GENERATING CUTTING PLANES THROUGH INEQUALITY MERGING FOR INTEGER PROGRAMMING PROBLEMS

by

RANDAL EDWARD HICKMAN

B.S., United States Military Academy, West Point, NY, 1997

M.S., Massachusetts Institute of Technology, Cambridge, MA, 2005

AN ABSTRACT OF A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Industrial and Manufacturing Systems Engineering

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2014

ABSTRACT

Integer Programming (IP) problems are a common type of optimization problem used to solve numerous real world problems. IPs can require exponential computational effort to solve using the branch and bound technique. A popular method to improve solution times is to generate valid inequalities that serve as cutting planes.

This dissertation introduces a new category of cutting planes for general IPs called inequality merging. The inequality merging technique combines two or more low dimensional inequalities, yielding valid inequalities of potentially higher dimension. The dissertation describes several theoretical results of merged inequalities.

This research applies merging inequalities to a frequently used class of IPs called multiple knapsack (MK) problems. Theoretical results related to merging cover inequalities are presented. These results include: conditions for validity, conditions for facet defining inequalities, merging simultaneously over multiple cover inequalities, sequentially merging several cover inequalities on multiple variables, and algorithms that facilitate the development of merged inequalities. Examples demonstrate each of the theoretical discoveries.

A computational study experiments with inequality merging techniques using benchmark MK instances. This computational study provides recommendations for implementing merged inequalities, which results in an average decrease of about 9% in computational time for both small and large MK instances. The research validates the effectiveness of using merged inequalities for MK problems and motivates substantial theoretical and computational extensions as future research.

GENERATING CUTTING PLANES THROUGH INEQUALITY MERGING FOR INTEGER PROGRAMMING PROBLEMS

by

RANDAL EDWARD HICKMAN

B.S., United States Military Academy, West Point, NY, 1997

M.S., Massachusetts Institute of Technology, Cambridge, MA, 2005

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Industrial and Manufacturing Systems Engineering

College of Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

2014

Approved by:

Major Professor

Dr. Todd Easton

ABSTRACT

Integer Programming (IP) problems are a common type of optimization problem used to solve numerous real world problems. IPs can require exponential computational effort to solve using the branch and bound technique. A popular method to improve solution times is to generate valid inequalities that serve as cutting planes.

This dissertation introduces a new category of cutting planes for general IPs called inequality merging. The inequality merging technique combines two or more low dimensional inequalities, yielding valid inequalities of potentially higher dimension. The dissertation describes several theoretical results of merged inequalities.

This research applies merging inequalities to a frequently used class of IPs called multiple knapsack (MK) problems. Theoretical results related to merging cover inequalities are presented. These results include: conditions for validity, conditions for facet defining inequalities, merging simultaneously over multiple cover inequalities, sequentially merging several cover inequalities on multiple variables, and algorithms that facilitate the development of merged inequalities. Examples demonstrate each of the theoretical discoveries.

A computational study experiments with inequality merging techniques using benchmark MK instances. This computational study provides recommendations for implementing merged inequalities, which results in an average decrease of about 9% in computational time for both small and large MK instances. The research validates the effectiveness of using merged inequalities for MK problems and motivates substantial theoretical and computational extensions as future research.

Contents

List of Figures

List of Tables

Acknowledgments

Thank you first to my wife Courtney and our children Meredith, Ashley, Robert, Laura, and Thomas. Perhaps this work will be seen as a small reflection of your long-standing love and support for the past many years.

I have also benefited immensely from the excellent mentorship and scholarly support from my advisor Dr. Todd Easton. It was a privilege to be your student.

Kansas State University provided the perfect environment to grow and learn as a scholar. I have always appreciated the warm reception and willingness to help by the members of my dissertation committee and the faculty and staff of the Department of Industrial and Manufacturing Systems Engineering at K-State.

Lastly, I offer my sincere appreciation to the United States Military Academy for giving me this opportunity to complete my PhD work and return as a member of the senior faculty. I look forward to rejoining the team soon!

Chapter 1

Introduction

Operations research techniques (analytics) have become the norm for twenty-first century leaders in both the public and private sectors [51]. Reduced budgets, limited resources, and increased global competition necessitate optimal resource management. Furthermore, modern leaders must make their decisions very rapidly in order to keep up with competitors and changing circumstances. These requirements motivate the continued development and refinement of optimization techniques that may be tailored to solve many such problems.

Integer programming (IP) is a form of discrete optimization where some or all of the decision variables are restricted to integer values. One popular variant of modeling an IP is known as the knapsack problem (KP), and modeling a KP with two or more constraints is known as the multiple knapsack (MK) problem. To assist the solution process for these problems, a common strategy is to further constrain the problem by adding new inequalities that separate the linear relaxation of the IP from the convex hull of potential integer solutions. Such inequalities are known as cutting planes.

Many of the most challenging real-world optimization problems contain integer programming components. In addition to general IP formulations, many of these problems include KP and MK formulations. Consequently, improved solution techniques for IPs, KPs, and

MKs are the topic of a vast body of literature demonstrating both theoretical extensions and many modern applications.

Some examples of general theoretical advancements include exact methods to solve the KP by Dudzinski and Walukiewicz [29] and extensions to the nonlinear KP by Hochbaum [47]. The development of new types of cutting planes may be seen in recent papers by Hooker [48] and Chu and Xia [20]. Inspiration for new theory is often motivated by applied problems, and this process begins with innovative modeling strategies. A recent text by Williams titled *Model Building in Mathematical Programming* [73] offers general insights to model and solve IPs and other optimization problems.

Integer programming formulations serve as a natural structure for many modern problems. A survey of recent papers highlights a wide variety of applications, and a few of the topics are described in greater detail. Three broad categories of applications include allocation of resources, logistics and transportation problems, and scheduling problems.

Bartlett, et. al. developed an application of the knapsack problem that may be used to allocate resources in a competitive environment [13]. Two uses highlighted in this paper include allocation of communication bandwidth for a myriad of purposes and allocating multiprocessor resources among numerous computer stations. Melachrinoudis and Kozanidis demonstrated the usefulness of IP as a tool to determine allocation of funds for highway safety improvements [59].

The logistics industry experiences significant benefit from optimization, and IP serves as a natural modeling structure for many transportation problems. The fleet assignment problem is solved as a large-scale IP by Hane, et. al. in [43]. Trick considers an IP formulation to optimize a particular transportation problem in [71]. Pajunas, et. al. used large-scale IP in a decision-support tool that aids the United States Postal Service by identifying cost savings opportunities in the surface transportation network [64]. IP also serves as a framework employed by Demirel and Gökeen in [24] for the relatively new discipline of reverse logistics.

Such reverse-flow modeling may incorporate ecological factors, government regulation, and socially responsible environmental concerns.

Scheduling problems are a third area for important applications of integer programming. Dowsland and Thompson use the knapsack framework of IP to model nurse scheduling at a large hospital [28]. Project scheduling is an important aspect of optimized manufacturing, and Seda demonstrates a multiple knapsack model to solve this problem in [67]. Hansen and Lidén use an IP formulation to schedule airline cabin crews for a large air-transportation provider in [44]. Easton, et. al. also used IP to facilitate sports scheduling and provide solutions to the traveling tournament problem in [30].

1.1 Research Motivation

While many continuous optimization problems can be solved very quickly, there is no known polynomial time algorithm for the optimal solution to integer programming problems unless $\mathcal{P} = \mathcal{NP}$. The branch and bound technique finds the optimal solution to IP problems in exponential time, but that is not satisfactory in many practical instances. Consequently, researchers have studied a variety of techniques to improve the solution time for IP problems.

Slow computational times for these integer programming problems limit the decision-maker's ability to use the results of the IP in a fast-paced global environment. Consequently, any method that yields faster solution times for integer programming problems has the potential to significantly improve the efficient and effective decision-making by modern leaders.

The research objective of this dissertation is to advance the discipline of cutting plane design and implementation in IP problems. This work develops and demonstrates the theoretical foundations for generating a new category of valid cutting plane inequalities for IP problems through inequality merging. Inequality merging yields a new class of valid inequalities that are fundamentally different from other known cutting planes, and the technique is shown to reduce the solution times of MK instances.

A significant accomplishment of this research is that it enables other scholars to extend inequality merging to a variety of integer programming problems. It is likely that merged inequalities of various forms fundamentally improve the understanding of solution techniques for IPs. Furthermore, inequality merging may become widely used helping practitioners solve complex IP problems more rapidly. These solutions should benefit companies, governments, and society alike, enabling twenty-first century leaders to make better decisions.

1.2 Research Contributions

Inequality merging is a new method to generate cutting planes for IP problems with a wide variety of natural extensions. The dissertation provides the theoretical foundations necessary to understand and implement the technique. The implementation of inequality merging is facilitated through several algorithms in this dissertation. A thorough computational study demonstrates that inequality merging reduces the solution times for benchmark MK instances. A variety of theoretical extensions are considered that may improve solution times under certain conditions.

The research contained in this dissertation answers the following fundamental questions:

i. Does inequality merging advance the discipline associated with using cutting planes to solve integer programming problems?

ii. What algorithms are recommended or required to implement inequality merging in a computer program?

iii. What are the best implementation strategies for inequality merging, and does the technique facilitate faster computation and solution times for integer programming problems when these strategies are followed?

iv. How can the initial theory of inequality merging be extended to new varieties of merged inequalities?

A more complete discussion of each research question is contained in the paragraphs below.

1.2.1 Theoretical Contributions Developing a New Category of Cutting Planes

The dissertation demonstrates that inequality merging is a theoretically proven concept that yields valid, higher dimensional cutting plane inequalities by merging two or more lower dimensional inequalities. This could be viewed as lifting one valid inequality into another valid inequality. The resulting merged inequality is theoretically stronger than the original inequalities, and it may be facet defining under some conditions.

The dissertation argues that inequality merging is a new approach for creating cutting planes in integer programming problems that is fundamentally different from any other known techniques. As such, the proposed methodology advances the understanding of using multiple cutting planes in an elegant and potentially beneficial manner for integer programming practioners. The dissertation provides a detailed treatment of two natural extensions to the basic inequality merging technique, thus continuing to advance the discipline as new inequality merging theory is developed and demonstrated.

1.2.2 Algorithms

Inequality merging is a new method to generate valid cutting planes for IP problems. Thus, the construction of merged inequalities requires algorithms that support the new theory. Several algorithms are presented that construct, strengthen and verify merged inequalities. The algorithms presented in this dissertation include the following contributions.

i. The Merging Over a Cover Inequality Algorithm (MOCIA) verifies the validity of a merged cover inequality.

ii. The Reducing ψ_p Algorithm is used to construct the initial (host) inequality before merging may occur.

iii. The Donor Coefficient Strengthening Algorithm (DCSA) may be used to strengthen coefficients attained in a merged inequality.

iv. The Sequential Cover Merging Algorithm (SCMA) may be used to construct a specific type of merged inequality, considered in depth as a theoretical extension of basic inequality merging.

1.2.3 Implementation Techniques and Computational Study

A computer program was developed that constructs merged inequalities for testing on benchmark MK problems. The dissertation considers several implementation strategies for the construction of merged cover inequalities. These include the choice of several possible pseudo-costing techniques, how many merged inequalities to generate, a possible decision to overlap rows when multiple cuts are added, and a possible decision to strengthen merged coefficients. A detailed examination of these variables is considered in the computational study, yielding recommended implementation strategies for MK instances.

A thorough computational study demonstrates the benefits of inequality merging when applied to benchmark MK instances. Inequality merging is tested using the current CPLEX solver [21] as the baseline for comparing computational performance with and without the generation of merged cutting planes. The program adds these new cutting planes to the constraint matrix in a preprocessing step. Inequality merging is shown to reduce computational effort by about 9% on both small and larger MK instances, when compared to baseline CPLEX settings while using the recommended implementation strategies.

1.2.4 Theoretical Extensions

Two theoretical extensions are considered in detail in this dissertation. The first extension merges information from multiple (donor) inequalities into the original (host) inequality simultaneously. Theoretical conditions are developed that support validity, and the extension is demonstrated in an example problem. This extension is included as an optional implementation strategy in the computational study. In some instances, this theoretical extension yields even shorter computational times than the baseline inequality merging technique.

The second theoretical extension is the concept of sequentially merging multiple inequalities on several different variables yielding a single merged inequality. The Sequential Cover Merging Algorithm constructs merged inequalities of this type, and sufficient conditions are shown that guarantee validity. The power of this theoretical extension is shown in an example problem, since merged inequalities of this type have higher numbers of variables with non-zero coefficients.

It is likely that extensions to the baseline theory may be most beneficial when tailored to specific classes of IP problems. Additional topics are offered as promising areas for future research.

1.3 Outline

Chapter 2 provides an overview of background information that is important to understand the theoretical contributions contained in this dissertation. Topics covered in chapter 2 include integer programming, polyhedral theory, knapsack problems, classic cover cuts, and lifting. Treatment of these topics includes both formal definitions and examples that facilitate a better understanding of some concepts.

Chapter 3 introduces the general, theoretical foundations of inequality merging. The technique combines two or more low-dimensional inequalities potentially yielding a valid inequality of higher dimension. The dissertation offers theoretical conditions for validity of the merged inequality and shows that the validity of a merged cover inequality may be verified in quadratic time. Conditions under which a valid merged inequality is facet defining are also presented. The section concludes with an example that demonstrates these concepts and argues that these inequalities are not a simple extension from past research.

Chapter 4 extends the general results for inequality merging to cover inequalities from a MK instance. Theoretical conditions for validity are extended to cover inequalities. Construction techniques of merged inequalities in a MK instance are introduced, providing techniques to select indices as members of both the original (host) and merging (donor) inequalities. A theorem proves that adherence to the ψ_p construction method yields a valid merged inequality, and an algorithm provides a mechanism that reduces ψ_p for construction purposes when necessary. Chapter 4 concludes with the introduction of a theoretical extension that merges over multiple donor inequalities simultaneously, thus strengthening merged coefficients. The DCSA algorithm implements this extension, and an example problem is provided.

Chapter 5 provides the results of the computational study. Several implementation strategies are investigated, including pseudo-costing techniques, the number of cutting planes generated, consideration of overlapping rows, and using the coefficient strengthening extension. Significant experimentation on smaller problems motivates recommended implementation strategies, and further analysis is conducted on larger MK instances. Experimental results demonstrate that inequality merging reduces computational time in benchmark MK instances when recommended implementation strategies are followed.

Chapter 6 offers the theoretical foundations for a second extension to the general inequality merging theory, sequentially merging on multiple covers and variables. The SCMA implementation algorithm is provided, and a theorem proves that the SCMA provides sufficient conditions that the sequentially merged cutting plane is valid. An example problem demonstrates this extension.

Chapter 7 provides a summary of important contributions from this dissertation. This research serves as a foundation for a new technique that has already been shown to yield undiscovered classes of cutting planes. As such, several areas of future research are discussed. These topics are likely to yield strong results for a variety of real-world applications.

Chapter 2

Background Information

Chapter two provides the theoretical background information that is required to understand the new contributions in this dissertation. Chapter 2 includes a discussion of integer programming, polyhedral theory, and the generation and use of cutting planes to facilitate solution processes for IPs. This includes a more careful examination of knapsack problems and cover cutting plane inequalities because the dissertation focuses on these areas to develop a new category of cutting planes for MK problems. Chapter two also provides sufficient treatment of other strategies such as classic categories of cutting planes, lifting, and recent IP research to distinguish these new contributions from other known techniques.

2.1 Integer Programming and Polyhedral Theory

An Integer Program (IP) is defined as Maximize $c^T x$ subject to $Ax \leq b$, $x \geq 0$ and $x \in \mathbb{Z}^n$ where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. Define the set of indices of an integer program to be $N = \{1, ..., n\}$. One source for significant theoretical background on IPs is the popular text *Integer and Combinatorial Optimization* by Nemhauser and Wolsey [60]. Integer programming has served as a fertile ground for theoretical and applied research for several decades, including recent publications by Bonami et. al. in 2012 [15], Hemmecke et. al. [45] in 2011, and Luedtke et. al in 2010 [57].

The branch and bound technique is the standard method used to solve integer optimization problems [56]. This method eventually converges to the optimal solution, but it may take an exponential amount of time. The branch and bound technique builds a branching tree where the nodes have ancestral properties with each other.

Fundamental to the branch and bound technique is the understanding of a linear relaxation (LR). Given an integer program, the linear relaxation is the same problem without the integer restriction, thus $x \in \mathbb{R}^n$. Since the integral constraint has been removed, the integer program becomes a linear program. Consequently, the solution is found quickly through the simplex method or other LP solution technique.

The nodes of the branch and bound tree represent linear relaxation problems. The root node of the tree is the original LR. While there exists at least one unfathomed leaf node, the branch and bound algorithm solves the LR of that node with its corresponding x_{LR} and z_{LR}. If x_{LR} is integer, then the node is fathomed. Furthermore, if z_{LR} is better than the best current integer solution (Z^*_{IP}), then a better integer solution has been found, $Z^*_{IP} = z_{LR}$, and x_{LR} is saved as the corresponding integer solution.

If x_{LR} is not integer, the algorithm considers three options. If $z_{LR} < Z^*_{IP}$, then the tree is fathomed at that node. If x_{LR} is infeasible, then the tree is fathomed at that node. Otherwise, $z_{LR} \geq Z^*_{IP}$ and x_{LR} is not integer. In this case, there exists $p < x_i < p + 1$ with $p \in \mathbb{Z}$. Two children nodes are created from this node. The LR problem of one child is the LR of the parent with the additional constraint $x_i \leq p$. The LR of the other child requires $x_i \geq p + 1$.

There are several different strategies to examine the branch and bound tree, including breadth first, depth first, and best child. There are also a variety of other techniques that are sometimes employed during branch and bound implementation. Anstreicher et.al.

discuss heuristic searching and randomized diving in [4]. Achterberg and Berthold discuss hybrid branching including pseudo-cost branching in [2]. Techniques like these may improve solution times for branch and bound.

The addition of a new inequality called a cutting plane is another common strategy to improve the solution times of the branch and bound technique. There are a wide variety of well-known cutting plane techniques. Several famous examples include Gomory fractional cuts (Gomory [35],), Chvátal-Gomory cuts (Chvátal [18] and Gomory [35],) disjunctive cuts (Balas and Perregaard [9]), and superadditive cuts (Gomory and Johnson [37] and Wolsey [76].) Polyhedral theory is a fundamental concept in the development of cutting planes.

