types of lifting. They includes up, down and middle lifting. Each of these types of lifting can be performed sequentially and probably simultaneously. In addition, the lifting coefficient can be either an exact value or an approximate value. Gutierrez [43] provides an excellent review of these various lifting techniques. The contribution of this thesis involves simultaneous uplifting over cover inequalities and so this thesis only provides a thorough review of uplifting.

2.4.1 Sequential Uplifting

The most commonly used lifting technique is sequential uplifting [4, 5, 44, 61, 70, 71]. In binary sequential lifting, the lifting coefficients are determined one by one. This is done by solving a series of optimization problems for each coefficient as discussed below.

Consider a valid inequality $\sum_{i=2}^{n} \alpha_i x_i \leq \beta$ that is valid for $P^{conv}_{x_1=0}$. Solving $Z^* = max\{\sum_{i=2}^{n} \alpha_i x_i : Ax \leq b, x_1 = 1, x \in \mathbf{B^n}\}$. If this is infeasible, then $x_1 = 0$ is a valid inequality. If not, then the inequality $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ is valid for any $\alpha_1 \leq \beta - Z^*$.

To illustrate sequential uplifting, let us reconsider Example 2.4. In the following example the cover is modified and now $C = \{1, 2, 3, 4\}$. Therefore, the valid inequality is $x_1 + x_2 + x_3 + x_4 \leq 3$ will be used. Since C is minimal, this inequality is facet defining over $P_C^{KP\ conv}$. The variables to be sequentially uplifted are $x_5, x_6, x_7, x_8, x_9, x_{10}$ and x_{11} in that order.

Example 2.8 Beginning with x_5 it becomes necessary to solve the following integer pro-

gram:

Maximize

$x_1 + x_2 + x_3 + x_4$

Subject to

$29x_1 + 29x_2 + 27x_3 + 25x_4 + 14x_5 + 14x_6 + 12x_7 + 11x_8 + 11x_9 + 10x_{10} + 10x_{11} \leq 90$

$x_5 = 1$

$x_i = \{0, 1\} \; \forall \; i = 1, 2, ..., 11.$

The optimal solution for the problem is $(1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0)$ and the objective function has the value $Z^* = 2$. Therefore, the coefficient for lifting x_5 is $\alpha_5 = 3$ - $2 = 1$. The new uplifted inequality can now be written as $x_1 + x_2 + x_3 + x_4 + 1(x_5) \leq 3$. The next variable to lift is x_6 and the following integer program needs to be solved.

Maximize

$x_1 + x_2 + x_3 + x_4 + x_5$

Subject to

$29x_1 + 29x_2 + 27x_3 + 25x_4 + 14x_5 + 14x_6 + 12x_7 + 11x_8 + 11x_9 + 10x_{10} + 10x_{11} \leq 90$

$x_6 = 1$

$x_i = \{0, 1\} \; \forall \; i = 1, 2, ..., 11.$

The optimal solution for this problem is $(1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0)$ and the objective function has the value $Z^* = 3$. Therefore, the coefficient for lifting x_6 is $\alpha_6 = 3$ - $3 = 0$. It can further be noted that all other variables from x_7 to x_{11} have coefficients less that x_6 and hence the optimal solution Z^* will at least be 3. It can thus be concluded that the coefficient for lifting $\alpha_i = 0 \; \forall \; i = \{6, 7, ..., 11\}$. Therefore, no more variables can be lifted into this inequality and the final sequentially lifted inequality can be written as $x_1 + x_2 + x_3 + x_4 + x_5 \leq 3$. The affinely independent points for this inequality as shown in the matrix below. Each column represents a unique point and so the matrix below should be read vertically.

$$\left|\begin{array}{cccccccccccc}
0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right|$$

Sequential lifting of variables is highly dependent on the sequence of the lifting. Hence, if the order of the variables being lifting is changed, the coefficient of lifting may also change. In this example, we lift 7 variables and hence there are 7! ways the sequence can vary. Notice that a different lifting order would lead to different coefficient of lifting; i.e. if x_{11} is lifted first; then its coefficient is set to 1 and all other coefficients become 0. Therefore, there are 6! ways to get a 1 as a coefficient for a variables out of the total 7! ways to sequentially uplift the 7 variables not in the cover. The 7 sequentially uplifted inequalities are: $x_1 + x_2 + x_3 + x_4 + x_5 \leq 3$, $x_1 + x_2 + x_3 + x_4 + x_6 \leq 3$, $x_1 + x_2 + x_3 + x_4 + x_7 \leq 3$, $x_1 + x_2 + x_3 + x_4 + x_8 \leq 3$, $x_1 + x_2 + x_3 + x_4 + x_9 \leq 3$, $x_1 + x_2 + x_3 + x_4 + x_{10} \leq 3$ and $x_1 + x_2 + x_3 + x_4 + x_{11} \leq 3$. The average of all the coefficients of lifting (α) for the sequentially lifted inequalities is $x_1 + x_2 + x_3 + x_4 + \frac{1}{7}(x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}) \leq 3$. It will be discussed later that simultaneously lifted inequalities can dominate the average sequentially lifted inequality.

2.4.2 Simultaneous Uplifting

Simultaneous lifting was introduced in 1978 by Zemel [76] for binary variables. The methodology was based on finding the extreme points from the solutions of exponentially many IPs. After work Zemel's work this area of research had been dormant for around 2 decades. Later, from around 1999 much research has been done through what is known as sequence independent lifting [3, 42, 66]. Sequence independent lifting tries to do away with the main drawback in lifting i.e. solving many integer programs and sacrifices the tightness of the cutting plane for a fast cut inequality. That is, sequence independent lifting is an approximate lifting technique and the inequalities formed could often be made stronger.

Simultaneous lifting has been an area of research for my advisor Easton [26] and his research student Kevin Hooker. They proposed a linear time algorithm to simultaneously lift a set of variables into a cover inequality for a binary knapsack problem. Let C represent a cover and $E \subseteq N \backslash C$. Variables in E are simultaneously lifted into the cover inequality; which takes the form $\sum_{i \in C} x_i + \alpha \sum_{j \in E} x_j \leq |C| - 1$ where α is the coefficient of lifting. This algorithm takes $O(|C| + |E|)$ time to generate a valid inequality assuming C and E are sorted descending.

Going back to Example 2.8 using Easton and Kevin Hooker's simultaneous lifting algorithm one generates the inequality as $x_1+x_2+x_3+x_4+\frac{1}{3}(x_5+x_6+x_7+x_8+x_9+x_{10}+x_{11}) \leq 3$. It can be seen that the simultaneously lifted inequality dominates the average of the sequentially lifted inequality, $x_1 + x_2 + x_3 + x_4 + \frac{1}{7}(x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}) \leq 3$.

Recently, Gutierrez [43] developed a technique to simultaneously lift sets of general integer variables. In her research work Gutierrez introduced Simultaneous Lifting Sets of Integer Variables Algorithm to simultaneously lift sets of general integer variables. Given a set $E \subset N = \{1, 2, ..., n\}$ and $F \subset N \setminus E$ and $F \neq \emptyset$. The constraint set $\sum_{i \in E} \alpha_i x_i \leq \beta$ is a valid inequality of $P^{IP}_{x_i=0 \ \forall \ i \in F}$. The simultaneously lifted inequality takes the form of $\alpha \sum_{i \in F} w_i x_i + \sum_{i \in E} \alpha_i x_i \leq \beta$ where α is the lifting coefficient and $w_i \geq 0$ is the scaling coefficient for variables i.

The newly introduced algorithm can be implemented with at most one integer program, that is α can be determined by a single IP. However, it should be noted that this technique requires a good guess for the appropriate scaling coefficients, to create a facet defining inequality by solving a single integer program.