2.1.1 Polyhedral Theory

Solutions to IPs are highly dependent on the geometry of the problem, and polyhedral theory is fundamental to understanding integer programming research. A half space is $\{x \in \mathbb{R}^n : \sum_{i=1}^{n} a_i x_i \leq b\}$, and a polyhedron is defined as the intersection of finitely many half spaces.

A set $S \subseteq \mathbb{R}^n$ is convex if and only if x^1 and $x^2 \in S$ implies $\lambda x^1 + (1-\lambda)x^2 \in S$ for every $\lambda \in [0, 1]$. A polyhedron is convex, and the convex hull of S, $conv(S)$, is the intersection of all convex sets that contain S.

Define P as the set of feasible points of an integer program, $P = \{x \in \mathbb{Z}^n_+ : Ax \leq b\}$. An inequality $\sum_{i=1}^{n} \alpha_i x_i \leq \beta$ is valid for $conv(P)$ if every $x \in P$ satisfies this inequality. Every valid inequality induces a face, F, of $conv(P)$ having the form $\{x \in conv(P) : \sum_{i=1}^{n} \alpha_i x_i = \beta\}$. A proper face is neither $\emptyset$ nor the entire region, $conv(P)$. Every face is a polyhedron. Theoretically, the usefulness of a valid inequality is measured by the dimension of the induced face. The strongest such inequalities are facet defining and have a dimension one less than the dimension of $conv(P)$. A more complete discussion of a polyhedron's facets may be found in works by Wolsey, [74] and [75].

The dimension of a polyhedron equals the maximum number of affinely independent points minus one. Let V be a finite set of points in $\mathbb{R}^n$, $V = \{v^i \in \mathbb{R}^n : i = 1, ..., w\}$. The points in V are affinely independent if and only if the unique solution to $\sum_{i=1}^{w} \lambda_i v_i = 0$ and $\sum_{i=1}^{w} \lambda_i = 0$ is $\lambda_i = 0$ for all $i = 1, ..., w$. In order to define points in $\mathbb{R}^n$, let ξ_j be the origin in n dimensions translated one positive unit in the j^{th} dimension, i.e. $\xi_2 = \{0, 1, 0, ..., 0\}$.

2.1.2 2-Dimensional Integer Programming Example

In general, an integer programming problem may have arbitrarily large dimension. A two-dimensional IP example is shown below. This problem is used to demonstrate several theoretical concepts and solution techniques.

Example 2.1

$$\begin{aligned} \text{Maximize} \quad & z = x_1 + x_2 \\ \text{Subject to} \quad & 4x_1 + 2x_2 \leq 15 \\ & 2x_1 + 3x_2 \leq 13 \\ & x_1,\, x_2 \ \in \mathbb{Z}_+ \end{aligned}$$

Figure 2.1 offers a graphical perspective of this problem. Observe that the set of feasible integer points P are identified by the large circles. Five interesting points in the graph are given letter designations, including the two points where the constraints intersect the axes, two integer points on the exterior envelope of the convex hull, and the linear relaxation point.

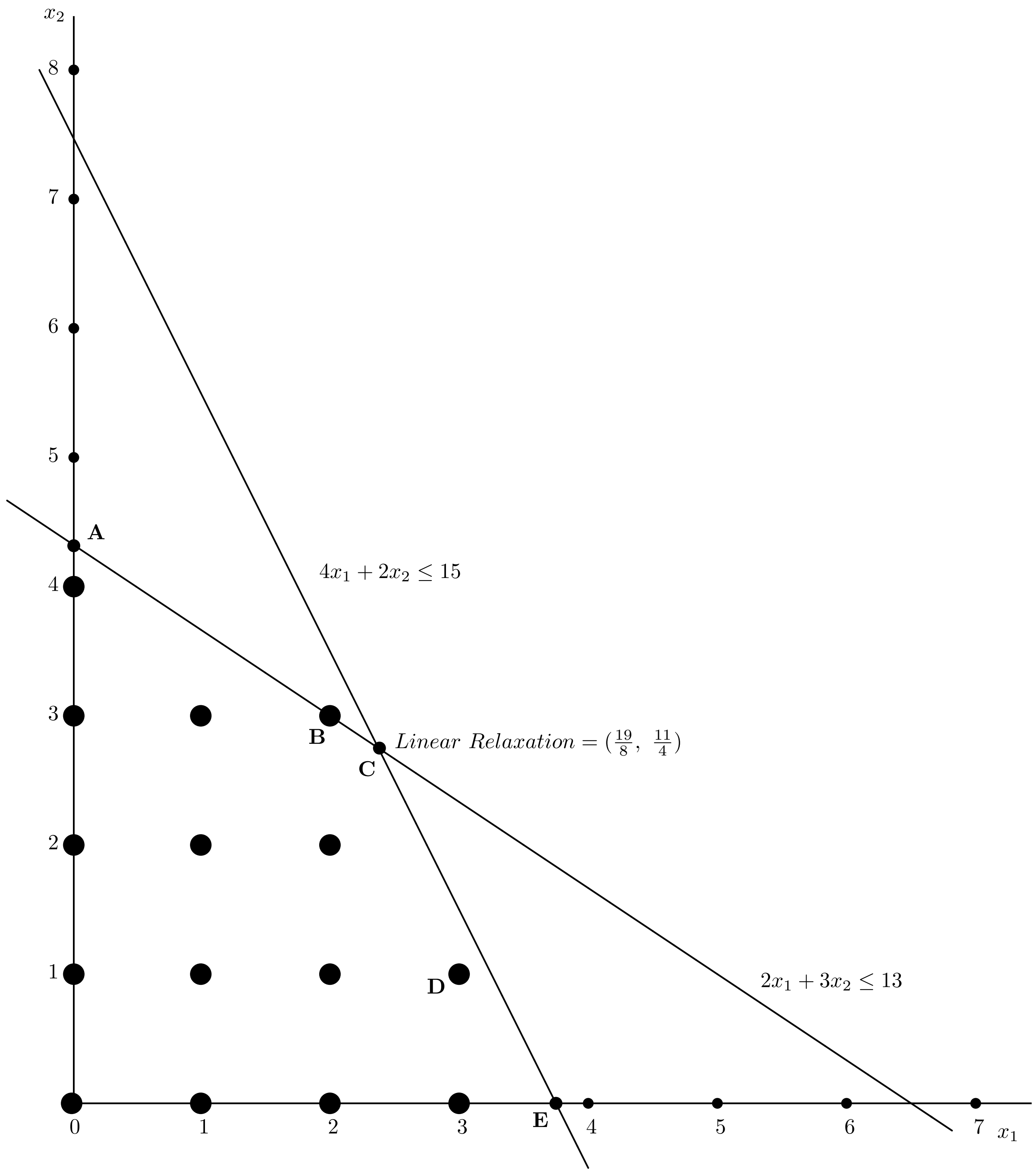

Figure 2.1: 2-Dimensional IP Example

The dimension of $conv(P)$ is clearly 2. More formally, the dimension of $conv(P)$ may be bounded from above since there are two variables (x_1 and x_2). The dimension of $conv(P)$ may also be bounded from below since there are three affinely independent points in P:

(0,0), (1,0), and (0,1). Thus the dimension of *conv*(P) must be 2. The first constraint, $4x_1 + 2x_2 \leq 15$ intersects the x_1 axis at point E, but it does not intersect any integral points in P. The second constraint, $2x_1 + 3x_2 \leq 13$ intersects the x_2 axis at point A and is incident to *conv*(P) at point B. Thus the second constraint is a face of dimension 0 (where the face is point B). Since the dimension of *conv*(P) is 2, the strongest faces in this problem are lines with dimension 1. The linear facets that define the convex hull of P are the following set of line segments: $\{[(0,0)$ to $(0,4)]$, $[(0,4)$ to $(2,3)]$, $[(2,3)$ to $(3,1)]$, $[(3,1)$ to $(3,0)]$, $[(3,0)$ to $(0,0)]\}$.

Since the objective function is $x_1 + x_2$, the best integral solution to this problem is at $(2,3)$, identified as point B, achieving $z^{IP} = 5$. If the integral constraint $x_1,\ x_2\ \in \mathbb{Z}_+$ is removed, the best solution is the linear relaxation (LR) at point C. Observe that $z^{LR} = \frac{41}{8}$ with $x_{LR} = (\frac{19}{8}, \frac{11}{4})$, where $z^{LR} > z^{IP}$.

As discussed in Section 2.1, a well-known technique to improve the branch and bound solution times for IP problems is the generation of valid inequalities that separate the linear relaxation solution from the convex hull of the IP. Such inequalities are known as cutting planes. The theoretically best cutting planes define facets of *conv*(P^{MK}), but any cutting plane that separates the linear relaxation from *conv*(P^{MK}) may be computationally useful.

The problem in Figure 2.1 can be solved by finding a cutting plane that separates the linear relaxation from the *conv*(P). In this case, let the cutting plane be the inequality $2x_1 + x_2 \leq 7$. Observe that this cut would include the portion of the convex hull that connects points B and D. With the inclusion of this inequality, the new $z^{LR} = 5$ at the point B. Since B is integral, $z^{IP} = z^{LR} = 5$, and the IP has been solved.

It can be demonstrated that the cutting plane $2x_1 + x_2 \leq 7$ is facet defining. Earlier, it was shown that *dim*(*conv*$(P)) = 2$. By inspection, it is trivially seen that $2x_1 + x_2 \leq 7$ is valid because no $x \in P$ has $\alpha^T x > \beta$. Now the dimension of the induced face must be determined, where $F = \{X \in$ *conv*$(P) : 2x_1 + x_2 = 7\}$. Since the point (0,0) is in P but it is not on the induced face, the dimension of the induced face is not the whole space. Thus $dim(F) \leq 1$. Two affinely independent points on the induced face are (2,3) and (3,1), corresponding to points B and D. Thus, $dim(F) \geq 1$. Therefore, $dim(F) = 1$, which is one less than the $dim(conv(P))$. Thus the cutting plane is facet defining.

2.2 Knapsack Problems

A commonly studied category of integer programs is the 0-1 knapsack problem (KP), defined as Maximize $\sum_{j=1}^{n} c_j x_j$ subject to $\sum_{j=1}^{n} a_j x_j \leq b$, and $x_j \in \{0,1\}$ for all $j \in N$ where c and $a \in \mathbb{R}^n_+$, $b \in \mathbb{R}_+$. The multiple knapsack (MK) problem is an integer program with a finite, m, number of knapsack constraints. Formally MK is defined as Maximize $c^T x$ subject to $Ax \leq b$ and $x \in \{0,1\}^n$ where $c \in \mathbb{R}^n_+$, $A \in \mathbb{R}^{m \times n}_+$, and $b \in \mathbb{R}^m_+$. General solution techniques may be found in Martello and Toth [58].

Recall that P is the set of feasible points of an integer program. Define the feasible points of a knapsack problem to be $P^{KP} = \{x \in \{0,1\}^n : a^T x \leq b\}$ where $a \in \mathbb{R}^n_+$. This dissertation focuses on the multiple knapsack problem with the corresponding feasible region defined as $P^{MK} = \{x \in \{0,1\}^n : Ax \leq b\}$ with $A \in \mathbb{R}^{m \times n}_+$. Valid inequalities are of the form $\sum_{j=1}^{n} \alpha_j x_j \leq \beta$ where every $x \in P^{MK}$ satisfies this inequality for $conv(P^{MK})$. All valid inequalities of $conv(P^{MK})$ have positive coefficients. The $conv(P^{MK})$ is full dimensional under standard assumptions that each $a_{i,j} \in A$ satisfies $a_{i,j} \leq b_i$ for all $i = 1, ..., m$ and $j = 1, ..., n$.

The 0-1 knapsack problem is appropriately named, motivated by the concept that a hiker would need to select which items to bring in their knapsack before leaving on a camping trip. Each of the n candidate items provides a corresponding value, and the hiker seeks to maximize the total value of the selected items constrained by practical considerations.

The knapsack problem can be modeled as an integer program. If item x_i is to be selected in IP formulations, then $x_i = 1$. If item x_i is not selected, then $x_i = 0$ for each $i \in N$. A basic knapsack problem has only one constraint, such as the cubic volume of the knapsack. A multiple knapsack problem has 2 or more constraints, including additional concerns such as the weight of the items being carried or the total cost of purchasing the items. Consequently, each candidate item j has attributes such as its value to the camper (c_j) and three $a_{i,j}$ coefficients concerning the knapsack size in cubic inches, ($a_{1,j}$), the weight of candidate items in pounds ($a_{2,j}$), and each item's price in dollars ($a_{3,j}$).

Beyond the namesake example, both the KP and MK problems have many important applications. Numerous researchers have developed solution techniques for KP and MK problems in a wide range of practical examples. These include project/portfolio selection problems by Chang and Lee [16], production planning and inventory problems by Dawande et al.[23], machine scheduling techniques by Kellerer and Strusevich [54], profit maximization applications by Dizdar et. al. [27] and Szeto and Lo [70] and storage management/packing problems by Shachnai and Tamir [68]. Large neighborhood search techniques for MK problems by Ahuja and Cunha may be seen in [3].

Since the theoretical contributions from this research focus on the multiple knapsack problem, the next section provides an example that provides a detailed discussion of KP and MK instances.

2.2.1 Example Knapsack/Multiple Knapsack Problem

Suppose that a PhD student decides to go on a week-long backpacking trip over Spring Break. The student determines a list of 22 candidate items to bring on the trip. The information in Table 2.1 shows the benefit (value score) of each item, along with the weight, volume, and purchasing price of each.

First, consider a KP problem with one constraint. In this instance, assume that the stu-

Index #	Item	Value	Weight (lbs)	Size (in^3)	Cost ($)
1	Food	75	14	470	32
2	Water Bottle	57	4	30	4
3	2nd Water Bottle	45	4	30	4
4	Tent	36	9	250	114
5	Sleeping Bag	53	5	335	75
6	2nd Set Clothes	28	2	155	50
7	3rd Set Clothes	19	2	155	50
8	First Aid Kit	40	0.5	25	6
9	Camera	20	1	15	180
10	Pillow	22	0.5	200	5
11	Flashlight	32	2	13	22
12	Spare Batteries	14	2	28	7
13	GPS	10	1	10	150
14	Cooking Pot	34	5	145	16
15	Stove Set	27	6	125	65
16	Matches	40	0.5	4	2
17	Hygiene Kit	43	2	70	10
18	Spare Shoes	27	3	100	60
19	Rain Jacket	39	3	50	40
20	Fleece Jacket	21	2	120	35
21	Water Purification Tabs	45	0.5	5	13
22	Knife	48	2	14	55

Table 2.1: Multiple Knapsack Data

dent is very strong and can carry all 22 items (71 pounds). Also, assume that the student already owns the equipment and does not have to make any purchases. In this case, the student's only concern is maximizing the value of the selected items as long as they fit in the knapsack with a capacity of 1,500 cubic inches. This KP problem is shown in Example 2.2 below.

Example 2.2

$$\text{Maximize} \sum_{i=1}^{22} c_j x_j$$

$$\text{Subject to } \sum_{i=1}^{22} a_{1,j}\ x_j \leq 1,500$$

$$x_j \ \in \ \{0,1\}^{22}$$

The optimal solution for the KP is $\{1,1,1,0,1,1,0,1,1,0,1,1,1,1,0,1,1,1,1,0,1,1\}$ with $z^{KP} = 650$. In this instance, the student brings every item except for the tent, the 3rd set of clothes, the pillow, the cooking stove, and the fleece jacket. This makes sense since the solution avoids relatively low-value items that have relatively larger cubic volumes.

The problem is a MK if all three constraints are considered. Suppose that the student needs to purchase all the equipment for the trip with a budget of only \$500 to purchase supplies. The student also decides to restrict the total weight of the items to 60 pounds. The problem is now much more strictly constrained because the student can only afford about half of the items. The student also needs to remove at least 11 pounds from the packing list. The MK problem is shown in Example 2.3.

Example 2.3

$$\text{Maximize} \sum_{j=1}^{22} c_j x_j$$

$$\text{Subject to } \sum_{j=1}^{22} a_{1,j}\ x_j \leq 1,500$$

$$\sum_{j=1}^{22} a_{2,j}\ x_j \leq 500$$

$$\sum_{j=1}^{22} a_{3,j}\ x_j \leq 60$$

$$x_j\ \in\ \{0,1\}^{22}$$

The optimal solution for the MK problem is $\{1,1,1,1,0,0,0,1,0,0,1,1,0,1,1,1,1,1,1,1,1,1\}$ with $z^{MK} = 623$. Notice that there are several changes from the previous solution. Now the student brings all the items on the packing list except for the sleeping bag, the 2nd and 3rd set of clothes, the camera, the pillow, and the GPS. Observe that the student is now bringing the tent, the cooking stove, and the fleece jacket (all previously omitted) because of the more strictly constrained environment.

2.3 Classic Cover Cuts

One popular type of cutting planes for KP and MK problems is called a cover cut. Cover cuts may decrease solution times for KP and MK problems in many instances. Knowledge of cover cuts is critical to this research because several theoretical contributions of this dissertation use cover cuts to generate more complex inequalities. Cover cuts have been studied extensively by Balas and Zemel [10], De Farias et al. [22], Hunsaker and Tovey [50], Nemhauser and Vance [61], and Park [63]. For a MK problem, a cover cut may be generated in one or more of the m constraints.

A set $C \subseteq N$ is a cover for row $i \in \{1,...,m\}$ if $\sum_{j\in C} a_{i,j} > b_i$. The corresponding cover inequality is valid for $conv(P^{MK})$ and takes the form $\sum_{j\in C} x_j \leq |C| - 1$. A cover C is a minimal cover if $C \setminus \{j\}$ is not a cover for all $j \in C$. If $C \subseteq N$ is a cover, define an extended cover as $E(C) = C \cup \{j \in N \; |a_j \geq a_i \; , \forall i \in C\}$. The valid inequality of the extended cover is $\sum_{j \in E(C)} x_j \leq |C| - 1$.

Consider the backpacking KP scenario shown in Table 2.1, and recall that the capacity of the student's knapsack is 1,500 cubic inches. Examining the largest candidate items, one cover would be $\{1, 5, 4, 10, 6, 7\}$ since the volume of these 6 items is 1,565 cubic inches. This means that a backpacker could choose at most 5 of these 6 items, and the corresponding cutting plane would be $x_1 + x_4 + x_5 + x_6 + x_7 + x_{10} \leq 5$. This cover is minimal because the removal of any of these items would no longer constitute a cover. Another minimal cover in the example KP problem is $\{4, 5, 6, 7, 10, 14, 15, 18, 20\}$ with valid inequality $x_4 + x_5 + x_6 + x_7 + x_{10} + x_{14} + x_{15} + x_{18} + x_{20} \leq 8$.