Example 2.9 The algorithm introduced by Gutierrez [43] is illustrated using the KP in Example 2.4. For ease of understanding here the method with multiple IPs is presented to obtain the best lifting coefficient. However, it is possible to perform this simultaneous lifting with a single IP.

$$29x_1 + 29x_2 + 27x_3 + 25x_4 + 14x_5 + 14x_6 + 12x_7 + 11x_8 + 11x_9 + 10x_{10} + 10x_{11} \leq 90.$$

The cover used in this problem is $\{1, 2, 3, 4\}$ and the problem aims to simultaneously lifting x_5, x_6 and x_7 into the cover inequality $x_1 + x_2 + x_3 + x_4 \leq 3$. For this example the scaling coefficient is set to $w_i = 1$ such that there is no priority among any of the indices to be lifted. The coefficient of lifting α is initialized to a arbitrarily high value (M). The problem is now changed as follows:

Maximize

$$x_1 + x_2 + x_3 + x_4 + M(x_5 + x_6 + x_7)$$

Subject to

$$29x_1 + 29x_2 + 27x_3 + 25x_4 + 14x_5 + 14x_6 + 12x_7 + 11x_8 + 11x_9 + 10x_{10} + 10x_{11} \leq 90$$

$$x_i = \{0, 1\} \ \forall \ i = 1, 2, ..., 11.$$

The optimal solution for the above problem is $(0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0)$ and the objective function has the value $Z^* = 1 + 3M$ which is greater than the right hand side of the cover inequality constraint i.e. 3 since M is a high value. Since $1+3M > 3$ the coefficient of lifting α can be obtained by using the optimal point obtained in the objective function which gives us, $0+0+0+1+(\alpha * 3) = 3$, thus $\alpha = \frac{3-1}{3} = \frac{2}{3}$. The new proposed uplifted inequality can now be written as $x_1 + x_2 + x_3 + x_4 + \frac{2}{3}(x_5 + x_6 + x_7) \leq 3$. Using this value of α the next IP problem is:

Maximize

$x_1 + x_2 + x_3 + x_4 + \frac{2}{3}(x_5 + x_6 + x_7)$

Subject to

$29x_1 + 29x_2 + 27x_3 + 25x_4 + 14x_5 + 14x_6 + 12x_7 + 11x_8 + 11x_9 + 10x_{10} + 10x_{11} \leq 90$

$x_i = \{0, 1\} \; \forall \; i = 1, 2, ..., 11.$

Solving the above problem again with the new coefficient of lifting α the optimal solution obtained is $(1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0)$ and $Z^* = 3.33$ which is again greater than 3. Therefore, the new coefficient of lifting $\alpha = \frac{3-2}{2} = \frac{1}{2}$ and the proposed inequality is $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6 + x_7) \leq 3$.

Using $\alpha = \frac{1}{2}$ in the above problem and resolving we get the new optimal solution is $(0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0)$ and the $Z^* = 3$. At this step the Z^* is not greater than the right hand side of the cover inequality and hence, the algorithm terminates. The final value of the coefficient of lifting is $\frac{1}{2}$ and the simultaneously lifted inequality is $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6 + x_7) \leq 3$.

The above example does not bring out the use of the scaling coefficient. The same example can be solved using scaling coefficients $w_i = 2$ for $i = \{5, 6, 7\}$ and $w_i = 1$ for $i = \{8, 9, 10, 11\}$. This would mean that there is twice as much weight for indices $\{5, 6, 7\}$ compared to $\{8, 9, 10, 11\}$.

Example 2.10 Reconsidering the previous example the problem can now be written as:

Maximize

$x_1 + x_2 + x_3 + x_4 + M(2x_5 + 2x_6 + 2x_7 + x_8 + x_9 + x_{10} + x_{11})$

Subject to

$29x_1 + 29x_2 + 27x_3 + 25x_4 + 14x_5 + 14x_6 + 12x_7 + 11x_8 + 11x_9 + 10x_{10} + 10x_{11} \leq 90$

$x_i = \{0, 1\} \; \forall \; i = 1, 2, ..., 11.$

Solving the above problem the optimal solution is $(0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1)$ and $Z^* = 10M > 3$. Therefore, $\alpha = \frac{3-0}{10} = \frac{3}{10}$. The proposed uplifted inequality now becomes is

$x_1 + x_2 + x_3 + x_4 + \frac{3}{10}(2x_5 + 2x_6 + 2x_7 + x_8 + x_9 + x_{10} + x_{11}) \leq 3.$

Now, replacing the new coefficient of lifting in the above problem and resolving the IP the value for the objective function is $Z^* = 3.5$ and optimal solution is $(0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1)$. Here since $3.5 > 3$ the lifting coefficient is re-calculated using the optimal point obtained, $\alpha = \frac{3-2}{2*1+1*1} = \frac{1}{5}$. The proposed lifted inequality now becomes $x_1 + x_2 + x_3 + x_4 + \frac{2}{5}(x_5 + x_6 + x_7) + \frac{1}{5}(x_8 + x_9 + x_{10} + x_{11}) \leq 3$.

Resolving the problem again we get the new optimal solution is $(0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0)$ and the $Z^* = 3$. At this step the Z^* is not greater than the right hand side of the cover inequality and hence, α is not changed anymore and the algorithm terminates. The final value of the coefficient of lifting is $\frac{1}{5}$ and the simultaneously lifted inequality is $x_1 + x_2 + x_3 + x_4 + \frac{2}{5}(x_5 + x_6 + x_7) + \frac{1}{5}(x_8 + x_9 + x_{10} + x_{11}) \leq 3$. Observe that this inequality dominates the average sequential lifted inequality.

Chapter 3

Finding Maximal Simultaneously Lifted Inequalities for Binary Knapsack Problems

The major contribution of this research is presenting a fast algorithm, called the Maximal Simultaneously Lifted Algorithm (MSLA). MSLA simultaneously lifts variables into cover inequalities for a binary knapsack problem. This algorithm quickly generates several inequalities, which can be used as cuts for solving large IPs. The proposed algorithm generates many inequalities, each such inequality frequently induces a large dimensional face. First, two definitions are needed.

Let $S = \{s_1, ..., s_{|S|}\} \subset N$ where S is sorted in ascending order so that $a_{s_i} \leq a_{s_j}$ for all $i < j$. Now define $S_r = \{s_{|S|-r+1}, ..., s_{|S|}\}$ and is the set of r elements of S that correspond to the smallest a values. Similarly, define $S^r = \{s_1, ..., s_r\}$ and is the set of r elements of S that correspond to the largest a values.

The input to MSLA is a knapsack constraint $\sum_{i=1}^{n} a_i x_i \leq b$, a minimal cover $C = \{i_1, i_2, i_3, ..., i_c\}$ and a set $E = \{j_1, j_2, ..., j_e\}$ such that $E \subset N \setminus C$. The sets C and E are arranged in descending order of their a coefficients such that $a_i \leq a_{i+1}$ and $a_j \leq a_{j+1}$ for

sets C and E, respectively.

Let $0 \leq p \leq |C| - 1$, where p represents the indices that correspond to the smallest a coefficients in the cover are selected and set to 1. The sum of these a_i values is given as $csum$. A variable E_{num} is used to record how many indices are chosen for lifting from E. In selecting E_{num} indices from E, these indices are always selected to be the indices that correspond to the largest a coefficients. A value q is maintained, which indicates the maximum number of indices that can be selected from $\{j_1, ..., j_{E_{num}}\}$ and is initially set to 1. The algorithm begins by selecting the first element in E, setting $esum$ to a_{j_1} value, $E_{num} = 1,\ q = 1$ and $\alpha = \infty$. Finally, $esum$ is always equal to $a_{j_{E_{num}-q+1}} + ... + a_{j_{E_{num}-1}} + a_{j_{E_{num}}}$.

The main idea behind the algorithm is that if $csum + esum \leq b$, then p elements from C and q elements from $\{j_1, ..., j_{E_{num}}\}$ are not a cover and so there exists a corresponding feasible point. Therefore, the lifting coefficient must be changed and $\alpha'_{p,E_{num}}$ is changed so that this point with p and q elements meets the new inequality at equality ($\alpha'_{p,E_{num}} = \frac{|C|-1-p}{q}$). Furthermore, both q and E_{num} increase by one. In contrast, if $csum + esum > b$; then p indices from the cover and any q indices selected from from $\{j_1, ..., j_{E_{num}}\}$ are a cover and so the lifting coefficient need not change ($\alpha'_{p,E_{num}} = \alpha'_{p,E_{num-1}}$). Now only E_{num} increases by one. These if statements are repeated until the size of E_{num} equals E.

This entire process iterates through all $p \in \{0, 1, ..., |C| - 1\}$. Upon completing this, let $\alpha^*_{E_{num}}$ be equal to the minimum$_{p\in\{0,1,...,|C|-1\}}\{\alpha'_{p,E_{num}}\}$. Thus, $\alpha^*_{E_{num}}$ is the optimal lifting coefficient for simultaneously lifting these E_{num} variables. Once this is obtained, one can easily generate the maximal simultaneously lifted inequalities by selecting inequalities that have E_{num} as large as possible for a particular value of $\alpha^*_{E_{num}}$. The final maximal simultaneously lifted inequality can be written as $\sum_{i\in C^p} x_i + \alpha^*_{E_{num}} \sum_{j\in E^{E_{num}}} x_j \leq |C| - 1$ where $E^{E_{num}}$ represents the currently lifted E_{num} variables of set E.

3.1 Maximal Simultaneously Lifting Algorithm

Initialization

C := set of variables a in minimal cover; { i_1, i_2, ... i_c}

E := set of variables to be simultaneously lifted; {j_1, j_2, ... j_e}

Sort C, E in descending order;

$p := |C| - 1$

Main Step

while $p \geq 0$ **loop**

$csum := \sum_{r=|C|-p+1}^{|C|} a_{i_r}$

$\alpha'_{p,0} := \infty$

$q := 1$

$E_{num} := 1$

while $E_{num} \leq |E|$ **loop**

$esum := \sum_{r=E_{num}-q+1}^{E_{num}} a_{j_r}$

$total = csum + esum$

if $(total \leq b)$ **then** $\alpha'_{p,E_{num}} := \frac{|C|-1-p}{q}$; $q := q + 1$

else $\alpha'_{p,E_{num}} := \alpha'_{p,E_{num}-1}$

$E_{num} := E_{num} + 1$

end loop

$p := p - 1$

end loop

Termination

For each E_{num} between 1 and $|E|$ report E_{num} with $\min_{p\in\{0,\ldots,|C|-1\}}\{\alpha'_{p,E_{num}}\}$.

The above algorithm can now be illustrated using the same example discussed in Chapter 2.

Example 3.1 Reconsider the knapsack constraint from Example 2.4.

$$29x_1 + 29x_2 + 27x_3 + 25x_4 + 14x_5 + 14x_6 + 12x_7 + 11x_8 + 11x_9 + 10x_{10} + 10x_{11} \leq 90.$$

Clearly, a cover is {1, 2, 3, 4} because the sum of the coefficients $29+29+27+25 = 112$ is greater than the right hand side of the above constraint inequality (90). This is also a minimal cover since if any index is removed from the cover, it is no longer a cover. Clearly, the valid cover inequality is $x_1 + x_2 + x_3 + x_4 <= 3$. Beginning with the above cover we simultaneously lift all other variables in the knapsack constraint; hence, $E = \{5, 6, 7, 8, 9, 10, 11\}$.

To begin the iteration p is set to $|C|-1 = 3$ and $csum$ is given by the sum of the smallest a_i $= a_2+a_3+a_4 = 29+27+25 = 81$ and the lifting coefficient $\alpha'_{3,0}$ is initialized to ∞. The main step begins with selecting the first index in set E i.e. 5, set $q = 1, E_{num} = 1$, the sum of the coefficients of E is given by $esum = a_5 = 14$ and $total = csum+esum = 81+14 = 95$. Since total is greater than the right hand side of the constraint inequality (90) and $\alpha'_{3,1} = \alpha'_{3,0} = \infty$. The next iteration q remains unchanged i.e. $q = 1$, $E_{num} = 2$, $esum = a_6 = 14$ and $total = csum + esum = 81 + 14 = 95$. Again the $total$ is greater than b and so $\alpha'_{3,2} = \infty$. Incrementing $E_{num} = 3$, $q = 1$, $esum = a_7 = 12$ and $total = 81 + 12 = 93$ which is greater than 90, therefore, $\alpha'_{3,3} = \infty$.

In the next iteration $E_{num} = 4$, $q = 1$ and $esum = a_8 = 11$ therefore, $total = 81+11 = 92$ and again $\alpha'_{3,4} = \infty$. Incrementing, $E_{num} = 5$, $q = 1$, $esum = a_9 = 11$, $total = 81+11 = 92$ thus, $\alpha'_{3,5} = \infty$. Again $E_{num} = 6$, $q = 1$, $esum = a_{10} = 10$, $total = 81 + 10 = 91$ and therefore, $\alpha'_{3,6} = \infty$. The last step $E_{num} = 7$, $q = 1$, $esum = a_{11} = 10$, $total = 81+10 = 91$ and total is still greater than the right hand side. Therefore, when $p = 3$ no variables can be to lifted from E and $\alpha'_{3,.} = \infty$. Moving to the next iteration begins with decrementing p.

Now, $p = 2$, $csum = a_3 + a_4 = 52$, $q = 1$, $E_{num} = 1$, $esum = a_5 = 14$ and $total =$

$csum + esum = 52 + 14 = 66$. Since 66 is less than 90 the lifting coefficient is $\alpha'_{2,1} = \frac{(3-2)}{1} = 1$. The proposed inequality with E_{num} lifted variable can be written as $x_1 + x_2 + x_3 + x_4 + 1x_5 \leq 3$. Since the *total* is less than 90, one more variable may be added into the set E_{num} and therefore, both q and E_{num} are incremented. In the second iteration $q = 2,\ E_{num} = 2,\ esum = a_5 + a_6 = 14 + 14 = 28$ and $total = 52 + 28 = 80$. Again 80 is less than 90; therefore the lifting coefficient $\alpha'_{2,2} = \frac{(3-2)}{2} = \frac{1}{2}$ and the potential inequality is $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6) \leq 3$.

Next q and E_{num} are incremented to $q = 3,\ E_{num} = 3,\ esum = 14 + 14 + 12 = 40$ and $total = 52 + 40 = 92$. Here the total greater than 90; hence $\alpha'_{2,3} = \alpha'_{2,2} = \frac{1}{2}$ and so the proposed inequality is $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6 + x_7) \leq 3$.

Now E_{num} increases to 4 and q remains at 3. This indicates that 3 of the 4 indices can be chosen. So $esum$ is now equal to $a_6 + a_7 + a_8 = 37$, $csum = 52$ and $total = esum + csum = 89$. For computational and theoretical speed, this can be determined by taking the prior value of $esum + csum = 92$ and subtracting $a_5 = 14$ and adding on $a_8 = 11$. This operation results in the same value of 89, which is less than 90 and so the new lifting coefficient is $\alpha'_{2,4} = \frac{(3-2)}{3} = \frac{1}{3}$. Therefore, the proposed inequality is $x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8) \leq 3$, and both E_{num} and q increase by one.

Now $q = 4,\ E_{num} = 5,\ total = 89 + 11 = 100$. The value of *total* is greater than 90; therefore, a_6 swaps with a_9, increment E_{num} to 6 and q remains at 4. Now *total* equals $100 - 14 + 11 = 97$, which is greater than 90; therefore, swapping a_7 with a_{10}, $total = 97 - 12 + 10 = 95$. The sum of the coefficients is still greater than 90. Again, E_{num} increase by one to 7 and a_8 swaps with a_{11} and $total = 95 - 11 + 10 = 94$. The sum is still greater than 90 but $E_{num} = 7 = |E|$ and hence the $p = 2$ iteration is complete. The last proposed inequality would be $x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}) \leq 3$.

For the next iteration decrement p, thus, $p = 1,\ q = 1,\ E_{num} = 1,\ esum = a_5 = 14,\ csum = a_4 = 25$ and $total = 25 + 14 = 39$. Since 39 is less than 90, $\alpha'_{1,1} = \frac{(3-1)}{1} = 2$ and the potential inequality is $x_1 + x_2 + x_3 + x_4 + 2(x_5) \leq 3$. Since the *total* is less than 90, lift one more variable and increment q and E_{num}. In the second iteration $q = 2,\ E_{num} = 2,\ esum =$

$a_5 + a_6 = 14 + 14 = 28$ and $total = 25 + 28 = 53$. Here, 53 less than 90; therefore, lifting coefficient $\alpha'_{1,2} = \frac{(3-1)}{2} = 1$ and the potential inequality is $x_1 + x_2 + x_3 + x_4 + 1(x_5 + x_6) \leq 3$.