Many such covers exist for any given constraint in the MK instance. The choice of variables depends on which pseudo-costing strategy is employed. Pseudo-costing strategies for integer programming problems were studied by Benichou, et. al. in [14] and Gauthier and Ribiere in [34]. The choice of pseudo-cost strategy designates which variables are more or less desirable for inclusion in additional cutting planes. Refalo used pseudo-cost strategies to improve constraint programming [65], and Achterberg, et. al. developed reliability branching rules for IPs as an extension of pseudo-costing in [1].

Pseudo-costing strategies that sort by the volume requirements of each item may choose largest items first, and that strategy yields the original cover $\{1, 5, 4, 10, 6, 7\}$ with the corresponding inequality $x_1+x_5+x_4+x_{10}+x_6+x_7 \leq 5$. However, if the pseudo-costing technique sorts by the value of each item, the associated cover is $\{1, 2, 5, 22, 3, 21, 17, 8, 16, 19, 4, 18, 20\}$ with the corresponding inequality $x_1 + x_2 + x_5 + x_{22} + x_3 + x_{21} + x_{17} + x_8 + x_{16} + x_{19} + x_4 + x_{18} + x_{20} \leq 12$.

Because of the large disparity between the volumes of different items, great care must be followed to ensure that a valid cover is attained. Observe that the last two items (indices

18 and 20) were selected out of order because their inclusion in the cover took the total sum of volumes in the cover to be exactly 1,503 cubic inches. Since the smallest volume of an item in the cover was 4 cubic inches (index 16), this took a careful selection of indices for the last two terms. Clearly, the size and composition of the cover can vary significantly as pseudo-costing strategies change.

2.4 Lifting

In some instances, cover inequalities may be strengthened through lifting. Additionally, some of the research contributions in this dissertation use restricted polyhedra similar to lifting. Although the contributions of this dissertation are distinctly different from known lifting techniques, some knowledge of lifting is helpful to understand the newly discovered category of cutting planes in this research.

Lifting, introduced by Gomory [36], takes a valid inequality of a restricted space and tilts it to become a valid inequality of a higher dimensional space. Formally, let $C \subset N$ and $K \in \mathbb{Z}^{|N \setminus C|}$, then the restricted IP for C is Maximize $c^T x$ subject to $Ax \leq b$, $x_i = k_i$ for all $i \in N \setminus C$, $x \geq 0$ and $x \in \mathbb{Z}^n$. Its feasible region is denoted as $P_{C,K} = \{x \in \mathbb{Z}^n_+ : Ax \leq b, x_i = k_i, \forall i \in N \setminus C\}$ and the corresponding integer polyhedron is $conv(P_{C,K})$. The newly discovered category of cutting planes in this dissertation merge restricted inequalities from polyhedra with $K = 0$. For notational convenience, define $conv(P_C)$ to be $conv(\{x \in \mathbb{Z}^n_+ : Ax \leq b, x_i = 0, \forall i \in N \setminus C\})$.

Lifting requires a set $C \subset N$, $K \in \mathbb{Z}^{|N \setminus C|}$ and a valid inequality $\sum_{i \in C} \alpha_i x_i + \sum_{i \in N \setminus C} \alpha_i x_i \leq \beta$ of $conv(P_{C,K})$. Lifting then creates a valid tilted inequality of the form $\sum_{i \in C} \alpha_i x_i + \sum_{i \in N \setminus C} \alpha'_i x_i \leq \beta'$ of $conv(P)$. The standard goal of lifting is to theoretically strengthen the original inequality by increasing its dimension.

There are several categories of lifting. Inequalities can either be generated sequentially

$|N \backslash C| = 1$ or simultaneously $|N \backslash C| \geq 2$. The coefficients can be either exact or approximate for α and β values. The restricted polyhedron can have variables fixed at the lower bounds for up lifting, at the upper bounds for down lifting, or someplace in between for middle lifting.

Numerous researchers have done work on sequential exact up lifting, such as Cho et al. [17], Gutierrez [41], Hammer et al. [42], and Wolsey [74]. Exact sequential lifting finds the maximum value of α_1 that maintains validity. Thus, one benefit of this technique is that it guarantees that the dimension of the face induced by the sequentially lifted inequality increases by at least 1 if such a value of α_1 may be obtained. While maintaining validity, the process may be repeated sequentially for α_2, α_3, etc. In such cases, exact sequential lifting increases the dimension of the induced face by even more.

In some instances, this process can transform a facet defining inequality from the restricted space into a facet defining inequality of the full space. However, one disadvantage is that exact lifting typically requires the solution of an integer program or multiple integer programs. Furthermore, the coefficients obtained are restricted to integer values. Exact lifting has been the topic of significant research because of its potential to yield strengthened inequalities. Examples include papers by Easton and Hooker [31], Kubik [55], and Zemel [77].

Sequence dependent lifting can be classified as an approximate simultaneous up lifting technique, shown in Atamtürk [7], Gu et al. [38], [39], and [40], and Shebalov and Klabjan [69]. Simultaneous lifting may yield stronger fractional α coefficients for several variables at the same time, requiring the solution of a single integer program. However, simultaneous lifting may not increase the dimension of the induced face in general. Rather, this technique often lends itself to computational benefits since simultaneous lifting strengthens α coefficients during preprocessing.

There are a variety of extensions and other categories of lifting. Additional approximate lifting methods can be found in Balas [8] and Weismantel [72]. Wolsey [74] also provides exact sequential down and middle lifting. More recently, Atamtürk and Narayanan generalized the theory of lifting to conic integer programming. [5]

2.5 Inequality Merging

This dissertation research provides the theoretical foundations for generating a new class of valid inequalities for integer programming problems through a process named inequality merging. Dey and Richard introduced the idea of sequential-merge facets in two-dimensional group problems in 2007 [26]. Their paper includes an operation that combines two facet-defining inequalities of one-dimensional group problems, creating a facet-defining inequality for two-dimensional group problems. In 2010, Dey and Wolsey developed two row mixed integer cuts through lifting [25]. Their process uses lifting functions to obtain integer coefficients for new cuts that combine information from two rows of a simplex tableau. Other than a similar naming scheme, the contributions in this dissertation are different because they apply to general KP and MK problems with arbitrary dimension. This technique also differs from Dey and Wolsey because their work uses lifting directly while this research generates cutting planes that are not readily attainable by direct implementation of known lifting techniques.

Inequality merging yields a new type of valid cutting plane inequalities for MK problems that are fundamentally different from other types of cutting planes, lifting techniques, or other known efforts to merge information from existing inequalities. The next chapter introduces the theoretical foundations of inequality merging including conditions for validity and conditions under which a merged inequality may be facet defining. Chapter 3 also provides an example problem that demonstrates inequality merging over the MK polyhedron.

Chapter 3

Theoretical Foundations of Inequality Merging

The bulk of the results in this chapter can also be found in my co-authored paper that has been accepted for publication in the *International Journal of Operations Research* [46].

The idea behind inequality merging is to combine information from two valid inequalities in a manner that yields a single, higher-dimensional valid cutting plane for integer programming problems. Designate the two low-dimensional inequalities as the host and donor inequalities. The basic form and the right-hand side of the merged inequality are defined by the original host inequality. In the process of merging, the host inequality replaces one or more of its terms with a collection of terms attained from the donor inequality. Validity is maintained by adjusting the coefficients of the terms gained from the donor inequality. In many cases, the result is a higher-dimensional cutting plane that merges information from both the host and donor inequalities.

The merging process is not symmetric, and care must be employed to designate the host and donor inequalities appropriately. Furthermore, merged inequalities of this type are not always attainable because of the need to preserve validity across all constraints. However, the method is shown to be particularly useful for instances such as the MK problem when it is relatively easy to find valid inequalities (e.g. cover inequalities) of low dimension, but somewhat more difficult to find higher-dimensional cutting planes.

Creating a merged inequality on a binary variable x_p requires $C^1 \subset N$, $p \in C^1$ and $C^2 \subseteq (N \backslash C^1) \cup \{p\}$. More formally, define the host inequality, $\sum_{j \in C^1} \alpha^1_j x_j \leq \beta^1$, as a valid inequality of $conv(P^{MK}_{C^1})$ such that β^1 and each α^1_j is a nonnegative integer for all $j \in C^1$ and $\alpha^1_p = 1$. Furthermore, let the donor inequality, $\sum_{j \in C^2} \alpha^2_j x_j \leq \beta^2$, be a valid inequality of $conv(P^{MK}_{C^2})$. Then the merged inequality is $\sum_{j \in C^1 \backslash \{p\}} \alpha^1_j x_j + \sum_{j \in C^2} \frac{\alpha^2_j}{\beta^2} x_j \leq \beta^1$.

The basic idea of inequality merging is that the merged variable, x_p, is bounded by the value 1. Since $\sum_{j \in C^2} \alpha^2_j x_j \leq \beta^2$ is valid, $\sum_{j \in C^2} \frac{\alpha^2_j}{\beta^2} x_j \leq 1$. Thus, it may be possible to replace x_p by this fractional inequality, creating a merged inequality. Several obvious questions arise, the first of which concerns the validity of such an inequality. The next section provides the theoretical foundations for this question.

3.1 Validity of Merged Inequalities

This section begins with a theorem providing conditions under which a merged inequality is valid. The result is extended to merging with a host cover inequality. Using this concept, the section concludes with a polynomial time algorithm to determine the validity of a merged cover inequality.

Theorem 1. *Let $C^1 \subset N$ with $p \in C^1$ have a corresponding valid host inequality $\sum_{j \in C^1} \alpha^1_j x_j \leq \beta^1$ of $conv(P^{MK}_{C^1})$ and $C^2 \subseteq (N \backslash C^1) \cup \{p\}$ have a valid donor inequality of $conv(P^{MK}_{C^2})$ of the form $\sum_{j \in C^2} \alpha^2_j x_j \leq \beta^2$. Furthermore, assume α^1 and β^1 are nonnegative integer coefficients* *with* $\alpha^1_p = 1$, $\alpha^2 \geq 0$ *and* $\beta^2 > 0$. *Let* $F = \{x \in P^{MK}_{C^1} : \sum_{j \in C^1} \alpha^1_j x_j = \beta^1,\ x_p = 0\}$. *If* $y + \xi_q$ *is infeasible for all* $y \in F$ *and* $q \in C^2$, *then the merged inequality*

$$\sum_{j \in C^1 \setminus \{p\}} \alpha^1_j x_j + \sum_{j \in C^2} \frac{\alpha^2_j}{\beta^2} x_j \leq \beta^1$$

is a valid inequality of $conv(P^{MK}_{C^1 \cup C^2})$.

Proof. Let x' be any point in $P^{MK}_{C^1 \cup C^2}$. Due to the fact that $\sum_{j \in C^1} \alpha^1_j x_j \leq \beta^1$ is valid over $conv(P^{MK}_{C^1})$, $\sum_{j \in C^1 \setminus \{p\}} \alpha^1_j x'_j \leq \beta^1$. The proof divides into two cases because α^1 and β^1 are nonnegative integer coefficients with $\alpha^1_p = 1$. First, assume $\sum_{j \in C^1 \setminus \{p\}} \alpha^1_j x'_j = \beta^1$. Let y' be x' projected onto F. By assumption, $y' + \xi_q$ is infeasible for each $q \in C^2$. Thus, $\sum_{j \in C^1 \setminus \{p\}} \alpha^1_j x'_j + \sum_{j \in C^2} \frac{\alpha^2_j}{\beta^2} x'_j \leq \beta^1$. Second, assume $\sum_{j \in C^1 \setminus \{p\}} \alpha^1_j x'_j < \beta^1$. Since x' and each α'_j are integer, $\sum_{j \in C^1 \setminus \{p\}} \alpha^1_j x'_j \leq \beta^1 - 1$. Because $\sum_{j \in C^2} \alpha^2_j x_j \leq \beta^2$ is valid over $conv(P^{MK}_{C^2})$, it implies that $\sum_{j \in C^2} \frac{\alpha^2_j}{\beta^2} x_j \leq 1$. Thus $\sum_{j \in C^1 \setminus \{p\}} \alpha^1_j x'_j + \sum_{j \in C^2} \frac{\alpha^2_j}{\beta^2} x'_j \leq \beta^1$, and the result follows.

□

Checking the conditions of Theorem 1 may not be easy for all merged inequalities. Restricting the host inequality to a cover inequality enables a polynomial time algorithm to verify validity. If $C^1 \setminus \{p\} \cup \{j\}$ is a cover in at least one of the MK's constraints for each $j \in C^2$, then the conditions of Theorem 1 are met.

An important extension is that verifying the validity of merging over a host cover inequality in a multiple knapsack instance can now be completed in $O(m(|C^1| + |C^2|))$ effort. The algorithm verifies that each $C^1 \setminus \{p\} \cup \{j\}$ is a cover in some constraint for all $j \in C^2$.

Merging Over a Cover Inequality Algorithm (MOCIA)

```
a. Initialization:

        Set $d_i$ to $\sum_{j \in C^1 \setminus \{p\}} a_{ij}$ for all $i = 1, ..., m$
        Set $validflag$ to $True$

b. Main Step:

        For each $j \in C^2$

            Set $flag$ to $False$

            For each $i = 1, ..., m$

                If $a_{ij} + d_i > b_i$, then set $flag$ to $True$

            end $for$

            if $flag = False$,  Then

                set $validflag$ to $False$

        end $for$

c. Output:

        Report $validflag$
```

The Initialization clearly requires $O(m|C^1|)$ effort due to the first step. The main step requires $O(m|C^2|)$ effort and reporting requires $O(1)$. Thus the MOCIA is extremely fast and runs in $O(m(|C^1| + |C^2|))$ effort. This is clearly a linear time algorithm for a knapsack problem. Observe that the MOCIA is bounded by $O(mn)$. Furthermore, reading in a generic multiple knapsack instance requires reading in A with mn elements. Thus, every algorithm on a generic multiple knapsack must be $\Omega(mn)$. Consequently, the MOCIA is $\Theta(mn)$.

Both the merging process and the verification of validity can be accomplished with minimal computational effort. Consequently, merged inequalities are now readily accessible and easily implemented by practitioners solving complex multiple knapsack problems.

3.2 Theoretical Strength of Merged Inequalities

Merging inequalities can create higher dimensional faces, and the dimension of the inequality's face defines its theoretical strength. The strongest such inequalities define facets of $conv(P^{MK})$. Theoretically, stronger inequalities tend to eliminate more linear relaxation space when applied as cutting planes. The following theorem demonstrates that under certain general conditions a valid merged inequality defines a higher dimensional face, which may be a facet.

Theorem 2. *Let $\sum_{j\in C^1} \alpha_j^1 x_j \leq \beta^1$ be a face of dimension η over the restricted space $conv(P_{C^1}^{MK})$ and let $\sum_{j\in C^2} \alpha_j^2 x_j \leq \beta^2$ be a face of dimension ϕ over $conv(P_{C^2}^{MK})$ such that the inequality merging technique on x_p yields a valid inequality of the form $\sum_{j\in C^1} \alpha_j^1 x_j + \sum_{j\in C^2} \frac{\alpha_j^2}{\beta^2} x_j \leq \beta^1$ of $conv(P_{C^1\cup C^2}^{MK})$. If the following 2 conditions hold, then the merged inequality has a face of dimension at least $\eta + \phi$ in $conv(P_{C^1\cup C^2}^{MK})$.*

i. There exists a point $x^{2'} \in F^2 = \{x \in P_{C^2}^{MK} : \sum_{j\in C^2} \alpha_j^1 x_j = \beta^2\}$ such that $x^1 - \xi_p + x^{2'} \in P_{C^1\cup C^2}^{MK}$ for each $x^1 \in \{x \in P_{C^1}^{MK} : x_p = 1\}$.

ii. There exists a point $x^{1'} \in F^1 = \{x \in P_{C^1}^{MK} : \sum_{j\in C^1} \alpha_j^1 x_j = \beta^1, x_p = 1\}$ such that

$x^{1'} - \xi_p + x^2 \in P^{MK}_{C^1 \cup C^2}$ *for each* $x^2 \in P^{MK}_{C^2}$.

Proof. By assumption, $\sum_{j \in C^1} \alpha^1_j x_j + \sum_{j \in C^2} \frac{\alpha^2_j}{\beta^2} x_j \leq \beta^1$ is valid over $conv(P^{MK}_{C^1 \cup C^2})$. Since $\sum_{j \in C^1} \alpha^1_j x_j \leq \beta^1$ defines a face of dimension η in $conv(P^{MK}_{C^1})$, there exist at least $\eta + 1$ affinely independent points $q^1_1, q^1_2, ..., q^1_{\eta+1} \in P^{MK}_{C^1}$ that meet $\sum_{j \in C^1} \alpha^1_j x_j \leq \beta^1$ at equality. Similarly, there exist at least $\phi + 1$ affinely independent points $q^2_1, q^2_2, ..., q^2_{\phi+1} \in P^{MK}_{C^2}$ that meet $\sum_{j \in C^2} \alpha^2_j x_j \leq \beta^2$ at equality.

Without loss of generality and by assumption *ii* there exists some i such that $q^1_1, ..., q^1_i$ have $x_p = 0$ and $q^1_{i+1}, ..., q^1_{\eta+1}$ have $x_p = 1$ where $i = 0, ..., \eta$. Notice that if $i = 0$, then all q^j have $x_p = 1$ for $j = 1, ..., \eta + 1$. Consider the following $\eta + \phi + 1$ points that are clearly feasible by assumptions *i* and *ii*. Without loss of generality, assume that $x^{2'}$ is q^2_1 and $x^{1'}$ is q^1_{i+1}. For clarity, these points are divided into three sets.

a. The first set of points are $q^1_1, ..., q^1_i$.

b. The second set of points are $q^1_{i+1} - \xi_p + q^2_1, q^1_{i+2} - \xi_p + q^2_1, ..., q^1_\eta - \xi_p + q^2_1$.

c. The third set of points are $q^1_{i+1} - \xi_p + q^2_2,\ q^1_{i+1} - \xi_p + q^2_3, ...,\ q^1_{i+1} - \xi_p + q^2_{\phi+1}$.

It is trivial to verify that each of these points satisfy $\sum_{j \in C^1} \alpha^1_j x_j + \sum_{j \in C^2} \frac{\alpha^2_j}{\beta^2} x_j = \beta^1$. Thus, these $\eta + \phi + 1$ points are on the face of the merged inequality. It remains to show that these points are affinely independent. The second set contains the point $y' = x^{1'} - \xi_p + x^{2'}$.