Again incrementing $q = 3,\ E_{num} = 3,\ esum = a_5 + a_6 + a_7 = 14 + 14 + 12 = 40,\ total = 25 + 40 = 65$ and $\alpha'_{1,3} = \frac{(3-1)}{3} = \frac{2}{3}$. This makes the potential inequality $x_1 + x_2 + x_3 + x_4 + \frac{2}{3}(x_5 + x_6 + x_7) \leq 3$. In the next iteration $q = 4,\ E_{num} = 4,\ esum = a_5 + a_6 + a_7 + a_8 = 14 + 14 + 12 + 11 = 51,\ total = 25 + 51 = 76$ and since 76 is less than 90 $\alpha'_{1,4} = \frac{(3-1)}{4} = \frac{1}{2}$. This makes the inequality $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6 + x_7 + x_8) \leq 3$.

Further, $q = 5,\ E_{num} = 5,\ esum = a_5 + a_6 + a_7 + a_8 + a_9 = 14 + 14 + 12 + 11 + 11 = 62,\ total = 25 + 62 = 87$ and $\alpha'_{1,5} = \frac{(3-1)}{5} = \frac{2}{5}$. This makes the potential inequality $x_1 + x_2 + x_3 + x_4 + \frac{2}{5}(x_5 + x_6 + x_7 + x_8 + x_9) \leq 3$. In the next step $q = 6,\ E_{num} = 6,\ esum = a_5 + a_6 + a_7 + a_8 + a_9 + a_{10} = 14 + 14 + 12 + 11 + 11 + 10 = 72$ and $total = 25 + 72 = 97$. Here 97 is greater than 90 and $\alpha'_{1,6} = \alpha'_{1,5} = \frac{2}{5}$ and the proposed inequality is $x_1 + x_2 + x_3 + x_4 + \frac{2}{5}(x_5 + x_6 + x_7 + x_8 + x_9) \leq 3$.

Now $E_{num} = 7,\ q = 6,\ esum = a_6 + a_7 + a_8 + a_9 + a_{10} + a_{11} = 14 + 12 + 11 + 11 + 10 + 10 = 68,\ total = 25 + 68 = 93$ where 93 is greater than 90. Thus, $\alpha'_{1,7} = \frac{2}{5}$ and the inequality is $x_1 + x_2 + x_3 + x_4 + \frac{2}{5}(x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}) \leq 3$.

For the last iteration decrement $p = 0$ and q and E_{num} increase between 1 and 7 to get various values for $\alpha'_{0,.} = 3,\ \frac{3}{2},\ 1,\ \frac{3}{4}, \frac{3}{5}, \frac{1}{2}$ and $\frac{3}{7}$ since every time the $total \leq 90$.

The various valid combinations of $p, q, \alpha'_{p,E_{num}}$ and E_{num} obtained from MSLA are reported in Table 3.1. The table shows all the values for the combination of p and q. The last column shows the minimum coefficient of lifting $\alpha^*_{E_{num}}$ for lifting E_{num} variables, which is reported. Thus, for the above example the simultaneously lifted inequalities are as follows:

$x_1 + x_2 + x_3 + x_4 + 1(x_5) \leq 3,$

$x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6) \leq 3,$

$x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6 + x_7) \leq 3,$

$x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8) \leq 3,$

	$p=3$		$p=2$		$p=1$		$p=0$		
E_{num}	q	$\alpha'_{3,E_{num}}$	q	$\alpha'_{2,E_{num}}$	q	$\alpha'_{1,E_{num}}$	q	$\alpha'_{0,E_{num}}$	$\min \alpha^*_{E_{num}}$
1	1	∞	**1**	**1**	1	2	1	3	**1**
2	1	∞	2	$\frac{1}{2}$	2	1	2	$\frac{3}{2}$	$\frac{1}{2}$
3	1	∞	**2**	$\mathbf{\frac{1}{2}}$	3	$\frac{2}{3}$	3	1	$\mathbf{\frac{1}{2}}$
4	1	∞	3	$\frac{1}{3}$	4	$\frac{1}{2}$	4	$\frac{3}{4}$	$\frac{1}{3}$
5	1	∞	3	$\frac{1}{3}$	5	$\frac{2}{5}$	5	$\frac{3}{5}$	$\frac{1}{3}$
6	1	∞	3	$\frac{1}{3}$	6	$\frac{2}{5}$	5	$\frac{1}{2}$	$\frac{1}{3}$
7	1	∞	**3**	$\mathbf{\frac{1}{3}}$	5	$\frac{2}{5}$	7	$\frac{3}{7}$	$\mathbf{\frac{1}{3}}$

Table 3.1: *Output Values Generated by the Algorithm*

$x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8 + x_9) \leq 3,$

$x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8 + x_9 + x_{10}) \leq 3$ and

$x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}) \leq 3.$

Although the algorithm reports many inequalities (in this case 7), many of these inequalities are dominated by other inequalities. For instance, the fourth inequality given by, $x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8) \leq 3$ is dominated by the seventh inequality $x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}) \leq 3$. Thus, the *non-dominated inequalities* are precisely the inequalities with the largest number of variables for a particular α value. From this example, there are three non-dominated inequalities. They are when $q = 1, 3$ and 7 and are as follows:

$x_1 + x_2 + x_3 + x_4 + 1(x_5) \leq 3,$

$x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6 + x_7) \leq 3,$

$x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}) \leq 3.$

From the above example it is clear that MSLA generates strong maximal lifted inequalities. That is, the inequalities are defined over the entire set E. The next section further

introduces theoretical results that show the benefits of MSLA.

3.2 MSLA Theoretical Results

MSLA generates many inequalities, which can each be facet defining. This section provides the necessary points to show that the dimension of each of these inequalities is sufficiently high to be facet defining inequalities in this particular example. This section also provides a strong theoretical result, which shows that this is frequently the case.

As mentioned earlier some of the inequalities are dominated by others. In this case each of these non-dominated inequalities is a facet defining inequality $P_{C\cup E}^{KP\ conv}$, which is equal to $P^{KP\ conv}$ in this particular case. The next part of this section deals with checking if the above non-dominated inequalities are facet defining.

Considering the first non-dominated inequality is $x_1 + x_2 + x_3 + x_4 + 1(x_5) \leq 3$. Here, $E_{num} = 1,\ q = 1,\ p = 2$ and $\alpha^*_{2,1} = 1$. The following matrix in Table 3.2 provides 11 affinely independent points. Again, as mentioned earlier each point is represented as a column and hence, the matrix should be read vertically.

As seen in the Table 3.2 the first four points are given by the affinely independent points from the minimal cover $\{1, 2, 3, 4\}$. The next point (0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0) is obtained by choosing the next 2 largest indices in C (since $p = 2$) and 1 index from E_{num} (since $q = 1$). The last set of 6 points are obtained by choosing one of the remaining indices this gives us 6 affinely independent points. Each point is feasible and satisfies the inequality $x_1 + x_2 + x_3 + x_4 + 1(x_5) \leq 3$ at equality and so this inequality is facet defining.

The second non-dominated inequality is $x_1 + x_2 + x_3 + x_4 + \frac{1}{2}(x_5 + x_6 + x_7) \leq 3$. Here, $p = 2,\ q = 2,\ E_{num} = 3$ and $\alpha^*_{2,3} = \frac{1}{2}$. The dimension of the face of the inequality is again 10 by the 11 affinely independent points shown in the Table 3.3. These points meet the inequality at equality and hence are facet defining over the entire space.

Again the first four points are obtained from the minimal cover C. The next set of 3

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

Table 3.2: *Affinely Independent Points for* $E_{num} = 1,\ q = 1,\ p = 2$ *and* $\alpha^*_{2,1} = 1$

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

Table 3.3: *Affinely Independent Points for* $E_{num} = 3,\ q = 2,\ p = 2$ *and* $\alpha^*_{2,3} = \frac{1}{2}$

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

Table 3.4: *Affinely Independent Points for* $E_{num} = 7,\ q = 3,\ p = 2$ *and* $\alpha^*_{2,7} = \frac{1}{3}$

points are obtained by the cyclical permutation of the combination of $p = 2$, $q = 2$ and $E_{num} = 3$, which indicates that 2 out of the 3 indices in E_{num} can be selected. The last set of 4 points are obtained by choosing one of the remaining indices this gives 4 affinely independent points with the largest indices in C and E_{num}.

The third non-dominated inequality is $x_1 + x_2 + x_3 + x_4 + \frac{1}{3}(x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11}) <= 3$. This inequality is obtained when $p = 2,\ q = 3,\ E_{num} = 7$ and $\alpha^*_{2,7} = \frac{1}{3}$. The dimension of the face of the inequality is again 10 by the 11 affinely independent points as shown in Table 3.4. These points prove it is a facet defining inequality since each of these points satisfy the inequality at equality and are feasible.

Therefore, the maximal simultaneously lifting algorithm has generated 3 facet defining inequalities. These inequalities can now be used as cuts to decrease the computational time for solving the IP. The following theorem shows that these maximally lifted inequalities are facet defining over $P^{KP\ conv}_{C\cup E}$ under very weak assumptions.

Theorem 3.2. *Any nondominated inequality reported from MSLA is facet defining over*

$P^{KP}_{C\cup E}{}^{conv}$ *as long as all of the following conditions are satisfied.*

i) The inequality dominates another inequality reported by MSLA.

ii) If $E_{num} - q \geq 2$*, then the set* $\{i_{|C|-p+1},\ i_{|C|-p+2}, ..., i_{|C|}, j_1, j_{|E|-q+2}, j_{|E|-q+3}, ..., j_{|E|}\}$ *is not a cover,*

iii) If $E^{E_{num}} \neq E$*, then* $\alpha_{p,E_{num}} > \alpha_{p,E_{num}+1}$.

where E_{num}, p and q are the numbers reported by the algorithm.

Proof:

We begin by showing that each inequality generated by MSLA is a valid inequality. MSLA proves validity by exhaustively checking every single relevant point that may make the inequality invalid. For a given p value, MSLA calculates $\sum_{i\in C^p} a_i = csum$ and a E_{num} is steadily increased from 1 to $|E|$. For each E_{num} a q value is maintained. This q value represents the number of E_{num} elements that are need to be valid. Thus, the $esum = \sum_{j\in E^q_{num}} a_j$. If this total $(csum + esum)$ is less than b, then $\alpha_{p,E_{num}}$ is changed so that this point with p element in C and q elements in E_{num} satisfies $\sum_{i\in C} x_i + \alpha_{p,E_{num}} \sum_{j\in E^{E_{num}}} x_j = |C| - 1$. If this total is greater than b, then this point is not feasible and the $\alpha_{p,E_{num}}$ value remains unchanged and is equal to $\alpha_{p,E_{num}-1}$. Thus, for a fixed p value, the inequality $\sum_{i\in C} x_i + \alpha'_{p,E_{num}} \sum_{j\in E^{E_{num}}} x_j \leq |C| - 1$ is valid for any $\alpha'_{p,E_{num}} \leq \alpha_{p,E_{num}-1}$.

The algorithm terminates by selecting the minimum $\alpha_{E_{num}}$ across all of p. Since the minimum is taken, the inequality is valid for each of the above feasible points. Therefore, the reported inequalities are valid.

Clearly, the dimension of $P^{KP}_{C\cup E}{}^{conv}$ is $|C| + |E|$. Furthermore, the origin does not meet any simultaneously lifted cover inequality at equality and so by Theorem 2.2, the dimension of the face induced by any reported inequality, $\sum_{i\in C} x_i + \alpha_{E_{num}} \sum_{j\in E^{E_{num}}} x_j \leq |C| - 1$, can be at most $|C| + |E| - 1$.

It suffices to find $|C| + |E|$ feasible and affinely independent points that satisfy $\sum_{i\in C} x_i + \alpha_{E_{num}} \sum_{j\in E^{E_{num}}} x_j = |C| - 1$. MSLA also reports both a p and q value. Observe that if

there are p variables set to one corresponding to elements in C and q variables set to one corresponding to elements in E that such a point meets $\sum_{i\in C} x_i + \alpha_{E_{num}} \sum_{j\in E^{E_{num}}} x_j \leq |C| - 1$ at equality.

MSLA requires that C is minimal and so $\sum_{i\in C\setminus\{k\}} e_i$ for each $k \in C$ is feasible and meets this inequality at equality.

Since $\sum_{i\in C} x_i + \alpha_{E_{num}} \sum_{j\in E^{E_{num}}} x_j = |C| - 1$ dominates at least one inequality. So $\alpha_{E_{num}-1} = \alpha_{E_{num}}$. Therefore, the point $\sum_{i\in C_p} e_i + \sum_{j\in E_q^{E_{num}-1}} e_j$ is feasible. Since E is sorted, the points $\sum_{i\in C_p} e_i + \sum_{j\in E_{q+1}^{E_{num}}\setminus\{k\}} e_j$ are feasible for all $k \in E_{q+1}^{E_{num}}$.

By assumption, if $E_{num} - q \geq 2$, then the set $\{i_{|C|-p+1}, i_{|C|-p+2}, ..., i_{|C|},\ j_1, j_{|E|-q+2}, j_{|E|-q+3}, ..., j_{|E|}\}$ is not a cover. So the points $\sum_{i\in C_p} e_i + e_{j_k} + \sum_{i\in E_{q-1}^{E_{num}}} e_i$ for $k = 1, ..., E_{num} - q - 1$ is feasible and meets the inequality at equality.

If $E^{E_{num}} \neq E$ and $\alpha_{p,E_{num}} > \alpha_{p,E_{num}+1}$, then MSLA changed $\alpha_{p,E_{num}}$ during the $E_{num}+1$ iteration of the p^{th} iteration. Therefore, the point $\sum_{i\in C_p} e_i + \sum_{j\in E^{E_{num}+1}\ _{q+1}} e_j$ is feasible. Thus, the points $\sum_{i\in C_p} e_i + \sum_{j\in E^{E_{num}}\ _{q-1}} e_j + e_k$ are feasible for all $k \in E \setminus E^{E_{num}}$.

These points are clearly affinely independent as the matrix has two cyclically permuted matrices with only one 0 over these permutations. In addition, there are several rows with only a single 1 in them.

□

Reexamining Example 3.1 helps to describe the above theoretical result. As mentioned earlier MSLA requires a minimal cover. The above example has 4 indices in C. Hence, for all the reported inequalities the first $|C|$ affinely independent points are given by the cyclical permutation of this set of cover indices.

In the above theorem part i) states that any inequality that dominates another inequality reported by MSLA is facet defining over $P_{C\cup E}^{KP\ conv}$. From the example it can be seen that the inequality at $p = 2,\ q = 2$ and $E_{num} = 3$ dominates $p = 2,\ q = 2$ and $E_{num} = 2$ with $\alpha'_{2,2} = \alpha'_{2,3} = \frac{1}{2}$. The iteration $p = 2$ and $E_{num} = 2$ is valid therefore, the point $(0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0)$ is feasible. Now since E is sorted descending, it can be seen that

selecting any 2 indices out of 3 possible is feasible and so the points $(0,0,1,1,1,0,1,0,0,0,0)$ and $(0,0,1,1,0,1,1,0,0,0,0)$ are both also feasible.

Condition *iii*) provides the other four affinely independent points for this inequality. Observe that $\alpha_{E^3} = \alpha_{2,E^3} = \frac{1}{2}$ and $\alpha_{2,E^4} = \frac{1}{3}$ and so condition *iii*) is met. Therefore, the point $(0,0,1,1,0,1,1,1,0,0,0)$ is feasible or α_{2,E^4} would have remained $\frac{1}{2}$. Again by the sorted order of a, this final one can be moved to any single one of the final 4 x_i values.