Perform the column operation of subtracting this point, $y^{'}$, from each point in the third set. The third set of points now take the form

$$q^1_{i+1} - \xi_p + q^2_j - y^{'} =$$

$$q^1_{i+1} - \xi_p + q^2_j - (x^{1'} - \xi_p + x^{2'}) =$$

$$q^1_{i+1} - \xi_p + q^2_j - (q^1_{i+1} - \xi_p + q^2_1) =$$

$$q^2_j - q^2_1$$

for each $j = 2, ..., \phi+1$. These new columns are clearly linearly independent following a well-known result found in various texts such as Nemhauser and Wolsey [60]. This result defines one affinely independent point, q^2_1, as the origin and then creates vectors by subtracting this new 'origin' from the other affinely independent points.

These operations yield a collection of points that are lower block diagonal. Since the points in sets one and two are affinely independent, the top block of these points are affinely independent and the bottom block is linearly independent as previously argued. Thus these $\eta+\phi+1$ points are affinely independent and so the merged inequality has a face of dimension at least $\eta + \phi$ in $conv(P^{MK}_{C^1 \cup C^2})$, concluding the proof.

□

Restricting both C^1 and C^2 to minimal covers provides a set of well defined conditions where facet defining inequalities are created. It is straightforward to show that merging over two minimal cover inequalities meets at least one of the conditions of Theorem 2. Thus, merging two minimal cover inequalities may induce a facet over the merged space with the additional conditions provided in the following corollary.

Corollary 1. *Let C^1 and C^2 be two minimal covers from possibly two distinct constraints that result in a valid merged inequality of $conv(P^{MK}_{C^1 \cup C^2})$ with the form*

$$\sum_{i \in C^1 \setminus \{p\}} x_i + \frac{1}{|C^2| - 1} \sum_{j \in C^2} x_j \leq |C^1| - 1.$$

If the following conditions are met:

i. The set $C^1 \setminus \{p\}$ is not a cover of any constraint.

ii. There exists an $s' \in C^2$ such that the sets $(C^1 \setminus \{r\} \setminus \{p\}) \cup (C^2 \setminus \{s'\})$ are not covers for each $r \in C^1 \setminus \{p\}$.

iii. There exists an $r' \in C^1 \setminus \{p\}$ such that the sets $(C^1 \setminus \{r'\} \setminus \{p\}) \cup (C^2 \setminus \{s\})$ are not covers for each $s \in C^2$,

then the merged inequality is facet defining over $conv(P^{MK}_{C^1 \cup C^2})$.

$\square$

The following example demonstrates the inequality merging technique in a multiple knapsack instance. The example includes the implementation of the MOCIA, the demonstration of validity consistent with Theorem 1, and the proof that the merged inequality is facet defining consistent with Theorem 2 and Corollary 1.

3.3 Inequality Merging Example

Consider the following multiple knapsack instance, with (3.1) and (3.2) as multiple knapsack constraints

$$13x_1 + 12x_2 + 11x_3 + 5x_4 + 3x_5 + 2x_6 + 2x_7 + 1x_8 \leq 36 \tag{3.1}$$

$$2x_1 + 4x_2 + 1x_3 + 7x_4 + 6x_5 + 8x_6 + 6x_7 + 5x_8 \leq 30 \tag{3.2}$$

and $x \in \{0,1\}^8$.

Let $C^1 = \{1,2,3,4\}$, $C^2 = \{4,5,6,7,8\}$, and $p = 4$. Since C^1 is a cover in (3.1) and C^2 is a cover in (3.2), (3.3) is a valid inequality of $conv(P^{MK}_{C^1})$ and (3.4) is a valid inequality of $conv(P^{MK}_{C^2})$.

$$x_1 + x_2 + x_3 + x_4 \leq 3 \tag{3.3}$$

$$x_4 + x_5 + x_6 + x_7 + x_8 \leq 4 \tag{3.4}$$

Merging the host (3.3) with the donor (3.4) on x_4 yields

$$x_1 + x_2 + x_3 + \frac{1}{4}x_4 + \frac{1}{4}x_5 + \frac{1}{4}x_6 + \frac{1}{4}x_7 + \frac{1}{4}x_8 \leq 3. \tag{3.5}$$

The validity of (3.5) is shown by verifying that setting all of the variables associated with $C^1 \setminus \{4\} \cup \{k\}$ to 1 is infeasible (a cover for some constraint for every $k \in C^2$). The sum of the $a_{1,j}$ coefficients for $j \in C^1 \setminus \{4\}$ is already equal to the righthand side, b_1, of (3.1), and all the $a_{1,j}$ coefficients in C^2 are positive in (3.1). Thus, $min_{k \in C^2} \sum_{j \in C^1 \setminus \{4\} \cup \{k\}} a_{1,j} > b_1$. This implies that $C^1 \setminus \{4\} \cup \{k\}$ is a cover in (3.1) for every $k \in C^2$. Consequently, the results of Theorem 1 show that (3.5) is a valid inequality of $conv(P^{MK}_{C^1 \cup C^2})$.

Following the result of Theorem 2, it can be shown that (3.5) is also facet defining. Observe that (3.3) induces a face of dimension $\eta = 3$ over the $conv(P_{C^1}^{MK})$. Four affinely independent points that meet (3.3) at equality are shown in Figure 3.1.

$$\left|\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array}\right| = \left|\begin{array}{cccc} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right|$$

Figure 3.1: Four Affinely Independent Points in (3.3)'s Face in $P_{C^1}^{MK}$

Note that the first column of Figure 3.1 is the only column with $x_4 = 0$. This implies $i = 1$ in Theorem 2. Similarly, (3.4) has a face of dimension $\phi = 4$ over $conv(P_{C^2}^{MK})$. Five affinely independent points that meet (3.4) at equality are shown in Figure 3.2.

$$\left|\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array}\right| = \left|\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{array}\right|$$

Figure 3.2: Five Affinely Independent Points in (3.4)'s Face in $P_{C^2}^{MK}$

Let $x^{1'} = (1,0,1,1,0,0,0,0)$ and observe that $x_4^{1'} = 1$. Also, let $x^{2'} = (0,0,0,1,1,0,1,1)$. It is easily verified that $x^{2'}$ satisfies $x^1 - \xi_p + x^{2'} \in P_{C^1 \cup C^2}^{MK}$ for each $x^1 \in \{x \in P_{C^1}^{MK} : x_4 = 1, \sum_{i \in C^1} x_i = 3\}$, achieving the first condition of Theorem 2. Again, it is easily verified that $x^{1'}$ satisfies $x^{1'} - \xi_p + x^2 \in P^{MK}_{C^1 \cup C^2}$ for each $x^2 \in \{x \in P^{MK}_{C^2}, \sum_{i \in C^2} x_i = 4\}$, obtaining the second condition of Theorem 2. Since both conditions of Theorem 2 are satisfied, the merged inequality (3.5) has a face of dimension at least $\eta + \phi = 7$ in $conv(P^{MK}_{C^1 \cup C^2})$. Thus, the merged inequality is facet defining, as shown by the 8 affinely independent points in Figure 3.3. These points follow directly from Figures 3.1 and 3.2 and the implementation of Theorem 2.

The vertical lines in Figure 3.3 separate the columns into the 3 categories of affinely independent points described in the proof of Theorem 2. The first column of Figure 3 corresponds to the set of points $q^1_1, ..., q^1_i$, where $x_4 = 0$ and $x_j = 0$ for all $j \in C^2$. Columns 2-4 of Figure 3.3 are points where $x^{2'}$ replaces the overlapped ξ_p, corresponding to points of the form $q^1_{i+1} - \xi_p + q^2_1, q^1_{i+2} - \xi_p + q^2_1, ..., q^1_\eta - \xi_p + q^2_1$. For these points, the quantity $x_4 = 1$ is replaced by four terms from $C^2 \setminus \{4\}$.

The final four columns in Figure 3.3 are points where $x^{1'} - \xi_p$ is added to the original columns from Figure 3.2, corresponding to points of the form $q^1_{\eta+1} - \xi_p + q^2_2$, $q^1_{\eta+1} - \xi_p + q^2_3, ..., q^1_{\eta+1} - \xi_p + q^2_{\phi+1}$. These points combine 4 of the 5 terms in C^2 with x_1 and x_3 in C^1. Thus, (3.5) is facet defining by Corollary 1.

$$\left|\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array}\right| = \left|\begin{array}{c|ccc|cccc} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{array}\right|$$

Figure 3.3: Eight Affinely Independent Points that Meet (3.5) at Equality

To prove a lower bound of at least dimension $\eta + \phi = 7$ in $conv(P^{MK}_{C^1 \cup C^2})$, the points in Figure 3.3 must be shown to be affinely independent. The proof of Theorem 2 subtracts

$x^{1'} - \xi_p + x^{2'}$, the second column of Figure 3.3, from each of the last 4 columns in Figure 3.3. The result is in Figure 3.4.

$$\left|\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array}\right| = \left|\begin{array}{c|ccc|cccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & -1 \end{array}\right|$$

Figure 3.4: Demonstration of Affine Independence of the Points in Figure 3.3, Step 1.

Conducting the column operations of adding each of the final four columns in Figure 3.4 to each of the columns 2-4 in Figure 3.4 results in the matrix shown in Figure 3.5.

$$\left|\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{array}\right| = \left|\begin{array}{c|ccc|cccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 4 & 4 & 4 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{array}\right|$$

Figure 3.5: Demonstration of Affine Independence of the Points in Figure 3.3, Step 2.

The columns in Figure 3.5 are clearly linearly independent and therefore trivially affinely independent. Thus

$$x_1 + x_2 + x_3 + \frac{1}{4}x_4 + \frac{1}{4}x_5 + \frac{1}{4}x_6 + \frac{1}{4}x_7 + \frac{1}{4}x_8 \leq 3 \tag{3.6}$$

is facet defining on $conv(P^{MK}_{C^1 \cup C^2}) = conv(P^{MK})$. Thus inequality merging can create facet defining inequalities.

Several other valid inequalities can be easily created, demonstrating the true power of merging. Every valid non-negative inequality of $P_{C^2}^{MK}$ is a donor inequality that can be merged with the C^1 cover constraint on x_4. For instance,

$$7x_4 + 6x_5 + 8x_6 + 6x_7 + 5x_8 \leq 30 \tag{3.7}$$

is valid trivially by (3.2). Observe that this can be strengthened to

$$7x_4 + 6x_5 + 8x_6 + 6x_7 + 5x_8 \leq 27 \tag{3.8}$$

because summing the largest four coefficients is 27. Thus,

$$x_1 + x_2 + x_3 + \frac{7}{27}x_4 + \frac{6}{27}x_5 + \frac{8}{27}x_6 + \frac{6}{27}x_7 + \frac{5}{27}x_8 \leq 3 \tag{3.9}$$

is a valid inequality of P^{MK} by Theorem 1. This is not facet defining, but it does eliminate linear relaxation points (e.g. $(\frac{1}{2},1,1,\frac{6}{7},0,1,0,0)$) that are not eliminated by (3.5). Consequently, once a set of variables is able to be merged, numerous valid merged inequalities can be generated with the inequality merging technique. This process may decrease preprocessing time and be computationally useful. Some additional results on this topic are discussed in the next chapter.

To argue that inequality merging is not a straightforward extension of a known technique, consider Example 1 and lifting techniques. Sequentially lifting over (3.3) or (3.4) generates integer coefficients and would not create (3.5). Using the simultaneous lifting method described in Gutierrez [41] and lifting $\{x_5, x_6, x_7, x_8\}$ into (3.3) yields a zero coefficient with the lifted inequality being $x_1 + x_2 + x_3 + x_4 \leq 3$. Simultaneous lifting $\{x_1, x_2, x_3\}$ into (3.4) also yields a zero coefficient and the resulting inequality is $x_4 + x_5 + x_6 + x_7 + x_8 \leq 4$. Thus any lifting starting with either of the cover inequalities does not generate (3.5).

Since all inequalities can be obtained by iteratively applying various lifting techniques (Balas and Zemel [11], Balas and Ng [12] and Zemel [77] [78]), clearly (3.5) can be achieved from lifting. In this case simultaneously lifting over $x_1 + x_2 + x_3 \leq 3$ would generate (3.5). Since the initial inequality does not eliminate any linear relaxation space from any binary IP, it is unlikely that a person would begin with this inequality given known lifting techniques unless an oracle was consulted. Therefore applied lifting techniques do not typically generate merged inequalities.

Similar to lifting, several techniques such as superadditive cuts (Gomory and Johnson [37] and Wolsey [76],) Chvátal-Gomory cuts (Chvátal [18] and Gomory [35],) and disjunctive cuts (Balas and Perregaard [9]) can generate or dominate all valid inequalities. However, there is not a straightforward or even fairly complex application of any of these techniques to generate (3.5). Thus, these methods are unlikely to yield a merged inequality by practitioners. It is also evident that (3.5) is not the direct result of a modular or a Gomory fractional cut (Gomory [35].) Consequently, to the author's knowledge, merged inequalities are a previously undiscovered class of valid inequalities, which may induce facets. Thus, inequality merging constructs a new category of cutting planes for integer programming problems.

In the specific instance of a single knapsack inequality, merging two distinct covers C^1 and C^2 achieves the same result as simultaneous lifting $C^1 \setminus \{p\}$ into C^2's constraint. However, since this only applies to the special case where C^1 and C^2 are covers of the same constraint, it does not diminish the contribution associated with inequality merging across different constraints.

This chapter introduced the theoretical results for inequality merging that apply to general IP problems with an example that demonstrated the theory on a MK instance. The theory may be extended and applied in different manners for different types of IPs. The next chapter focuses on multiple knapsack problems and cover inequalities. Several theoretical extensions and implementation algorithms are introduced, where inequality merging is used to generate cutting planes from cover inequalities in a MK instance.

Chapter 4

Inequality Merging with Cover Inequalities in the Multiple Knapsack

Chapter 3 presented general theory for inequality merging in integer programming problems. The remainder of the dissertation focuses on the multiple knapsack. Chapter 4 provides theoretical extensions to the general theory that focus on merging over cover inequalities.

This chapter contains a theorem providing conditions under which merging cover inequalities yields a valid merged inequality. Algorithms for the construction process are presented, including discussion of pseudo-costing techniques and using a parameter to select appropriate merging indices. Additional theoretical results present conditions under which merging may occur over multiple donor covers in a MK instance simultaneously. An algorithm is provided that implements this type of simultaneous merging across multiple constraints. This section concludes with an example demonstrating these theoretical results, and it demonstrates that merging cover inequalities creates a new category of cutting planes.

4.1 Merging over Two Cover Inequalities

This section describes how to merge two cover inequalities into a single inequality with a theorem that describes conditions for validity. To facilitate the arguments for validity, this section defines eligible indices using a calculated value ψ. A second theorem proves that using the value of ψ to select candidate indices guarantees that merging covers yields a valid inequality. An algorithm is presented to modify the value of ψ, which may be helpful in some instances.

It is straightforward to find cover inequalities in a MK instance. One cover is called the host cover and the other cover is called the donor cover. Merging the host and donor cover inequalities creates a single merged inequality. Some notation was deliberately adjusted in this chapter to support theoretical contributions with cover inequalities. The general theory in Chapter 3 used C^1 and C^2 to designate sets of indices that had non-zero coefficients in the valid host and donor inequalities. Theoretical results using cover inequalities designate these sets differently, where the host cover C^{host} replaces the set C^1 and the donor cover C^{donor} replaces the set C^2.

Without loss of generality, select a host cover designated as C^{host} from row r in a multiple knapsack instance. Thus the cover inequality $\displaystyle\sum_{i \in C^{host}} x_i \leq |C^{host}| - 1$ is a valid inequality of $conv(P^{MK})$. Another cover from any row s may be a donor cover designated as C^{donor} if $|C^{donor} \cap C^{host}| \leq 1$. Thus, $\displaystyle\sum_{i \in C^{donor}} x_i \leq |C^{donor}| - 1$ is a valid inequality of $conv(P^{MK})$.

Merging the host and donor cover inequalities occurs on binary variable x_p where $p = C^{host} \cap C^{donor}$ or if $C^{host} \cap C^{donor} = \emptyset$, then $p \in C^{host}$. Since x_p is bounded by 1 and $\displaystyle\sum_{i \in C^{donor}} \frac{1}{|C^{donor}| - 1} x_i \leq 1$, it follows that x_p could be replaced in the host cover inequality with the C^{donor} indices with coefficients $\frac{1}{|C^{donor}|-1}$. This follows the theory presented in Chapter 3, and thus merged cover inequalities have the form shown below.

$$\sum_{i \in C^{host} \setminus \{p\}} x_i + \sum_{i \in C^{donor}} \frac{1}{|C^{donor}| - 1} x_i \leq |C^{host}| - 1$$

If the merged inequality is valid, then this inequality includes more nonzero coefficients than either C^{host} or C^{donor}. This inequality has the potential to be a theoretically stronger inequality that merges some of the information from two covers into a single inequality. Since cover inequalities are relatively easy to find, this procedure can rapidly produce cutting planes that contain more nonzero coefficients than standard cover inequalities. The question remains as to whether or not the merged inequality is valid, and conditions are provided in the following theorem.

Theorem 3. *Let C^{host} be a cover from row r and C^{donor} be a cover from some row s in a MK instance such that $|C^{host} \cap C^{donor}| \leq 1$. Define index $p \in C^{host}$ as the merging index with the restriction that if $|C^{host} \cap C^{donor}| = 1$, then $p = C^{host} \cap C^{donor}$. If $C^{host} \setminus \{p\} \cup \{i\}$ is a cover in at least one row of the MK instance for each $i \in C^{donor}$, then the merged cover inequality, $\sum_{i \in C^{host} \setminus \{p\}} x_i + \sum_{i \in C^{donor}} \frac{1}{|C^{donor}| - 1} x_i \leq |C^{host}| - 1$, is valid for conv$(P^{MK}_{C^{host} \cup C^{donor}})$ and conv(P^{MK}).*

Proof. Let x' be any point in P_{MK}. Define $q = \sum_{i \in C^{host} \setminus \{p\}} x'_i$. If $q = |C^{host}| - 1$, then $\sum_{i \in C^{donor}} x'_i = 0$ because $C^{host} \setminus \{p\} \cup \{i\}$ is a cover in some constraint for each $i \in C^{donor}$. Thus, $\sum_{i \in C^{host} \setminus \{p\}} x'_i + \sum_{i \in C^{donor}} \frac{1}{|C^{donor}| - 1} x'_i \leq |C^{host}| - 1$.