Condition *ii*) is satisfied when $E_{num} = 7$. The returned values for this inequality are $p = 2,\ q = 3,\ E_{num} = 7$ and $\alpha = \frac{1}{3}$ and so $7 - 3 > 2$. In this case one cover must be checked to get the affinely independent points. Since the set is not a cover, the point $(0,0,1,1,1,0,0,0,0,1,1)$ is feasible and therefore, the value of x_5 can be swapped with x_6 and x_7. Since the inequality dominates another inequality, the final four affinely independent points are generated similar to condition *i*).

Easton and Hooker [26] presented an algorithm that can simultaneously lift a set of variables. Their algorithm takes $O(|F| + |C|)$ time to generate one simultaneously lifted cover inequality by lifting all of the variables in F. This result assumes that the sets are sorted. Obviously, one could apply Easton and Hooker's algorithm $|E|$ times to generate the output of MSLA. However, their check for facet defining would only be facet defining over $P_{C\cup F}^{KP\ conv}$ where F is the number of variables lifted in the set. Since only the last case would have $F = E$, their theoretical results are weaker than the results presented here.

Besides providing stronger theoretical results, MSLA is also faster than iteratively applying Easton and Hooker's algorithm. Iteratively applying their algorithm would have a running time of $O(|E|^2 + |C||E| + |C|log|C|)$ effort. MSLA improves the running time by eliminating the $|E|^2$ coefficient. Formally,

Theorem 3.3. *MSLA requires* $O(|E||C| + |C|log|C| + |E|log|E|)$ *effort.*

Proof:

The two main input sets C and E are arranged in descending order and would take $O(|C|log|C|)$ and $O(|E|log|E|)$ time, respectively. Next the algorithm takes a constant p

number of indices from the cover C and iterates it for all indices in E. Each iteration requires $O(|E|)$ effort since each of the sums can be maintained cumulatively as described in Example 3.1. This iteration is then repeated for varying values of p, which can have a maximum number of $|C|$. It can be clearly seen that the main step would take a maximum running time of $O(|E||C|)$. Thus, MSLA requires a running time of $O(|E||C| + |C|\log|C| + |E|\log|E|)$.

□

With the theoretical improvements established, the attention now turns toward the computational advantages of MSLA. This topic is the focus of the next chapter.

Chapter 4

Computational Results

The most important result of this thesis is to generate many maximal facet defining inequalities by simultaneously lifting variables into the cover inequality. These simultaneously lifted cover inequalities are used as cuts and added to the problem as constraint inequalities, which cut off some portion of the linear relaxation without removing any integer point. This process may or may not solve problems faster and this chapter focuses on the computational benefits of MSLA.

The computational study uses randomly generated knapsack problems. The coefficient of the variables a_i are uniformly generated between 50,000 to 100,000. Each c_i is equal to a_i. The value of the right hand side b is calculated as a fraction of the sum of the coefficients given by $s\sum_{i=1}^{n} a_i$. The fraction s called the *slackness ratio* used for this analysis are $\frac{1}{4}$, $\frac{1}{2}$ and $\frac{3}{4}$, which helps gauge the efficiency of the cuts between varied types of knapsack problems.

The study is carried out for different values of n, which varies between 50 and 200 and this presents a comprehensive view of the computational time for different sizes of problems. To avoid anomalies 50 KPs are created for each instance size which means for each instance of n and s 50 KPs are created and hence, a total of 600 instances are generated and solved.

All the variables in the constraint are arranged in descending order based on their co-

efficients a_i. The cover C is chosen as the first i variables that form a minimal cover. To ensure that the coefficient of lifting α is greater than 0, the set E is chosen such that the a_i for each variable in E was greater than the $b - \sum_{i=2}^{|C|} a_i$.

Most solvers can quickly solve knapsack problems. Some knapsack problems that require exponential time are provided in [19]. The result has been further strengthened to show that there are KPs that require exponential branches even if all of the of the sequence independent lifting covers are added to the formulation [45]. These results use large random numbers for a_i solution and force $c_i = a_i$. Therefore, in this computational analysis $c_i = a_i$ is used to generate hard KPs.

The computational study is done on Pentium IV 2.2 GHz processor with 384 MB of RAM and the integer programs were solved using CPLEX 10.0 [68] at its default settings. The upper limit to solve a problem using CPLEX is set at 3600 seconds. This setting is chosen because the problems included in this analysis are large and hard. Therefore, if a problem runs for 3600 seconds it terminates at that point and outputs the current integer solution Z^*.

The algorithm reports several inequalities and the average number of cuts generated is given in Table 4.1. It can be seen that the number of cuts generated with different slackness ratios are different. It is observed that the number of cuts generated by the algorithm also is larger when the set $|E|$ is larger. As expected this would be when the slackness ratio is the 0.25 or 0.5. For example, when slackness ratio is 0.25 and n is 150 then the number of indices in E is 105 and therefore, the number of cuts generated is 18, which is high. Therefore, the larger the set E, the higher the number of cuts generated. When $s = 0.5$ the number of cuts generated are still similar to the number in $s = 0.25$ since $|E|$ is sufficiently high to generate cuts. On the other hand, when the slackness ration $s = 0.75$ the number of cuts generated is between 7 - 14. This could be attributed to the number of indices selected for the cover C is highest for the slackness ratio of 0.75.

Among the 600 instances for the various types of problems used it was observed that there were some instances when $n = 50, 100$ that no cut was generated. Again, such instances were

Num of Vars	Slackness Ratio	Num in C	Num in E	Num of Cuts
50	0.25	11	36	7
	0.50	23	16	7
	0.75	36	8	4
100	0.25	21	46	11
	0.50	45	31	12
	0.75	70	13	6
150	0.25	32	105	18
	0.50	66	48	17
	0.75	105	21	8
200	0.25	42	106	19
	0.50	88	58	20
	0.75	140	37	14
Avg	-	57	44	12

Table 4.1: *Cuts Generated*

found when $s = 0.75$ which can be attributed to the fact that in these instances $|E| = 8, 13$. The number of such instances are 4-8 in a total of 600 instances. This indicates that when KP instances have large slack, then $|C|$ is large which would be generally expected and the number of cuts is lower.

Clearly, as the problem size increases the number of cuts output by MSLA also increases significantly. On an average 12 cuts are generated for all types of the problem.

The only overhead introduced by the lifting process is in the form of preprocessing. The preprocessing step involves determining a minimal cover C and the set E for lifting the variables. It runs through the iteration and as discussed in Chapter 3 the running time was calculated as $O(|C||E| + |C|log|C| + |E|log|E|)$ to generate the maximal lifting inequalities.

Num of Vars	Slackness Ratio	Preprocessing Time (sec)
50	0.25	0.000
	0.50	0.000
	0.75	0.000
100	0.25	0.005
	0.50	0.001
	0.75	0.000
150	0.25	0.014
	0.50	0.000
	0.75	0.006
200	0.25	0.012
	0.50	0.016
	0.75	0.003

Table 4.2: *Preprocessing Time*

These inequalities are then passed into CPLEX as additional constraint to the KP before CPLEX solves the IP. From the table it is noted that for a 200 variable problem and lifting 106 variables requires 0.012 seconds.

As mentioned in theoretical results in Section 3.2 MSLA reduces computational effort to generate the cut. Also from Table 4.2 notice that $|C| < |E|$ in many instances where $s = 0.25$ or 0.5 and these instances generate a higher number of cuts. Therefore, eliminating the $|E|^2$ term from Eason and Hooker's [26] running time is computationally very beneficial.

It can be observed that since preprocessing is dependent on $|C|$ and $|E|$ thus, the preprocessing time increases as the cover set and lifted set are increased in number. From Table 4.2 we can see the pre-processing time observed for the IPs solved for this analysis is less

than $\frac{1}{50}^{th}$ of a second for all the observations. Clearly, the preprocessing time would not be considerable compared to the enormous reduction in computational time for solving the IP and will not cause any kind of serious slow down even on easy to solve IPs. Therefore, we can confidently say that the processing time for this algorithm is negligible and MSLA could be implemented even on easy to solve IPs.

MSLA aims to reduce computational time by adding additional cuts to the KP. These cuts are generated by simultaneously lifting variables into the cover inequality. Therefore, the cover inequality is further strengthened to generate stronger cuts. The next analysis deals with the time required to solve the problem. Obviously, the a good cover inequality would also reduce computational time; however, MSLA further improves this and greatly reduces computational time.

Table 4.3 gives an overview of the relative decrease in computational time when using this algorithm compared to the time taken by CPLEX to solve the IP. The first column lists the different combinations of the number of variables and the slackness ratio. As mentioned earlier the number of variables vary between 50 - 200 and slackness ratio s is varied between 0.25 - 0.75.

It is interesting to see the difference in the computational time while generating cuts using MSLA. As mentioned earlier the cuts were generated using the cover inequality. Therefore, it should be necessary to check if this reduction in computational time is just due to the cover inequality or by the simultaneously lifting of variables into this inequality. Table 4.3 gives this comparison of different cuts used to solve the KPs. The third column in the table shows the time taken to solve the original KP using CPLEX 10.0. This would be CPLEX at its default settings. The fourth column lists the time taken to solve the KP using CPLEX with the additional cover constraint. The fifth column gives the time required to solve the KP instances using MSLA and adding all the generated cuts. Finally the last two columns contrast the results obtained from MSLA with those of CPLEX. Hence, the table brings out the increased efficiency gained by using MSLA.

From the computational study conducted it can be seen that the algorithm is extremely

		CPLEX 10.0	Cover	MSLA	Comparison	
Num of Vars	Slackness Ratio	Time to Solve	Time to Solve	Time to Solve	Diff in Time	Percent Improvement in Time
50	0.25	29.03	29.6	28.1	0.91	3
	0.50	19.75	31.5	18.9	0.87	4
	0.75	30.16	42.7	22.9	7.3	24
100	0.25	40.3	66.4	40.1	0.15	3
	0.50	81.1	67.7	63.6	17.5	22
	0.75	57.2	57.6	54.2	3.01	5
150	0.25	140.9	276.9	58.2	82.7	59
	0.50	64.0	72.5	57.5	6.5	10
	0.75	86.0	99.4	75.9	10.12	12
200	0.25	187.7	150.9	87.2	100.4	53
	0.50	333.3	208.7	51.9	281.3	84
	0.75	162.5	146.5	157.5	5.08	3
Avg	0.25	99.5	130.9	53.4	46.1	46
	0.50	124.5	95.1	46.6	66.7	58
	0.75	83.9	86.6	78.1	1.9	23
Total	-	102.7	104.2	59.7	42.9	41

*Time provided in seconds

Table 4.3: *Comparing Computational Time For Different Techniques*

Num of Vars	Slackness Ratio	Num Nodes by CPLEX	Num Nodes Using Cut From Algorithm
50	0.25	89912	90082
	0.50	46939	58266
	0.75	96741	70809
100	0.25	104567	101759
	0.50	100694	178670
	0.75	113180	158934
150	0.25	230071	112631
	0.50	148131	144037
	0.75	192191	184416
250	0.25	38778	120476
	0.50	190472	117649
	0.75	232981	275010

Table 4.4: *Nodes Fathomed To Solve The IP*

efficient when compared to CPLEX for larger problems. It can be seen from Table 4.3 that the problems with 50 variables CPLEX performs equally good as the problem with additional cut generated from MSLA. This could be because the problems are so quickly solved.

However the for the larger problems the benefit is substantial. The time required to solve the IP is reduced by 10 - 59% for just 150 variables while for 200 variables the improvement is between 18 - 97 %. This huge improvement is also expected for many large real world problems. It is postulated that the reduction in computational time increases as the problem becomes larger. Therefore, MSLA should be added as a module in leading software packages.

From Table 4.4 reports the average numbe rof nodes in branch and bound tree and it provides an explanation on why MSLA is so much faster than CPLEX. The number of

nodes taken to solve the large optimization problems is immensely reduced by adding the cuts generated by MSLA. Clearly, the improvement in time is greater for larger problems with more variables. This could be attributed to the fact that large problems require large number of nodes that need to be fathomed by the branch and bound method used by CPLEX.

From the above discussion it can be concluded that not only does the algorithm proposed in this thesis solve problems faster but it also reduces the number of nodes that need to be fathomed by CPLEX. I stongly suggest that MSLA should be used as a module in any leading software to help solve large complex IPs.

Chapter 5

Conclusion and Future Work

The most important result for this thesis is the Maximal Simultaneously Lifting Algorithm (MSLA). This algorithm quickly generates numerous valid inequalities, which can be used as cuts for solving large IPs. Each such generated inequality induces a large dimensional face as discussed in Chapter 3

One advantage of MSLA is that it is computationally fast. The running time for this algorithm is given by $O(|E||C| + |C|log|C| + |E|log|E|)$. This running time is an improvement over the previously existing method proposed by Easton and Hooker [26]. Besides an improved running time, an additional benefit of MSLA is that its generation of inequalities provides stronger theoretical results than the results presented by Easton and Hooker.

On an average 12 cuts are reported by MSLA for random knapsack problems varying in size from 50 - 200. The additional pre-processing time observed for these IPs is less than $\frac{1}{50}^{th}$ of a second for all the observations. Clearly, this preprocessing time is acceptable and will not cause any kind of serious slow down of the process even on simple IP problems.

From the computational study conducted it can be seen that the algorithm is extremely efficient when compared to CPLEX for larger problems. The time required to solve the IP is reduced by 10 - 59% for just 150 variables while for 200 variables the method is 18 - 97 % faster. It is postulated that the reduction in computational time increases as the

problem becomes larger. This huge improvement is also expected for many large real world problems. Therefore, MSLA should be added as a module in leading software packages.

5.1 Future Research

Simultaneous lifting of variables through MSLA has generated interesting results and this thesis has opened numerous avenues for further research in the area of simultaneous lifting of variables some of which are discussed in this section. These research areas can be broadly classified into theoretical research and computational work.

5.1.1 Theoretical Research Areas

MSLA takes an input of C, E and a single knapsack constraint. However, an IP can have multiple knapsack constraints. An logical area of research could extend MSLA for knapsack problems with multiple constraints. It would be a challenging to find lifting coefficients that would be feasible over all constraints. That is first choose a cover over the set of constraints. Once a variable is lifted into the cover inequality with coefficient α this point should be feasible for all the constraints in the knapsack problem. It would be a very challenging area to be able to satisfy multiple constraints simultaneously and still be feasible. However, this would provide a very strong cut and may give substantial computational improvements to extremely hard problems.

MSLA chooses a single set E for lifting variables, but there could be multiple sets say E, F, G and H. The variables for these sets could be chosen from different areas of a very large instance. These variables once lifted could form a very strong inequality and prove to be very beneficial. Thus, the reported inequalities would likely be facet defining over $P^{KP\ conv}_{CUEUFUGUH}$.

The algorithm proposed in this thesis deals with lifting binary integer variables for a knapsack problem i.e. $x_i = \{0, 1\}\ \forall\ i = \{1, 2, ..., n\}$. These results could be extended to a

knapsack problem with general integer variables. Research must be done to account for the changes by removal of the binary assumptions.

Similarly, simultaneous lifting technique could be applied to general IP problems. The algorithm proposed in this thesis is restricted to knapsack problem and it would be an interesting study to extend this research into general IPs, but this appears to be a daunting task.

5.1.2 Computational Areas

In the computational study the variables in C and E are arranged in descending order with respect to their coefficients a_i. The first i variables are taken as the cover C and the next j variables are taken as the lift set E. Choosing the variables in C and E are important for the kind of inequality generated by MSLA. The study could be expanded to choose a cover C that would be most beneficial and extend the cover on both sides to include variables with greater than and less a_i values than the cover variables. This would bring forward interesting results and may lead to better cuts.

Also a more extensive computational study could help bring out more detailed analysis and give further insight in areas that could potentially reduce the computational time for an IP. It would also provide further evidence that MSLA should be implemented in commercial code.

Bibliography

[1] Ahmed, S. and N. V. Sahinidis (2003). "An approximation scheme for stochastic integer programs arising in capacity expansion ," *Operations Research*, **51** (3), 461-471.