If $q \leq |C^{host}| - 2$, then $\sum_{i \in C^{donor}} \frac{1}{|C^{donor}| - 1} x'_i \leq 1$ since C^{donor} is a cover. Thus, $\sum_{i \in C^{host} \setminus \{p\}} x'_i + \sum_{i \in C^{donor}} \frac{1}{|C^{donor}| - 1} x'_i \leq |C^{host}| - 1$ and the result follows.

□

Theorem 3 describes which indices can be used to create a donor cover. These candidate indices can be easily found based upon a ψ threshold, which is associated with the host cover inequality and the merging variable. Given a host cover C^{host} in row r and a designated merging variable $p \in C^{host}$, then $\psi_p = b_r - (\sum_{i \in C^{host}} a_{r,i} - a_{r,p}) + 1$.

The purpose of ψ_p is to identify indices that can be used to create a donor cover from any row s. Define these potential donor indices as $N_{\psi_p} = \{i \in N \setminus (C^{host} \setminus \{p\}) : a_{r,i} \geq \psi_p\}$. If C^{donor} is a cover and $C^{donor} \subseteq N_{\psi_p}$, then merging the host and donor cover on x_p results in a valid merged inequality as the following theorem proves.

Theorem 4. *Assume a multiple knapsack instance, a host cover C^{host} from row r and a merging variable x_p with $p \in C^{host}$. Let C^{donor} be a cover in some row s such that $a_{r,i} \geq \psi_p$ for all $i \in C^{donor}$, then*

$$\sum_{i \in C^{host} \setminus \{p\}} x_i + \frac{1}{|C^{donor}| - 1} \sum_{i \in C^{donor}} x_i \leq |C^{host}| - 1$$

is a valid inequality of $conv(P^{MK})$.

Proof. For contradiction assume that there exists a point $x' \in P_{MK}$ such that x' does not satisfy $\sum_{i \in C^{host} \setminus \{p\}} x'_i + \frac{1}{|C^{donor}| - 1} \sum_{i \in C^{donor}} x'_i \leq |C^{host}| - 1$. The proof divides into two cases, $\sum_{i \in C^{host} \setminus \{p\}} x'_i = |C^{host}| - 1$ and $\sum_{i \in C^{host} \setminus \{p\}} x'_i \leq |C^{host}| - 2$.

Assume $\sum_{i \in C^{host} \setminus \{p\}} x'_i = |C^{host}| - 1$. Since x' violates the merged inequality, there exists an $x'_q = 1$ with $q \in C^{donor}$. Examining row r results in $\sum_{i \in C^{host} \setminus \{p\}} a_{r,i} + a_{r,q} \geq \sum_{i \in C^{host} \setminus \{p\}} a_{r,i} + \psi_p$. By definition $\sum_{i \in C^{host} \setminus \{p\}} a_{r,i} + \psi_p = b_r + 1$. Thus, x' is not a feasible point, a contradiction.

Next, assume $\sum_{i \in C^{host} \setminus \{p\}} x'_i \leq |C^{host}| - 2$. Since C^{donor} is a cover in row s, $\sum_{i \in C^{donor}} x'_i \leq |C^{donor}| - 1$. Thus, x' does not violate the merged inequality.

$\square$

To implement inequality merging, the user must designate a host constraint r and find a host cover in this row. Pseudo-costing is a common method to prioritize variables, discussed in Benichou, et.al. [14], Achterberg, et.al [1], and Gauthier and Ribiere [34]. This pseudo-costing prioritization scheme is then used to choose indices to create C^{host}. Two useful pseudo-costing attributes in MK instances are the reduced cost of the variable and its coefficient in row r.

Once pseudo-costs are associated with indices in C^{host}, any merged variable x_p may result in a ψ_p that is large. In such a case, there may only be a few or possibly zero candidate indices from which to build a donor cover. Without a donor cover, merging is impossible. To alleviate this issue, the following algorithm modifies the host cover to reduce ψ_p so that valid donor covers are more likely to exist.

The input to the Reducing ψ_p Algorithm is a multiple knapsack instance, a valid host cover from row r, and a merging variable x_p with $p \in C^{host}$. In addition, a threshold $\tau \in [0, 1]$ is provided. The output of this algorithm is a new host cover inequality with a new merging variable, which are denoted by $C'^{\ host}$ and $x_{p'}$, respectively.

Reducing ψ_p Algorithm

```
a. Initialization:
    $\psi_p \leftarrow b_r - (\sum_{i \in C^{host}} a_{r,i} - a_{r,p}) + 1$
    $C'^{\ host} \leftarrow C^{host} \setminus \{p\}$
    $a_{total} \leftarrow \sum_{i \in C'^{\ host}} a_{r,i}$
b. Main Step:
    For each $q \in N \setminus C'^{\ host}$
        If $\tau\psi_p \leq a_{r,q} \leq \psi_p - 1$, Then
            $C'^{\ host} \leftarrow C'^{\ host} \cup \{q\}$
            $a_{total} \leftarrow a_{total} + a_{r,q}$
        If $a_{total} > b_r$, Then
            exit $for$ loop
    end $for$
c. Output:
    If: $a_{total} > b_r$, Then
        report $C'^{\ host}$ with $x_q$ as the merging variable $x_{p'}$
     Else return no improvement to $\psi_p$
```

When the Reducing ψ_p Algorithm terminates successfully, observe that $C'^{\ host}$ is a cover since it satisfies the condition that $\sum_{i \in C'^{\ host}} a_{r,i} > b_r$. When this happens, the last index q added to $C'^{\ host}$ becomes the newly determined merged variable $x_{p'}$, and

$$\psi_{p'} = b_r - (\sum_{i \in C'^{\ host}} a_{r,i} - a_{r,p'}) + 1.$$

Since $\psi_{p'} < \psi_p$, smaller $a_{r,i}$ coefficients may identify acceptable additional variables for use in C^{donor}. This increases the likelihood of achieving a valid C^{donor}, thus increasing the opportunity for construction of merged cover cutting planes.

In some instances, the Reducing ψ_p Algorithm terminates successfully with a new cover $C'^{\ host}$ and a new value $\psi_{p'}$, but it may not have a sufficient number indices in $N_{\psi_{p'}}$ to construct C^{donor}. If this happens, the Reducing ψ_p Algorithm should be used iteratively, reducing ψ_p incrementally each time, until a suitable C^{donor} is attained.

Observe that the Reducing ψ_p Algorithm also requires a careful selection of τ to achieve stronger results in many instances. A small value of τ tends to allow smaller a coefficients to enter $C'^{\ host}$. When this happens, the size of $C'^{\ host}$ may become undesirably large or fail to generate a cover. Similarly, if the dimension of the MK instance is relatively small, including too many variables in the host cover may result in fewer (or possibly zero) attainable donor cover inequalities in other rows. However, smaller values of τ may have desirable properties if the problem has many variables because decreasing ψ_p allows many new candidate indices for donor inequalities on other rows.

Higher values of τ may have similar benefits or problems with some MK instances. Very high values of τ may allow few (or zero) new candidate indices in N_{ψ_p}. In such instances, it is more likely that the reducing ψ_p algorithm fails to return a new $C'^{\ host}$ and/or fails to reduce the value of ψ_p. Even if the algorithm succeeds, higher values of τ tend to result in relatively smaller reductions in ψ_p, possibly requiring multiple calls to this procedure when a valid merged inequality is not yet attainable. However, a high-dimensional MK instance has a greater likelihood of achieving the valid merged inequality even when τ is relatively high. In this case, higher values of τ may yield stronger results with host and donor covers of smaller size.

Given this sensitivity to τ, a careful selection of τ should consider the dimension of the MK instance and computational requirements. For practical purposes, it is recommended to consider values of τ between 0.3 and 0.7.

The Reducing ψ_p Algorithm is a linear algorithm for each specified τ value. The initialization requires $O(|C^{host}|)$. The mainstep could search through all other indices, so it performs in $O(N \setminus |C^{host}|)$ effort. Thus, the algorithm runs in $O(n)$, which is linear for a fixed τ.

4.2 Merging over Multiple Donor Cover Inequalities Simultaneously

This section presents a method to strengthen the previous results by merging on multiple donor covers at the same time. Conditions are provided to create valid inequalities by merging over three or more cover inequalities simultaneously. Another algorithm is presented to search for the strongest merging coefficients among multiple potential donor rows in the MK instance.

Simultaneous merging over multiple donor covers begins with a C^{host} cover from a MK constraint with $p \in C^{host}$ and its associated ψ_p and set N_{ψ_p}. The inequality

$$\sum_{i \in C^{host} \setminus \{p\}} x_i + \sum_{j \in N_{\psi_p}} \alpha_j x_j \leq |C^{host}| - 1$$

is likely to be valid for any $\alpha_j \leq \frac{1}{|C_j|-1}$ where $C_j \subseteq N_{\psi_p}$ is any cover from any constraint of the MK instance with $j \in C_j$. Thus, the strongest such inequality would select $\alpha_j = \frac{1}{|C_j|-1}$

where $C_j \subseteq N_{\psi_p}$ and $j \in C_j$ is the maximum cardinality cover from any row.

The check of validity must assure that there does not exist a feasible point which violates this new inequality. Prior to this result, define c^q_{min}, α^q_{min}, and a^q_{min} as shown below.

$$c^q_{min} = min_{Q \subset C^{host} \setminus \{p\},\ |Q|=q} \{\sum_{i \in Q} a_{r,i}\}$$
$$\alpha^q_{min} = min_{D \subseteq N_{\psi_p}} \{|D| : \sum_{i \in D} \alpha_i > |C^{host}| - q - 1\}$$
$$a^q_{min} = min_{D' \subseteq N_{\psi_p} : |D'| = \alpha^q_{min}} \{\sum_{i \in D'} a_{r,i}\}$$

These values support the second condition described in Theorem 5.

Theorem 5. *Let C^{host} be a cover from an MK constraint with $p \in C^{host}$, corresponding value ψ_p and associated set N_{ψ_p}. Then the inequality*

$$\sum_{i \in C^{host} \setminus \{p\}} x_i + \sum_{j \in N_{\psi_p}} \alpha_j x_j \leq |C^{host}| - 1$$

is valid for $conv(P^{MK})$ for any $\alpha_j \leq \frac{1}{|C_j|-1}$ where $j \in C_j \subseteq N_{\psi_p}$ is any cover from any constraint of the MK instance as long as one of the following conditions holds

i) $\sum_{i \in N_{\psi_p}} \alpha_i \leq 1$

ii) $c^q_{min} + a^q_{min} > b_r$ *for all integer* $q \in \{1, 2, ..., |C^{host}| - 1\}$.

Proof. Let $x' \in P_{MK}$. Since $i \in N_{\psi_p}$, then $\{i\} \cup C^{host} \setminus \{p\}$ is a cover in row r. If $\sum_{i \in C^{host} \setminus \{p\}} x'_i = |C^{host}| - 1$, then $\sum_{i \in N_{\psi_p}} x'_i = 0$. Thus, $\sum_{i \in C^{host} \setminus \{p\}} x'_i + \sum_{i \in N_{\psi_p}} \alpha_i x'_i \leq |C^{host}| - 1$ for every value of α_i.

Assume i) is true and $\sum_{i \in C^{host} \setminus \{p\}} x'_i \leq |C^{host}| - 2$. Then $\sum_{i \in N_{\psi_p}} \alpha_i x_i \leq 1$ because i) is true. Consequently, $\sum_{i \in C^{host} \setminus \{p\}} x'_i + \sum_{j \in N_{\psi_p}} \alpha_j x'_j \leq |C^{host}| - 1$.

Assume ii) is true and let $q = \sum_{i \in C^{host} \setminus \{p\}} x'_i$. By ii) $\sum_{j \in N_{\psi_p}} \alpha_j x'_j \leq q$ or x' violates row r. Thus, $\sum_{i \in C^{host} \setminus \{p\}} x'_i + \sum_{j \in N_{\psi_p}} \alpha_j x'_j \leq |C^{host}| - 1$.

$\square$

An immediate result of Theorem 5 is an algorithm to merge over multiple donor covers simultaneously. This algorithm explores all rows to determine the smallest eligible covers of each merging variable in N_{ψ_p}. This translates into the strongest α coefficient for each merging variable. The input to the Donor Coefficient Strengthening Algorithm (DCSA) is an MK instance, a host cover C^{host} from row r, and an index $p \in C^{host}$.

Donor Coefficient Strengthening Algorithm

```
a. Initialization:

        $\psi_p \leftarrow b_r - (\sum_{i \in C^{host}} a_{r,i} - a_{r,p}) + 1$
        $N_{\psi_p} \leftarrow \{i \in N \setminus (C^{host} \setminus \{p\}) : a_{r,i} \geq \psi_p\}$
        Let $A'_{r'}$ be $N_{\psi_p}$ sorted according to the $a$ coefficients in row $r' \in \{1, ..., m\}$

b. Main Step

        For each $i \in N_{\psi_p}$
            Set $\alpha_i \leftarrow 0$
            For each $r' \in \{1, ..., m\}$
```

```
                $coversum \leftarrow a_{i,r'}$

                $coversize \leftarrow 1$

                $j \leftarrow 1$

                While $coversum \leq b_{r'}$ and $j \leq |N_{\psi_p}|$

                    If $j \neq i$, Then

                        $coversum \leftarrow coversum + A'_{r',j}$

                        $coversize \leftarrow coversize + 1$

                    $j \leftarrow j + 1$

                end $while$

                if $coversum > b_{r'}$ and $\alpha_i < \frac{1}{coversize-1}$, Then

                    $\alpha_i \leftarrow \frac{1}{coversize-1}$

            end $for$

        end $for$
```

c. Output

Return α, an array of the largest merging coefficients for the indices in N_{ψ_p}.

The DCSA identifies the smallest donor covers possible for each index in N_{ψ_p} from each row in the MK instance using the indices sorted in each row by the a values. Observe that the DCSA does not guarantee a valid inequality, but it does identify the strongest possible merged inequality. If the reported merged inequality satisfies a condition of Theorem 5, then it is a valid inequality.

The computational effort for DCSA's initialization is $O(|C^{host}|+m|N_{\psi_p}|log(|N_{\psi_p}|))$. The mainstep requires $O(m|N_{\psi_p}|^2)$. Thus DCSA's total computational effort is $O(|C^{host}| + m|N_{\psi_p}|^2)$. Although this is a cubic run time, the DCSA performs quickly in practice.

The next section demonstrates the theoretical merging techniques presented in this section through an example MK instance. The example also implements the algorithms shown in this section.

4.3 Example Cover Inequality Merging in a Multiple Knapsack

The following example demonstrates the theoretical concepts discussed earlier, including the validity of merging two cover inequalities from Theorem 3 and the calculation and usage of the ψ_p term to find a valid merged inequality using Theorem 4. It also shows an improvement to the host cover via the Reducing ψ_p Algorithm. The example further strengthens the merged cover inequality using the concepts in Theorem 5 and the implementation of DCSA. A discussion demonstrates that merged cover inequalities are a new category of cutting planes for MK instances.

Consider multiple knapsack constraints of the form $Ax \leq b$ with $n = 14$ and $m = 2$ where

$$A = \begin{bmatrix} 20 & 18 & 16 & 16 & 15 & 12 & 11 & 10 & 10 & 8 & 6 & 5 & 5 & 3 \\ 14 & 19 & 13 & 6 & 6 & 20 & 5 & 12 & 11 & 20 & 14 & 14 & 6 & 12 \end{bmatrix}$$

and

$$b = \begin{bmatrix} 79 \\ 75 \end{bmatrix}.$$

Designate the first constraint as the host constraint, $r = 1$. This example uses a pseudo-costing strategy that chooses terms for C^{host} according to the size of the coefficients in the first constraint. The host constraint is sorted by this metric, while the second constraint is not. Based on the sorted coefficients, the most desirable cover is $\{1, 2, 3, 4, 5\}$. Since index 5 is least desirable, it is the merging index, $p = 5$. Since $a_{1,1} + a_{1,2} + a_{1,3} + a_{1,4} = 70$ and the corresponding right-hand side is 79, $\psi_5 = (79 - 70) + 1 = 10$. Because $a_{1,5}, a_{1,6}, a_{1,7}, a_{1,8}$ and $a_{1,9}$ are all greater than or equal to 10, the candidate indices for the donor cover are restricted to $N_{\psi_5} = \{5, 6, 7, 8, 9\}$.

Observe that the candidate indices for the donor inequality do not attain a cover in either the first or second constraint. The Reducing ψ_p Algorithm is used to change the host cover to create a smaller ψ. This smaller value of ψ should facilitate additional candidate indices for the donor cover.

Let $\tau = \frac{1}{2}$, then the Reducing ψ_p Algorithm seeks a host cover with a ψ value that is less than or equal to 5. In this case $\{5\}$ is eliminated from the host cover, and the host cover adds an index with a coefficient between 5 and 9. Indices 10, 11, 12, and 13 are all suitable and index 11 is added to C'^{host}. However, $C'^{host} = \{1, 2, 3, 4, 11\}$ is not a cover. Including either index 12 or 13 would create a host cover and by desirability $C'^{host} = \{1, 2, 3, 4, 11, 12\}$.

The new value for ψ_{12} is reduced exactly by the coefficient of the first added index, $a_{1,11}$. Thus $\psi_{12} = 4$, and the candidate indices for the donor cover are $N'_{\psi_{12}} = \{5, 6, 7, 8, 9, 10, 12, 13\}$. There exist several covers in constraint two from this candidate set. One such cover is $C^{donor} = \{6, 8, 9, 10, 12\}$. Since a donor cover now exists, C'^{host} becomes C^{host}.

The algorithm has now determined the valid host cover inequality (4.1) and the valid donor cover inequality (4.2). Merging the host with the donor on x_{12} yields the valid merged inequality (4.3) according to Theorem 4.

$$x_1 + x_2 + x_3 + x_4 + x_{11} + x_{12} \le 5 \tag{4.1}$$

$$x_6 + x_8 + x_9 + x_{10} + x_{12} \le 4 \tag{4.2}$$

$$x_1 + x_2 + x_3 + x_4 + \frac{1}{4}x_6 + \frac{1}{4}x_8 + \frac{1}{4}x_9 + \frac{1}{4}x_{10} + x_{11} + \frac{1}{4}x_{12} \le 5. \tag{4.3}$$

The following arguments demonstrate Theorems 3 and 4 in practice. Verifying the validity of (4.3) requires that coefficients in row one associated with $C^{host} \setminus \{12\} \cup \{j\}$ are a cover for some constraint for every $j \in C^{donor}$. The sum of the $a_{1,j}$ coefficients over $C^{host} \setminus \{12\}$ is 76. Clearly $C^{host} \setminus \{12\} \cup \{j\}$ is a cover in the first knapsack as long as $a_{1,j} \ge 4$ and $j \notin C^{host} \setminus \{12\}$. Since all candidate donor indices have $a_{1,j} \ge 4 = \psi_{12}$, (4.3) is verified as a valid inequality of $conv(P^{MK})$.

Observe that numerous other minimal donor covers exist when $p = 12$. Two other examples are $\{5, 6, 7, 9, 10, 12\}$ with corresponding inequality $x_5+x_6+x_7+x_9+x_{10}+x_{12} \leq 5$ and $\{5, 6, 7, 8, 9, 10, 13\}$ with corresponding inequality $x_5+x_6+x_7+x_8+x_9+x_{10}+x_{13} \leq 6$. Accordingly, each of these donor cover inequalities could be merged with the host cover inequality yielding the following valid merged inequalities

$$x_1 + x_2 + x_3 + x_4 + \frac{1}{5}x_5 + \frac{1}{5}x_6 + \frac{1}{5}x_7 + \frac{1}{5}x_9 + \frac{1}{5}x_{10} + x_{11} + \frac{1}{5}x_{12} \leq 5 \qquad (4.4)$$

$$x_1 + x_2 + x_3 + x_4 + \frac{1}{6}x_5 + \frac{1}{6}x_6 + \frac{1}{6}x_7 + \frac{1}{6}x_8 + \frac{1}{6}x_9 + \frac{1}{6}x_{10} + x_{11} + \frac{1}{6}x_{13} \leq 5. \qquad (4.5)$$

Each of these merged inequalities remove linear relaxation points and are thus cutting planes. For instance, the point (1,1,1,1,0,0,0,0,0,$\frac{1}{4}$,1,0,0,0) is eliminated by each of these merged inequalities. Additionally, it is simple to find points in P^{LR} that are satisfied by two of the three merged inequalities, but eliminated by the other inequality. Thus, each merged inequality is eliminating distinct regions of the linear relaxation space.

Returning to the original host cover $\{1, 2, 3, 4, 11, 12\}$, it is also possible to generate new families of merged inequalities if merging on $p = 11$ instead of $p = 12$. By changing the index selected for merging, $\psi_{11} = 5 = (79 - 75) + 1$ with corresponding candidate donor indices $\{5, 6, 7, 8, 9, 10, 11, 13\}$. Similar to the examples shown previously, many possible new donor covers now exist. For instance, $C^{donor} = \{6, 8, 9, 10, 11\}$ yields

$$x_1 + x_2 + x_3 + x_4 + \frac{1}{4}x_6 + \frac{1}{4}x_8 + \frac{1}{4}x_9 + \frac{1}{4}x_{10} + \frac{1}{4}x_{11} + x_{12} \leq 4. \qquad (4.6)$$

The idea of ψ guarantees validity, but it is not necessary to merge covers. Consider $C^{host} = \{1, 2, 3, 4, 6\}$ with $p = 4$. In the first constraint, $\{1, 2, 3, 5, 6\}$ is a cover and so $\{5\}$ is a candidate index. The second constraint has several relevant covers: $\{1, 2, 3, 6, 9\}$,

$\{1,2,3,6,10\}$, $\{1,2,3,6,11\}$, $\{1,2,3,6,12\}$ and $\{1,2,3,6,14\}$. Thus, the candidate indices are now $\{4,5,9,10,11,12,14\}$ by Theorem 4. Another cover in the second constraint is $\{4,9,10,11,12,14\}$, which results in the following merged constraint.

$$x_1 + x_2 + x_3 + \frac{1}{5}x_4 + x_6 + \frac{1}{5}x_9 + \frac{1}{5}x_{10} + \frac{1}{5}x_{11} + \frac{1}{5}x_{12} + \frac{1}{5}x_{14} \leq 4 \tag{4.7}$$

Such constraints may be more useful computationally since they are incorporating covers from multiple constraints to obtain validity. For instance, the linear relaxation point (1,1,1,$\frac{1}{3}$,0,1,0,0,$\frac{1}{4}$,0,0,0,0,$\frac{1}{3}$) is eliminated by this inequality.

To demonstrate Theorem 3, an additional row is added to this example. Now consider the following multiple knapsack instance

$$A = \begin{bmatrix} 20 & 18 & 16 & 16 & 15 & 12 & 11 & 10 & 10 & 8 & 6 & 5 & 5 & 3 \\ 14 & 19 & 13 & 6 & 6 & 20 & 5 & 12 & 11 & 20 & 14 & 14 & 6 & 12 \\ 4 & 6 & 7 & 6 & 18 & 3 & 17 & 15 & 19 & 4 & 16 & 8 & 9 & 14 \end{bmatrix}$$

and

$$b = \begin{bmatrix} 79 \\ 75 \\ 73 \end{bmatrix}.$$

Again, consider $C^{host} = \{1,2,3,4,11,12\}$, with $\psi_{12} = 4$ and the set of eligible donor indices $N_{\psi_{12}} = \{5,6,7,8,9,10,12,13\}$. For each index in $N_{\psi_{12}}$, the DCSA forces this index as the first element in a cover and then adds other indices according to the sorted order for each row. Observe that $\{5,6,7,8,9,10,12,13\}$ is not a cover in row 1, so only rows 2 and 3 are considered.

For index 5, the smallest covers are $\{5, 6, 10, 12, 8, 9\}$ and $\{5, 9, 7, 8, 13\}$ in rows 2 and 3, respectively. Continuing this logic for each of the other indices results in Table 4.1. The smallest covers are listed in the order in which the DCSA adds indices to the cover.

Index	Smallest Cover	Row	α
5	$\{5, 9, 7, 8, 13\}$	3	$\frac{1}{4}$
6	$\{6, 10, 12, 8, 9\}$	2	$\frac{1}{4}$
7	$\{7, 9, 5, 8, 13\}$	3	$\frac{1}{4}$
8	$\{8, 9, 5, 7, 13\}$	3	$\frac{1}{4}$
9	$\{9, 5, 7, 8, 13\}$	3	$\frac{1}{4}$
10	$\{10, 6, 12, 8, 9\}$	2	$\frac{1}{4}$
12	$\{12, 6, 10, 8, 9\}$	2	$\frac{1}{4}$
13	$\{13, 9, 5, 7, 8\}$	3	$\frac{1}{4}$

Table 4.1: Applying DCSA to Find the Strongest Coefficients

Thus the simultaneous merged inequality is

$$x_1 + x_2 + x_3 + x_4 + \frac{1}{4}x_5 + \frac{1}{4}x_6 + \frac{1}{4}x_7 + \frac{1}{4}x_8 + \frac{1}{4}x_9 + \frac{1}{4}x_{10} + x_{11} + \frac{1}{4}x_{12} + \frac{1}{4}x_{13} \leq 5. \quad (4.8)$$

Observe that this new inequality dominates all of the previous inequalities. Furthermore, both donor rows are necessary to achieve this inequality. For instance, the smallest cover in row 3 containing index 6 has 6 indices. Thus row 2 is necessary to find a cover containing index 6 with only 5 indices, obtaining the $\frac{1}{4}$ coefficient on x_6 in (4.8). Similarly, the smallest cover in row 2 containing index 7 has 6 indices. Thus row 3 is necessary to obtain a 5-element cover containing x_7 and the corresponding coefficient $\frac{1}{4}$ for x_7 in (4.8).

To argue the validity of (4.8), consider Theorem 5. Since $\sum_{j \in N_{\psi_{12}}} \alpha_j = 2$, condition *i*) is not satisfied. For condition *ii*), observe that $c^1_{min} = a_{1,11} = 6$, $c^2_{min} = a_{1,11} + a_{1,4} = 22$, and similarly the other values are $c^3_{min} = 38$, $c^4_{min} = 56$, and $c^5_{min} = 76$. Determining the values for α_{min} yield that α^1_{min}, α^2_{min}, and α^3_{min} do not exist as $\sum\limits_{j \in N_{\psi_{12}}} \alpha_j = 2$. However, $\alpha^4_{min} = 5$ because it requires five variables with coefficients in $N_{\psi_{12}}$ to be set to one to arrive at a value strictly larger than $1 = |C^{host}| - 4 - 1$. Since $|C^{host}| - 5 - 1 = 0$, $\alpha^5_{min} = 1$.

Since α^1_{min}, α^2_{min}, and α^3_{min} do not exist, only a^4_{min} and a^5_{min} are determined. The value of $a^4_{min} = a_{1,13} + a_{1,12} + a_{1,10} + a_{1,9} + a_{1,8} = 38$. Similarly, $a^5_{min} = a_{1,13} = 5$. Condition *ii*) of Theorem 5 checks $c^4_{min} + a^4_{min} = 56 + 38 = 94 > 79$, and $c^5_{min} + a^5_{min} = 76 + 5 = 81 > 79$. Thus, (4.8) meets condition *ii*) of Theorem 5 and it is valid. As a note, observe that checking $q = |C^{host}| - 1$ is always satisfied by the definition of N_{ψ_p}.

The idea of Theorem 5 is to use α_{min}, c_{min}, and a_{min} to guarantee validity of the simultaneously merged inequality. If the merged inequality is invalid, then there is a collection of the variables set to 1 in the host constraint that violates the merged inequality. Using α_{min}, c_{min} and a_{min} tests a hypothetical collection of variables that would be most likely to induce invalidity.

To clarify the purpose of α_{min}, c_{min}, and a_{min}, consider the following discussion. In this example, invalidity could occur if $x_1 + x_2 + x_3 + x_4 + x_{11} = 4$. The minimum contribution of $\sum a_{1,1} + a_{1,2} + a_{1,3} + a_{1,4} + a_{1,11}$ such that four of them are taken is $6+16+16+18 = 56 = c^4_{min}$. With $x_1 + x_2 + x_3 + x_4 + x_{11} = 4$, invalidity could occur if $\frac{1}{4}x_5 + \frac{1}{4}x_6 + \frac{1}{4}x_7 + \frac{1}{4}x_8 + \frac{1}{4}x_9 + \frac{1}{4}x_{10} + \frac{1}{4}x_{12} + \frac{1}{4}x_{13} > 1$. This requires at least 5 merged variables to be 1 ($5 \times \frac{1}{4} > 1$). Thus, $\alpha^4_{min} = 5$. Finding the 5 smallest a coefficients in the host constraint corresponding to any simultaneously merged variable derives $a^4_{min} = 5 + 5 + 8 + 10 + 10 = 38$. Since $38 + 56 > 79 = b_1$, the most likely hypothetical point to violate the inequality is not feasible in the host row. Thus, the inequality is valid when $x_1 + x_2 + x_3 + x_4 + x_{11} = 4$.

This fast check in Theorem 5 is a sufficient condition, but it is not a necessary condition. It is possible to not have the smallest coefficients in the merged inequality linked to the smallest a coefficients. Thus, this check guarantees validity, but there may exist valid simultaneously merged inequalities that violate the check.

The final benefit of this example demonstrates that merging cover inequalities are not an immediate extension of known methods. Clearly, the general inequality merging in Chapter 3 did not merge multiple donor covers and could not obtain (4.8). As argued in Chapter 3, general inequality merging or cover inequality merging is also fundamentally different from other popular cutting plane generation techniques such as C-G cuts (Chvátal [18] and Gomory [35]), disjunctive cuts (Balas and Perregaard [9]), Gomory cuts (Gomory [35]), or superadditive cuts (Gomory and Johnson [37] and Wolsey [76]). Theoretically, these methods could generate (4.8), but they would require numerous iterative applications to find this cutting plane. Such a result is unlikely to occur without the consultation of an oracle to select initial inequalities, weights or other necessary input.

There are similarities between merging cover inequalities and some categories of lifting. Any type of sequential lifting has integer coefficients [75], and sequence independent lifting would require all non-cover coefficients in this example to be 0 [40]. Thus neither of these methods generate (4.8). While simultaneous lifting could theoretically generate (4.8) [41], it would require starting with the trivial cutting plane $x_1 + x_2 + x_3 + x_4 + x_{11} \leq 5$ and furthermore having a perfect guess of proper weights. Consequently, inequality merging yields inequalities similar to (4.8) that are extremely unlikely to be produced by lifting techniques.

A single call to the DCSA creates (4.8) and requires $O(nm^2)$ effort. Thus, merging over cover inequalities is a new method to obtain previously unknown inequalities. Given the large size of most multiple knapsack problems, the flexibility of the construction algorithms are usually capable of finding strong candidate C^{host} and C^{donor} inequalities.

Both Chapters 3 and 4 have provided theoretical properties of merged inequalities. The question remains as to whether or not these inequalities can be computationally useful. Chapter 5 provides the results of a thorough computational study, demonstrating the practical effectiveness of inequality merging on benchmark multiple knapsack problems.

Chapter 5

Computational Study

Chapter 5 includes the results of a thorough computational study that validate the effectiveness of inequality merging for multiple knapsack problems. The computational study compares solution times for multiple knapsack problems both with and without the use of merged inequalities. This is accomplished using CPLEX 12.5 [21] to determine the computational requirements with default CPLEX settings and then comparing them to the computational requirements for the same problems with the addition of merged cover inequalities. The results are obtained on a PC with an i7-4770 processor at 3.4 Ghz with 8 GB of RAM. Since larger problems become memory intensive, CPLEX writes out to memory node files when necessary.

The study considers a variety of implementation strategies including different pseudo-costing techniques, the number of merged inequalities added, the possibility of overlapping rows when multiple cuts are added, and the option to use the Donor Coefficient Strengthening Algorithm when constructing merged inequalities. Significant experimentation focused on smaller problems, and this process yielded the recommended implementation strategies for general (larger) problems. Following the recommended construction and implementation strategies, merged inequalities produced average reductions of about 9% of computational effort required for MK problems as compared to baseline CPLEX settings.

5.1 Multiple Knapsack Instances

There are numerous types of multiple knapsack problems. A standard benchmark set of MK instances from the $OR-Library$ [62] are used for this study. The instances chosen for this study were developed by Chu and Beasley in 1998 [19]. The test problems include nine classes containing 5, 10, and 30 rows with 100, 250, and 500 variables. These problems are saved in files titled mknapcb1,...,mknapcb9. Each class contains 30 MK instances. These 30 instances are divided into groups of 10 based upon a tightness ratio. These 270 benchmark instances are randomly created using the following method.

Let $a_{i,j}$ be an integer uniformly distributed between 0 and 1,000. With these random coefficients, $b_i = \lfloor s \sum_{j=1}^{n} a_{i,j} \rfloor$ for each $j = 1, ..., m$. The value of s is called the tightness ratio, which is .25 for the first 10 instances, .5 for the second ten instances, and .75 for the final ten instances. The cost values are determined by $c_j = \lfloor \sum_{i=1}^{m} \frac{a_{i,j}}{m} + u \rfloor$, where u is a uniform number between 0 and 500.

For this computational study, the first ten instances are considered because the tightness ratio has the lowest value of 0.25. When the tightness ratio is 0.5 or higher, then C^{host} tends to include more variables. Since the variables in C^{host} are prohibited from being in N_{ψ_p}, a higher tightness ratio reduces the size of N_{ψ_p} which decreases the likelihood of finding a donor cover in any row.

The problems developed by Chu and Beasley span a wide gamut of size and difficulty. The three files of problems with only 100 columns were very easy for modern computers to solve quickly. In particular, many of the smallest problems could be solved in less than a second, and the vast majority of these problems solved so quickly that there was very little difference between the baseline CPLEX and the inclusion of merged inequalities. Given the trivial size of these problems, the computational study did not consider problems with only 100 variables.

Conversely, the largest problems (10×500), (30×250), and (30×500) were so big that they typically took over a day to solve each of the sub-problems in those files. Consequently, it took at least 2-3 weeks to solve the 10 sub-problems in order to determine a complete set of run output data. Thus computational results were not pursued for those instances due to the prohibitively long computational requirements.

Instead, the computational study focuses its attention on medium sized problems in files mknapcb2 (5×250) and mknapcb5 (10×250). Of these two files, the smaller problems (5×250) were used for extensive experimentation since the first 10 instances could be solved in approximately 5-10 minutes. These problems were complex enough to demonstrate the effect of adding merged inequalities, and they solved quickly enough to allow substantial experimental variations with numerous subordinate iterations. Thus, the experimentation on smaller problems motivated several insights concerning construction techniques and implementation strategies that tended to yield better results.

The second file (mknapcb5, with problems of size 10×250) served as a test-ground for larger problems. It typically took a total of 1-2 days to solve the first 10 instances in this file. By using the recommended strategies learned from the smaller problems, fewer iterations of the larger problems still allowed for similar average improvements in computational requirements. These larger problems were excellent representatives of difficult, real-world problems, and the observed reductions in computational requirements validated the theoretical advancements in this dissertation as effective methods to help solve modern MK problems more rapidly.

5.2 Testing Implementation Strategies

This section describes the implementation techniques used to generate merged cover inequalities. Experimentation with the smaller MK problems focused on implementation strategies that varied three critical components. First, what pseudo-costing strategies generate the most effective merged cover inequalities? Second, should there be overlapping of rows between host and donor inequalities if multiple cuts are added? Third, how many merged inequalities should be added?

5.2.1 Pseudo-Costing Strategies

A primary implementation strategy focused on the choice of pseudo-cost weighting techniques used to designate the indices in the host inequality. The two attributes considered in this study are the reduced costs of each variable and the a coefficients of each variable in the row corresponding to the host inequality. Three different options for pseudo-costing were evaluated.

One option primarily sorts on the reduced costs of the variables. This option tends to select host cover inequality indices that are more likely to appear in the optimal linear relaxation solution. Consequently, tested instances of the merged inequalities in this option were typically near the right-hand side at the root node, and on some occasions they eliminated the root node's linear relaxation solution.

A second option sorts on the a coefficient values of the indices in the host constraint. This policy usually builds stronger host cover inequalities because fewer indices were necessary to construct the host covers, and the corresponding cover inequalities had smaller right-hand sides than the first option. Larger host cover coefficients also tend to make it easier to adjust ψ_p if necessary to find valid indices for the donor inequality.

The third option considered in this study offered equal weight to both attributes when constructing host cover inequalities. Thus the indices are sorted on a combination of a coefficients and reduced costs. This technique attempts to gain some of the benefits of both attributes without focusing exclusively on only one of the two.