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

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

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

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

[6] Balas, E. and E. Zemel (1980). "An algorithm for large zero-one knapsack problems ," *Operations Research*, **28** (5), 1130-1154.

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

[8] Balas, E. and S. M. 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] Benders, J. F. (1962). "Partitioning procedures for solving mixed variables programming problems," *Numerische Mathematik* **4**, 238-252.

[11] Bertsimas, D. and Demir R. (2002). " An Approximate Dynamic Programming Approach to Multidimensional Knapsack Problems," *Management Sciences* **48** (4), 550-565.

[12] Bitran G. R. and A. C. Hax (1981). "Disaggregation and resource allocation using convex knapsack problems with bounded variables ," *Management Sciences* **27** (4),431-441.

[13] Blair, C. (1976). "Two rules for deducing valid inequalities for 0-1 problems," *SIAM Journal of Applied Mathematics* **31**, 614-617.

[14] Borchers, B. and J. E. Mitchell (1997). "A computational comparison of branch and bound and outer approximation algorithms for 0-1 mixed integer nonlinear programs," *Computers & Operations Research*, **24** (8), 299-701.

[15] Bretthauer, K. M., and B. Shetty (1995). "The nonlinear resource allocation problem," *Operations Research*, **43** (4), 670-684.

[16] Campbell. J. F. (1994). "Integer programming formulations of discrete hub location problems," *European journal of operational research*, **72** (2), 387-405.

[17] Caprara. A., M. Fischetti, P. Toth, D. Vigo, P. L. Guida (1997). "Algorithms for railway crew management ", *Mathematical Programming*, **79** (1-3), 125-141.

[18] Carr, R. (1996). "Separating over classes of TSP inequalities defined by 0 node-lifting in polynomial time," *Integer Programming and Combinatorial Optimization. 5th International IPCO Conference Proceedings*, 460-474.

[19] Chavatal, V. (1980). "Hard knapsack problems," *Operations Research*, **5**, 266-277.

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

[21] Chvátal, V., (1973). "Edmonds polytopes and a hierarchy of combinatorial problems", *Discrete Mathematics*, **4**, 305-337.

[22] Cooper, M. W. (1981). "A survey of methods for pure nonlinear integer programming," *Management Science*, **27** (3) 353-361.

[23] 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.*

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

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

[26] Easton, T. and K. Hooker., (2007) "Simultaneously lifting sets of variables and scaled multiple cover inequalities for knapsack polytopes," to appear in *Discrete Optimization.*

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

[28] Freidenfelds. J (1980). "Capacity expansion when demand is a birth-death random process ," *Operations Research*, **28** (3), 712-721.

[29] Gilmore, P. C., and R. E. Gomory (1965). "Multistage Cutting Stock Problems of Two and More Dimensions," *Operations Research*, **13** (1), 94-120.

[30] Gilmore, P.C., and R. E. Gomory (1966). "The Theory and Computation of Knapsack Functions," *Operations Research*, **14** (6), 1045-1074.

[31] Glover, F., (1975). "Improved linear integer programming formulations of nonlinear integer problems," *Management Science*, **22**, 455-460.

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

[33] Gomory, R. (1960). "Solving linear programming problems in integers," *Combinatorial Analysis*, R.E. Bellman and M. Hall, Jr. eds., American Mathematical Society 211-216.

[34] Gomory, R. (1963). "An algorithm for integer solutions to linear programs," *Recent Advances in Mathematical Programming*, R. Graves and P. Wolfe, eds. McGraw-Hill 269-302.

[35] Gomory, R. (1963). "An all-integer programming algorithm," *Industrial Scheduling*, J.F. Muth and G. Thompson, eds. Prentice-Hall 193-206.

[36] Gomory, R., (1969). "Some polyhedra related to combinatorial problems," *Linear Algebra and it Applications*, **2**, 451-588.

[37] Gomory, R. and E. Johnson (1972). "Some continuous functions related to corner polyhedra," *Mathematical Programming*, **3**, 23-85.

[38] Gomory, R. and E. Johnson (1973). "The group Problem and subadditive functions," *Mathematical Programming*, T. Hu and S. Robinson, eds. Academic Press, 157-184.

[39] Gottlieb, E. S. and M. R. Rao, (1990). "The generalized assignment problem: Valid inequalities and facets," *Mathematical Programming*,**46** (1-3).

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

[41] Gu, Z., G. Nemhauser, and M. Savelsbergh (1999). "Lifted cover inequalities for 0-1 integer programs: computation," *Mathematical Programming*, **85** (3), 439-467. 109-121.

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

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

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

[45] Hunsaker, B. and C. Tovey (2005). "Simple lifted cover inequalities and hard knapsack problems," *Discrete Optimization*, **2** (3), 219-228.

[46] Karabuk, S. and S.D. Wu (2003). "Coordinating strategic capacity planning in the semiconductor industry ," *Operations Research*, **51** (6), 839-849.

[47] Karp, R.M. (1972)."Reducibility among combinatorial problems," *Complexity of computer computations*, **Plenum Press, New York** 85-103.

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

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

[50] Laguna, Manuel (1998). "Applying robust optimization to capacity expansion of one location in telecommunications with demand uncertainity," *Management Sciences*, **44** (11), S101-S110.

[51] Lawler, E.L. and J.M. Moore (1969). "A functional equation and its application to resource allocation and sequencing problems ," *Management Science*, **16** (1), 77-84.

[52] Li, H.(1992). "An approximate method for local optima for nonlinear mixed integer programming problems," *Computers and Operations Research*, **19** (5), 435-445.

[53] Liu, Y.H., J.M. Chuang, and M.J. Hwang (2006). "Solving a nonlinear integer program for allocating resources," *Mathematical and Computer Modeling*, **44** (3), 377-381.

[54] Mansini, R. and M. G. Speranza (2002). "A multidimensional knapsack model for asset-backed securitization ," *Journal of the Operational Research Society*, **53**, 822-832.

[55] Marcotte, O., (1985). " The cutting stock problem and integer rounding ," *Mathematical Programming*, **33** (1), 82-92.

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

[57] Nemhauser, G. L. and Z. Ullmann, (1969). "Discrete Dynamic Programming and Capital Allocation," *Management Sciences*, **15** (9), 494-505.

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

[59] The New Scientist Journal Home Page,

http://www.newscientist.com/article.ns?id=dn3080.

[60] Nilsson, O. and D. Sjelvgren,(1995) "Mixed-integer programming applied to short-term planning of a hydro-thermal system," *IEEE Transactions on Power Systems*, **11** (1), 281-286.

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

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

[63] Pursimo J. M., H. K. Antila, M. K. Vilkko and P. A. Lautala (1998). "A short-term scheduling for a hydropower plant chain," *International Journal of Electrical Power and Energy Systems*, **20** (8), 525-532.

[64] 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.

[65] Santanu, S. D. and P. R. Jean-Philippe, (2006). "Linear programming based lifting and its application to primal cutting plane algorithms," School of Industrial Engineering, Purdue University.

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

[67] Trick, M. A., (2003). "A Dynamic Programming Approach for Consistency and Propagation for Knapsack Constraints," *Annals of Operations Research*, **118** (1-4), 73-84.

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

[69] Wedlin, D. (1995) "An algorithm for large scale 0-1 integer programming with application to airline crew scheduling," *Annals of Operations Research*,**57**(1), 238-301.

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

[71] Wolsey, L.A. (1975b). "Facets and strong valid inequalities for integer programs," *Operations Research*, **24**, 367-372.

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

[73] Xiaohong Guan, Ni Ernan, Li Renhou and P. B. Luh, (1997)." An optimization-based algorithm for scheduling hydrothermal powersystems with cascaded reservoirs and discrete hydro constraints," *IEEE Transactions on Power Systems*, **12** (4), 1775-1780.

[74] Yen, J. W., J. R. Birge (2006). "A stochastic programming approach to the airline crew scheduling problem," *Transportation Sciences*, **40** (1), 3-14.

[75] Yu, G (1996). "On the max-min 0-1 knapsack problem with robust optimization applications," *Operations Research*, **44** (2), 407-415.

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

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