5.2.2 Overlapping Rows

The second idea considered as an implementation strategy was whether or not overlapping rows improves the performance of the merged inequalities when multiple merged cuts are added. Consider an instance where two cuts are added. Suppose that the first cut merged cover inequalities from rows 1 and 2 with row 1 contributing the host cover inequality and row 2 serving as one of the rows that contributes variables from the corresponding donor cover inequality. If the second cut used row 2 to contribute the second host inequality and any other row(s) contributing donor cover variables, this would be considered an overlap. Selection of a previously unused row as the second host inequality is not an overlap.

The concept of forcing an overlap tests the hypothesis that overlapping rows allows the algorithm to glean more information from the overlapped row. This requires the same row to contribute both the host cover information according to the pseudo-costs of each coefficient and the donor cover information for a second merged inequality according to the value of ψ_p in the second host inequality.

Use of the Donor Coefficient Strengthening Algorithm (DCSA) from Chapter 4 would significantly increase the likelihood of an overlap, since all rows are considered in the process of strengthening donor coefficients. In practice, employing the DCSA almost always defined an overlap regardless of the choice of subsequent host inequalities when multiple merged cutting planes were generated. Thus, use of the DCSA was part of this experimental process. Some experimentation used the DCSA and some did not. However, employing the DCSA constituted an overlap for experimental purposes.

5.2.3 Varying the Number of Merged Inequalities

The final implementation strategy studied on the smaller MK instances considered the number of merged cover inequalities generated. Adding a single merged inequality is equivalent to the addition of a single cutting plane to the existing problem. It is typically possible to find multiple merged inequalities for each MK instance, and adding more cutting planes has theoretical appeal since more linear relaxation space may be removed.

However, there is also theoretical appeal associated with adding relatively fewer merged inequalities since the addition of each constraint increases the size of the A matrix and basis. This may escalate the computational time required, slowing the computational process for each iteration of branch-and-bound.

If each row of the original constraint matrix is considered as the host inequality at most once, then the size of the A matrix is at most doubled with new merged cutting planes. The experimental question considered how many merged inequalities was best, balancing the benefit of additional cuts with the detrimental effect of increasing the size of the A matrix too much. Thus, the computational study evaluated the inclusion of several numbers of merged cutting planes ranging from 1 to m.

To evaluate the implementations, each test problem was solved with and without merged cover inequalities. The baseline solution for each problem uses the CPLEX 12.5 solver at its default settings. Code was developed that added merged inequalities as a preprocessing step to the given problems, and then CPLEX 12.5 solved the problems a second time. The following sections discuss the computational results and provide insights and recommendations.

5.3 Computational Results

The computational study considered the variations of each implementation strategy by testing both small and large instances. Instead of reporting the time in seconds, the data below compares computational ticks in CPLEX. Fischetti, et. al. argue the benefit of this technique in [33], and Ju, et. al. use a similar process in [52]. Ticks provide a more accurate comparison between the experimental runs because the computational time in seconds is subject to variability on different computers with multiple computers running different implementation strategies and/or pseudo-costing construction techniques. However, computational ticks alleviate this concern of variability between machines by reporting a deterministic measure of computational effort.

Merged inequalities are constructed and added as preprocessing cuts. This construction process is extremely rapid, and the preprocessing times were less than 0.003 seconds for every multiple knapsack instance in this study. The total preprocessing time for all 10 instances was less than 0.02 seconds. Thus, the computational effort associated with preprocessing is negligible and does not diminish the computational savings from inequality merging.

A summary of the primary findings are discussed in the paragraphs below with tabular summaries of some selected output. Since the two categories of MK test problems in this computational study included 10 multiple knapsack subordinate instances, most of the tables compare the aggregate total ticks required to solve all 10 problems using the baseline CPLEX 12.5 and the inequality merging technique.

5.3.1 Computation Results for Small Problems

Problems from the smaller MK instances with 5×250 A matrices (file mknapcb2) offered an excellent opportunity for experimenting with each of the implementation strategies. Since these problems solved relatively quickly, they were used to test a variety of implementation strategies. Observe that there are combinatorially many row combinations that are possible for each implementation strategy. An extensive experimentation conducted a thorough, but not exhaustive, process of varying row combinations for each strategy.

Tables 5.1 and 5.2 include selected output from these experiments on the smaller MK problems, showing the best known computational results that correspond to each strategy. Since there are 5 rows in the smaller test problems, each implementation strategy was tested with the inclusion of 1-5 merged inequalities. Table 5.1 shows the results for iterations with 1, 2, or 3 merged inequalities added. Table 5.2 shows the results with 4 or 5 merged inequalities added. A later conclusion recommends constructing 1-3 merged inequalities, so the data is separated in this manner as a convenience for display purposes.

The pseudo-costing categories are split into three sub-columns in Tables 5.1 and 5.2. These represent a focus on reduced costs, a focus on a values, and a balanced approach between the two. The numbers in those columns represent how many cuts followed each of the possible pseudo-costing strategies in that particular experiment.

# Merged	Overlap	Pseudo-Costing Strategy			Total Ticks	Percent
Cuts	Rows	Red. Costs	Balanced	a Values	(10 probs.)	Improv.
Baseline	Baseline	0	0	0	81497	Baseline
1	N/A	1	0	0	**70895**	13.0%
1	N/A	0	0	1	**69669**	14.5%
1	N/A	0	1	0	75868	6.9%
2	Yes	2	0	0	**72376**	11.2%
2	Yes	0	0	2	78840	3.3%
2	Yes	0	2	0	**71668**	12.1%
2	Yes	1	0	1	**71305**	12.5%
2	Yes	0	1	1	**67634**	17.0%
2	Yes	1	1	0	76272	6.4%
2	No	2	0	0	81022	0.6%
2	No	0	0	2	76956	5.6%
2	No	0	2	0	77947	4.4%
2	No	1	0	1	**64417**	21.0%
2	No	0	1	1	**72356**	11.2%
2	No	1	1	0	79088	3.0%
3	Yes	3	0	0	74985	8.0%
3	Yes	0	0	3	**72123**	11.5%
3	Yes	0	3	0	**67794**	16.8%
3	Yes	1	1	1	**72593**	10.9%
3	No	3	0	0	80178	1.6%
3	No	0	0	3	75445	7.4%
3	No	0	3	0	77490	4.9%
3	No	1	1	1	77448	5.0%
Merged Average					74099	9.1%

Table 5.1: Changing Implementation Strategies for Smaller MK Problems, 1-3 Cuts

Observe that inequality merging outperformed the baseline CPLEX computational ticks for all strategies in Table 5.1 with 1, 2, or 3 added inequalities, and inequality merging also outperformed the baseline CPLEX for most of the 4 and 5 cut strategies in Table 5.2. Additionally, several of the better results in both tables achieved improvements in excess of 10% over the baseline computational ticks required to solve the first 10 MK instances in the smaller problems.

Extensive experimentation on the smaller MK instances (5×250) informed the best

# Merged	Overlap	Pseudo-Costing Strategy			Total Ticks	Percent
Cuts	Rows	Red. Costs	Balanced	a Values	(10 probs.)	Improv.
Baseline	Baseline	0	0	0	81497	Baseline
4	Yes	4	0	0	80230	1.6%
4	Yes	0	0	4	79756	2.1%
4	Yes	0	4	0	80494	1.2%
4	Yes	1	2	1	73751	9.5%
4	No	4	0	0	80606	1.1%
4	No	0	0	4	81981	-0.6%
4	No	0	4	0	**72744**	10.7%
4	No	1	2	1	74820	8.2%
5	Yes	5	0	0	82279	-1.0%
5	Yes	0	0	5	**72882**	10.6%
5	Yes	0	5	0	82423	-1.1%
5	Yes	1	3	1	76820	5.7%
5	No	5	0	0	77944	4.4%
5	No	0	0	5	78817	3.3%
5	No	0	5	0	83201	-2.1%
5	No	1	3	1	75256	7.7%
Merged Average					78375	3.8%

Table 5.2: Changing Implementation Strategies for Smaller MK Problems, 4-5 Cuts

implementation strategies used for larger problems. Examination of each experimental parameter in isolation is shown in Tables 5.3 - 5.5. In each instance, the average computational times of the best known results from all relevant lines in Tables 5.1 and 5.2 are averaged to show aggregated results for each experimental parameter.

Table 5.3 considers the different pseudo-costing strategies. Observe that many of the experimental runs in Tables 5.1 and 5.2 included a pure strategy (all reduced costs, all a values, or all balanced cuts). However, some of the experimental runs include a mixture of strategies such as the 3 cut scenario with 1 cut of each pseudo-costing strategy. Experiments of this type are listed under 'Mixture of Strategies' in Table 5.3.

Notice that each of the three pure strategies performed well, at about the same level of improvement. However, there may be some additional benefit to mixing pseudo-cost strategies if multiple merged inequalities are being generated. Any of the pseudo-costing

	Pseudo-Costing Strategy			
	All Reduced Costs	All Balanced	All a Values	Mixture of Strategies
Average Ticks	77835	76625	76274	73569
% Improvement	4.5%	6.0%	6.4%	9.8%

Table 5.3: Average Ticks of Pseudo-Costing Strategies from Tables 5.1 and 5.2

strategies may work well, but a mixed strategy is often preferred if multiple cuts are added. Thus, all of the same pseudo-costing strategies were considered for the larger MK instances presented later in this chapter.

Table 5.4 uses the experimental output from Tables 5.1 and 5.2 to compare whether or not overlapping rows is preferred when multiple merged inequalities are generated.

	Overlapping Rows	
	No	Yes
Computational Ticks	77095	75294
% Improvement	5.4%	7.7%

Table 5.4: Average Ticks of Overlapping Row Strategies from Tables 5.1 and 5.2

Merged inequalities almost always improved the computational time, regardless of the overlapping strategy. However, it appears that deliberate overlapping of rows provides even stronger results if multiple cutting planes are added. This is consistent with the theory motivating Theorem 5. Overlapping allows the algorithm to search in rows that had previously been used to generate a host cover inequality for an earlier merged cut. If the DCSA is employed, the algorithm may also search all candidate rows including those that had previously generated a host inequality.

It is easy to implement overlapping on general MK instances, and overlapping typically outperformed experimental variations without overlapping. Thus all future experimentation included overlapping as part of the recommended implementation strategy. Overlapping may occur by deliberately forcing an overlap while constructing merged inequalities or by use of the DCSA. Both types of overlapping were explored in greater detail in the larger MK instances.

The third experimental variation to consider is the number of merged inequalities that were generated and implemented before solving the MK problems. Since there are 5 rows for the smaller MK instances in this example, the experimental output from Tables 5.1 and 5.2 includes all $1, ..., m{=}5$ possible numbers of added merged inequalities assuming that each row serves as the host inequality no more than once. Table 5.5 shows the average performance of all strategies from Tables 5.1 and 5.2 that employ each number of constructed merged inequalities.

	Number of Merged Covers Added				
	1	2	3	4	5
Computational Ticks	72144	74157	74757	78181	78702
% Improvement	11.5%	9.0%	8.3%	4.2%	3.4%

Table 5.5: Average Ticks of Quantity of Cut Strategies from Tables 5.1 and 5.2

The benefit of inequality merging is immediately apparent when a single merged cutting plane is added. Adding two or three merged inequalities also performed well, with strong improvements over the baseline CPLEX ticks. It is not surprising that there are diminishing returns as the number of merged inequalities increases, with the worst performance occuring as the number of generated cuts approaches the m number of rows in the original A matrix. This is intuitive since each added merged inequality increases the size of the new A matrix during preprocessing, and matrix manipulation during the solution process is slowed by a larger A matrix.

This result was made even more clear by the separation of experimental output into Tables 5.1 and 5.2. The average performance from Table 5.1 reduced computational effort by about 9%, while the experimental runs with 4 or 5 merged inequalities in Table 5.2

provided an average improvement of only 4%. Thus, the best results appear to occur when the number of added merged inequalities is limited to 1, 2, or 3.

These experimental results from the smaller MK instances support three conclusions for use in larger MK instances:

i. Each of the pseudo-costing strategies performs well. If multiple merged inequalities are added, the best results appear to occur when using a variety of pseudo-costing strategies while generating multiple cutting planes. However, specific geometries of the MK instance may cause one strategy to outperform the others. All pseudo-costing strategies should be considered for larger problems.

ii. Deliberate overlapping of rows almost always outperformed the same experimental runs that prohibited overlapping rows. Only test strategies that use overlapping of rows for all larger problems.

iii. The number of added merged inequalities quickly creates diminishing returns because of additional computational requirements as the A matrix and basis grow in size. Preferred implementation strategies should focus on including 1, 2, or 3 merged cutting planes.

These conclusions provided critical insights for the best implementation strategies used for larger problems. As the problems increased in size, the computational time quickly increased. The same implementation strategies tended to yield the strongest results with larger problems, as shown in the next section.

5.3.2 Computational Results for Larger Problems

The earlier work was performed on relatively small MK problems that solved quickly. This facilitated an extensive analysis of the implementation strategies described above. However, the remaining problems in the computational study took considerably longer to solve. The remainder of the computational study focuses on MK instances with 10 constraints and 250 variables. These problems are representative of many real-world MK problems that could be solved within 1-2 days.

As before, the primary metric used to compare experimental strategies is the total computational effort (reported CPLEX ticks) required to solve the first 10 MK problems. Following the recommended implementation strategies from the previous section, a variety of pseudo-costing strategies are considered and all runs allow overlapping of rows. Experimentation focuses on the construction of 1, 2, or 3 merged inequalities, and the output from these runs is given in Table 5.6. For comparison, some experimentation with higher numbers of constructed merged inequalities is provided in Table 5.7.

As with the smaller MK problems, there were combinatorially many combinations of rows that could be tested for each implementation strategy. The experimentation considered many such variations. The best known results for each implementation strategy are shown in Tables 5.6 and 5.7.

# Merged Cuts Added	Pseudo-Costing Strategy Red. Costs	Balanced	a Values	Total Ticks (10 problems)	Percent Improvement
Baseline	0	0	0	30,994,459	Baseline
1	1	0	0	29,949,459	3.4%
1	0	0	1	30,268,076	2.3%
1	0	1	0	29,614,573	4.5%
2	2	0	0	**20,166,265**	**34.9%**
2	0	0	2	29,347,409	5.3%
2	0	2	0	30,881,549	0.4%
2	1	0	1	28,518,016	8.0%
2	0	1	1	29,975,494	3.3%
2	1	1	0	29,718,811	4.1%
3	3	0	0	**20,412,908**	**34.1%**
3	0	0	3	29,362,710	5.3%
3	0	3	0	30,908,925	0.3%
3	1	1	1	29,903,185	3.5%
Average				28,260,350	8.8%

Table 5.6: Changing Implementation Strategies for Larger MK Problems, 1-3 Cuts

# Merged Cuts Added	Pseudo-Costing Strategy Red. Costs	Balanced	a Values	Total Ticks (10 problems)	Percent Improvement
Baseline	0	0	0	30,994,459	Baseline
4	4	0	0	31,841,297	-2.7%
4	0	0	4	31,495,916	-1.6%
4	0	4	0	32,067,429	-3.5%
4	1	2	1	32,032,889	-3.4%
5	5	0	0	**20,789,585**	**32.9%**
5	0	0	5	31,172,648	-0.6%
5	0	5	0	32,866,353	-6.0%
5	1	3	1	31,273,350	-0.1%
10	10	0	0	37,095,448	-19.7%
Average	(all)			31,053,810	-0.2%
Average	(omits 10-cut run)			30,382,517	2.0%

Table 5.7: Changing Implementation Strategies for Larger MK Problems, 4 or more Cuts

Observe that restricting the number of merged inequalities to 1, 2, or 3 yielded strong computational results with an average improvement of about 9%. In contrast, adding 4 or more cuts often required more computational time than the baseline CPLEX settings. The following paragraphs explain the results from Tables 5.6 and 5.7 in greater detail.

Table 5.6 shows the best known results for the large MK problems when the recommended implementation strategies are followed. In these instances, inequality merging continues to provide an average improvement of about 9% over the baseline CPLEX computational effort. This is roughly the same level of improvement observed in the smaller MK instances. Most important, following the recommended implementation strategies always improved the solution times. This provides strong evidence that inequality merging is a beneficial technique for MK problems, and the reduction of computational ticks correlates to hours of time savings for large problems.

Notice that three of the set-ups in Tables 5.6 and 5.7 performed extremely well, reducing the computational ticks by more than 30%. The savings in each of these experiments corresponded to approximately 12 hours of computational time. In this case, all 3 of the best set-ups used pseudo-cost strategies that focus on reduced costs. Clearly a focus on reduced costs had the best impact for this particular grouping of larger MK instances, but that may not be the case in general. Previous analysis from Table 5.3 suggested that different pseudo-costing techniques may be preferred for particular problems, but focusing on reduced costs was actually the least preferred in that grouping of smaller MK instances. Identifying the reason that certain methods dominate other pseudo-costing techniques in particular problems is an excellent area for future research.

Table 5.7 also illustrates the significant impact of diminishing returns as the number of constructed merged inequalities exceeds three. The additional computational burden of adding too many new inequalities to the A matrix and basis caused negative 'improvements' in every case except one. Since the larger MK instances had 10 rows in the original A matrix, one example of adding 10 cuts is presented to accentuate the diminishing returns. In this 10-cut example, the best known result required almost 20% more computational ticks than the baseline CPLEX.

Further analysis of the output from larger MK instances also considers the two different types of overlapping. While all of these experimental runs used overlapping of some type, an additional policy difference was studied in this example. In some cases, randomly selected overlapping rows were determined *a priori*, restricting all overlaps to specific donor inequalities. In other instances, the algorithm searched every possible donor inequality to find the strongest coefficients for each variable. While the second policy has obvious theoretical appeal, additional computational time and less sparse cuts may imply diminishing returns when the DCSA searches every candidate donor. In some instances, randomly assigned specified rows outperformed searching all donor rows. In other instances, the DCSA gave the strongest results.

Table 5.8 shows the best solution times for each of the 10 MK instances in the larger files. In addition, the table also describes the implementation strategy that yielded the best result for each problem. Merging improved the solution times for each of the 10 problems, with an average reduction of computational requirements by 25.8%. However, the best single result for each sub-problem came from a wide variety of implementation strategies. These include instances that search all donor rows with the DCSA and other instances that consider only specified randomly-selected donor inequalities that define single overlaps.

Problem	Baseline	Merging	Percent		Implementation Strategy	
#	Ticks	Ticks	Improv.	Cuts	Pseudo-cost	Donor Rows
1	1,955,055	**128,467**	**93.4%**	3 cuts	Reduced Costs	All
2	203,122	160,209	21.1%	1 cuts	Balanced	Specified
3	316,729	265,573	16.2%	3 cuts	a Values	Specified
4	1,964,804	1,710,877	12.9%	2 cuts	Red. Cost & a Val.	Specified
5	6,735,442	6,300,815	6.4%	2 cuts	Red. Cost & a Val.	Specified
6	331,058	288,987	12.7%	1 cut	a Values	Specified
7	224,004	208,500	6.9%	1 cut	a Values	All
8	17,630,931	**5,993,211**	**66.0%**	5 cuts	Reduced Costs	Specified
9	651,113	563,288	13.5%	2 cuts	Red. Cost & a Val.	All
10	982,201	895,267	8.8%	3 cuts	a Values	Specified
Average			25.8%			

Table 5.8: Best Merging Performance by Problem for $m = 10$ and $n = 250$

Several important observations become apparent from the data in Table 5.8.

i. Solution times (ticks) were reduced in all 10 instances by employing some form of cover inequality merging.

ii. Problems 1 and 8 experienced dramatic reductions of 93% and 66% respectively. In both instances, the strongest implementation strategy focused on reduced costs. This is consistent with the conclusions from Tables 5.6 and 5.7 for this particular collection of MK instances.

iii. Observe that problem 8 took substantially longer than the other 9 problems given the baseline CPLEX 12.5 solver. Furthermore, the use of cover inequality merging reduced the computational requirements for this problem from over 17 million ticks to about 6 million ticks. This corresponded to a savings of about 13 hours of computational time. Not every set-up achieved this level of improvement in problem 8, but several implementation strategies achieved similar results. This dramatic improvement of a single large problem contributed to the majority of the strong results for the three best instances in Tables 5.6 and 5.7. A natural conclusion is that the greatest single benefit of cover inequality merging may exist when applying the technique to very difficult problems that require the most computational effort. It is difficult to predict which problems fall into this category, but it is an encouraging result that argues for the development of merged cutting planes to improve solution times for large or complex problems.

iv. The best results for the 10 subordinate problems include 3 instances with simultaneous merging on all rows (through the use of the DCSA) and 7 instances that specify the randomly selected donor overlap row(s) *a priori*. Additionally, observe that the two best results included one of each overlapping strategy, where the DCSA facilitated the single best percentage improvement in problem 1. It is clear that each strategy yields strong results in specific instances, and neither strategy obviously dominates the other.

The next natural extension would be application of the most successful implementation strategies to some of the very large problems proposed by Chu and Beasley in [19]. Some initial experimentation began on MK instances with A matrices of size 30×250 since there had been such strong results with the large problems in this study (10×250 A matrices) shown in Tables 5.6 and 5.7. Unfortunately, each of the 10 subordinate problems of size 30×250 took between 2-5 days to solve, often requiring over 100 million ticks each. The collective time required to solve all 10 MK instances exceeded three weeks. A very few experimental runs revealed some instances where inequality merging saved numerous hours of computational time, but it was impractical to conduct a thorough examination of such complex problems with different implementation strategies.

The computational study validates inequality merging as an effective technique that reduces computational time for multiple knapsack problems. Preferred implementation strategies provide the strongest results, yielding an average reduction of computational effort by about 9%. The computational study provides strong evidence that inequality merging yields productive cutting planes for MK problems, and it is likely that similar computational improvements will be achieved for other categories of general IP problems.

Chapter 6

Sequentially Merging Inequalities on Multiple Covers and Variables

The theoretical contributions and algorithms presented in Chapters 3 and 4 served as the foundation for the computational study shown in Chapter 5. Using the recommended implementation strategies, inequality merging has already reduced computational times for benchmark MK problems by about 9%. This exciting result encourages significant future research in constructing other merged inequalities that may reduce computational time for MK and general IP problems.

Several promising theoretical extensions will be discussed in Chapter 7 as areas for future research. The dissertation considered one extension in Chapter 4 by strengthening donor coefficients in merged inequalities. Chapter 6 provides an in-depth treatment of a second theoretical extension called sequentially merging multiple cover inequalities on multiple variables. This chapter provides theoretical foundations, an implementation algorithm, a theorem presenting conditions for validity, and an example problem to highlight the potential of this sequential merging technique on multiple variables.

The motivation for sequentially merging on multiple variables has its roots in the computational study. Recall that generating and implementing large numbers of merged cuts contributed to significant diminishing returns when the number of merged inequalities became too large. It is also likely that some of the information from multiple merged cuts is seen as redundant. Thus large numbers of cuts slowed down the calculations as the constraint matrix and basis grew in size. However, it appears to be theoretically and computationally desirable to include more merges with increased information concerning a higher number of variables with non-zero coefficients.

The sequential merging technique described in this chapter offers many of the same theoretical benefits of multiple merged cuts, but the information is combined into a single sequentially merged inequality. This strategy yields a merged inequality with more non-zero coefficients than the merging techniques presented in Chapters 3 and 4, and the example instance in this chapter yields a merged inequality where all variables achieve non-zero coefficients. The technique should be well suited for MK instances and other IP problems with sparse matrices.

6.1 Theoretical Foundations for Sequential Merging

The idea of sequentially merging multiple cover inequalities on multiple variables is to repeatedly merge cover inequalities on distinct variables from the host inequality which yields a single merged inequality. Restricting to host and donor cover inequalities provides some starting results for this concept.

Given a MK instance and a C^{host} cover from row r, a sequentially merged inequality on multiple covers and variables follows the following procedure. An index $p \in C^{host}$ is selected and a donor cover that contains p is found in some row s. This creates a merged inequality following Theorem 3. A new index $p' \in C^{host} \setminus \{p\}$ is selected. Another donor cover that contains p' is found in some row s', and that cover is then merged into the previously merged inequality. This process can continue through all of the indices in C^{host}, but it is dependent upon the particular instance.

Since multiple merges may occur on distinct variables in C^{host} and these merges occur sequentially, several new definitions are necessary. Define the remaining host indices $C^{rem\ host}$ to be the indices in C^{host} from row r that have not yet been merged. Furthermore, define the indices from C^{host} that have already been merged to be $C^{merge\ host}$. Clearly $C^{rem\ host} \cup C^{merge\ host} = C^{host}$.

The indices in $C^{rem\ host}$ have $\alpha_j = 1$ in the current sequentially merged inequality because C^{host} is a cover and the indices in $C^{rem\ host}$ have not yet been merged. Define C_p^{donor} to be the donor cover that merges on index p for $p \in C^{rem\ host}$. It is further assumed that $p \in C_p^{donor}$. The indices in $C^{merge\ host}$ have $\alpha_j = \frac{1}{|C_j^{donor}|-1}$. In general, requiring $p \in C_p^{donor}$ is not necessary, and the reader should be able to make the straightforward adjustments if so desired.

A sequentially merged inequality using multiple covers and variables takes the form

$$\sum_{i \in C^{rem\ host}} x_i + \sum_{p \in C^{merge\ host}} \frac{1}{|C_p^{donor}| - 1} \sum_{i \in C_p^{donor}} x_i \leq |C^{host}| - 1.$$

To create a sequentially merged inequality on multiple variables requires a sequentially merged inequality with $C^{rem\ host}$ and $C^{merge\ host}$. Merging selects a $p' \in C^{rem\ host}$ and then finds a cover that contains p'.

The purpose of ψ from the previous chapter is to determine which indices are eligible based solely on constraint r. A similar strategy exists in a sequentially merged inequality and is denoted as $\psi_{p'}^S$. This new value involves the previously merged donor cover's coefficients. Define $a'_p = (\sum_{i \in C_p^{donor}} a_{r,i}) - max_{i \in C_p^{donor}} a_{r,i}$ for each $p \in C^{merge\ host}$. Also, let

$$\psi_{p'}^S = b_r - (\sum_{i \in C^{rem\ host} \setminus \{p'\}} a_{r,j} + \sum_{p \in C^{merge\ host}} a'_p) + 1.$$

Define a taboo list of indices that includes members of any C_p^{donor} sets and any member of C^{host} except for index p'. Denote this taboo list as $T_{p'} = (C^{host} \setminus \{p'\}) \cup_{p \in C^{merge\ host}} C_p^{donor}$. The candidate indices for finding a donor cover are now denoted as $N_{\psi_{p'}^S} = \{i \in N \setminus T_p' : a_{r,i} \geq \psi_{p'}^S\}$. A cover that includes p' is found in any row and it becomes $C_{p'}^{donor}$. The index p' is now taken away from $C^{rem\ host}$ and added to $C^{merge\ host}$. Additional nonzero coefficients are added to the sequentially merged inequality.

This process of sequentially merging inequalities may occur iteratively for previously unmerged indices in C^{host}, and it can potentially continue for each index in C^{host} in which case $C^{rem\ host} = \emptyset$. However, there is no guarantee that such donor covers exist for a given index. Furthermore, merging certain donor covers may create an invalid inequality.

The Sequential Cover Merging Algorithm (SCMA) determines the validity of a sequentially merged inequality after all of the merging is completed. An interesting aspect is that an interim sequentially merged inequality may be invalid, but the final merged inequality may be valid. Thus, one may wish to only check for validity after all sequential merging is complete.

The process begins by creating a table with one set of two columns, T^α and T^a. Each row of the table reflects a number of indices selected from C_i^{donor} for each $i \in C^{host}$. If no indices are selected from C_i^{donor}, then $T_i^\alpha = 0$ and $T_i^a = 0$, which is the first row of the table. If an index $i \in C^{host}$ is not merged, then the a column for index i has $a_{r,i}$ and the α column has a 1. If $i \in C^{host}$ is merged, then the a column for index i has $|C^{donor}|$ entries. The a values are $\displaystyle\sum_{q=|C_i^{donor}|-j+1}^{|C_i^{donor}|} a_{r,q}$ with the α values equal to $\frac{j}{|C_i^{donor}|-1}$ for each $j \in C_i^{donor}$. In this formula, it is assumed that C_i^{donor} is presented in descending order according to the coefficients in row r of the A matrix.

The SCMA verifies validity by exhaustively checking instances where setting variables to one force the α value over $|C_i^{host}| - 1$. If such a point is feasible in row r, then the sequentially merged inequality cannot be guaranteed to be valid. The input to SCMA is a multiple knapsack instance, a host cover, associated donor covers, and a sequentially merged inequality on multiple variables. The formal description of SCMA is as follows.

Sequential Cover Merging Algorithm

```
a. Initialization
      For each $i \in C^{host}$
       $T^a_{i,0} \leftarrow 0$
       $T^\alpha_{i,0} \leftarrow 0$
         If $i \in C^{rem\ host}$, Then
            $T^a_{i,1} = a_{r,i}$ and $T^\alpha_{i,1} \leftarrow 1$
         end $if$
         If $i \in C^{merge\ host}$, Then
            For ($j = 1$ to $|C^{donor}_i|$)
               $T^a_{i,j} \leftarrow T^a_{i,j-1} + a_{r,|C^{donor}_i|-j+1}$
               $T^\alpha_{i,j} \leftarrow T^\alpha_{i,j-1} + \frac{1}{|C^{donor}_i|-1}$
            end $for$
         end $if$
      end $for$
b. Main Step
      For each $i \in C^{host}$
         $counter_i \leftarrow 0$
```

```
    If $(i \in C^{rem\ host})$, Then
      $c_i \leftarrow 1$
    Else
      $c_i \leftarrow |C_i^{donor}|$
    end $else$
$a \leftarrow 0$
$\alpha \leftarrow 0$
While $counter_1 \leq |C_1^{donor}|$
    If $\alpha > |C^{host}| - 1$ and $a \leq b_r$, Then
      Report sequentially merged inequality as invalid and terminate.
    Else
      $counter_{|C^{host}|} \leftarrow counter_{|C^{host}|} + 1$
      If $counter_{|C^{host}|} \leq |C^{host}|$, Then
        $\alpha \leftarrow \alpha + T^{\alpha}_{|C^{host}|, counter_{|C^{host}|}} - T^{\alpha}_{|C^{host}|, counter_{|C^{host}|-1}}$
        $a \leftarrow a + T^{a}_{|C^{host}|, counter_{|C^{host}|}} - T^{a}_{|C^{host}|, counter_{|C^{host}|-1}}$
      Else
        $q \leftarrow |C^{host}|$
        While $(counter_q > c_q)$
          $a \leftarrow a - T^{a}_{q,c_q}$
          $\alpha \leftarrow \alpha - T^{\alpha}_{q,c_q}$
          $counter_q \leftarrow 0$
          $q \leftarrow q - 1$
```

```
            $counter_q \leftarrow counter_q + 1$
            if $(counter_q \leq c_q)$, Then
              $\alpha \leftarrow \alpha + T^{\alpha}_{q,\ counter_q} - T^{\alpha}_{q,\ counter_q - 1}$
              $a \leftarrow a + T^{a}_{q,\ counter_q} - T^{a}_{q,\ counter_q - 1}$
          end $while$
        end $else$
      end $while$
    end $else$
  end $for$
```

c. Output

Report that the sequentially merged inequality is valid.

SCMA is an exhaustive enumerative algorithm because the $counter_{|C^{host}|}$ increases by one each iteration. If this counter ever exceeds the number of indices associated with the last index in C^{host}, then this counter reverts to 0 and the $counter_{|C^{host}|-1}$ increases by one. If $counter_{|C^{host}|-1}$ exceeds the number of indices associated with this index in C^{host}, then its counter is reset to zero and $counter_{|C^{host}|-2}$ increases by one. This logic continues and only terminates when $counter_1$ exceeds its number of associated indices. At each step, necessary updates are made to a and α. If $\alpha > |C^{host}| - 1$ and $a \leq b$, then the algorithm reports that the inequality is not valid. Thus, SCMA exhaustively examines each opportunity to discover a hypothetical invalid point, and it verifies that such a point is not feasible in row r.

The initialization of SCMA trivially requires $O(|C^{host}| max_{i \in C^{host}} |C_i^{donor}|)$ to create the table for each index in C^{host}. Each step of the main step requires $O(1)$ effort. The $counter_q$

term is incremented $(\Pi_{i\in C^{rem\ host}}2)\Pi_{i\in C^{merge\ host}}(|C_i^{donor}|+1)$ many times. Thus, the main step requires $O(2^{|C^{rem\ host}|}\Pi_{i\in C^{merge\ host}}|C_i^{donor}|)$ effort. The worst case happens with only minimal merging where $|C^{host}|$ is approximately n. Thus the SCMA is bounded by $O(2^n)$.

It is important to use care when implementing the SCMA. The algorithm requires exponential effort, and worst case scenarios approach $O(2^n)$. However, the run time is much faster as the number of sequential merges increases and the size $|C^{rem\ host}|$ decreases. Since the algorithm is designed to verify inequalities that have experienced multiple sequential merges, practical implementations experience much lower computational effort. Improvements to the design of SCMA may decrease computational effort in some instances, and this possibility is recommended as an area of future research.

The following theorem provides conditions for which sequential merging on multiple variables creates a valid inequality.

Theorem 6. *Given a MK instance and a valid sequentially merged inequality on C^{host} with $C^{rem\ host}\subset C^{host}$ of the form $\sum_{i\in C^{rem\ host}} x_i + \sum_{p\in C^{merge\ host}} \frac{1}{|C_p^{donor}|-1}\sum_{i\in C_p^{donor}} x_i \leq |C^{host}|-1$. Let $p'\in C^{rem\ host}$ and $C_{p'}^{donor}\subseteq N_{\psi_{p'}^S}$. If SCMA reports that the inequality is valid, then the sequentially merged inequality is a valid inequality of $conv(P_{MK})$.*

Proof. For contradiction, assume that SCMA reports that $\sum_{i\in C^{rem\ host}} x_i + \sum_{p\in C^{merge\ host}} \frac{1}{|C_p^{donor}|-1}\sum_{i\in C_p^{donor}} x_i \leq |C^{host}|-1$ is valid, but it is not. There must exist an $x'\in P_{MK}$ such that $\sum_{i\in C^{rem\ host}} x'_i + \sum_{p\in C^{merge\ host}} \frac{1}{|C_p^{donor}|-1}\sum_{i\in C_p^{donor}} x'_i > |C^{host}|-1$. For each $q\in C^{rem\ host}$ define $cnt_q=1$ if $x'_q=1$ and 0 if not. For each $q\in C^{merge\ host}$ define $cnt_q=\sum_{i\in C_q^{donor}} x'_i$.

Since the mainstep of the *SCMA* is an exhaustive check, one iteration of the SCMA has $counter_q = cnt_q$ for each $q\in C^{host}$. During this iteration, the value of α exceeds $|C^{host}|-1$.

The value of a is $\sum\limits_{q \in C^{host}} T^a_{q,cnt_q}$. Since x' is feasible, $\sum_{i \in N} a_{r,i}x'_i \leq b_r$. Clearly $a \leq \sum\limits_{i \in N} a_{r,i}x'_i$ and thus $a \leq b_r$. However, SCMA would have reported that the inequality is not valid, a contradiction and the result follows.

$\square$

The SCMA serves as an excellent screening tool. According to the discussion in Theorem 6, any sequentially merged inequality that passes the examination of SMCA is provably valid. However, it is possible for SCMA to report that the merged inequality is invalid, when in actuality the inequality is valid. The SCMA only checks row r for validity. In certain situations, a point or set of points, denoted by X, may appear to be feasible in row r, but the points are infeasible due to some other row s. In such an instance, $\sum\limits_{i \in N} a_{r,i}x'_i \leq b_r$, but $\sum\limits_{i \in N} a_{s,i}x'_i > b_s$ in some other row s for each $x' \in X$. In such a scenario, the SCMA reported an invalid inequality, but $\sum\limits_{i \in C^{rem\ host}} x_i + \sum\limits_{p \in C^{merge\ host}} \frac{1}{|C^{donor}_p| - 1} \sum\limits_{i \in C^{donor}_p} x_i \leq |C^{host}| - 1$ could be valid.

A modification to SCMA could check all other row combinations before terminating. However, there would be exponentially many such checks, potentially requiring an untenable amount of computational effort. Thus, the SCMA serves as a screening tool and terminates if row r does not prove validity.

6.2 Example Problem showing Sequential Merging on Multiple Variables

Consider the multiple knapsack instance shown below with $n = 12$ and $m = 4$, where

$$A = \begin{bmatrix} 21 & 19 & 17 & 13 & 12 & 8 & 8 & 7 & 7 & 6 & 4 & 3 \\ 18 & 6 & 5 & 14 & 19 & 5 & 8 & 6 & 4 & 8 & 6 & 7 \\ 7 & 14 & 8 & 7 & 6 & 13 & 19 & 12 & 5 & 3 & 4 & 8 \\ 7 & 5 & 20 & 6 & 4 & 5 & 8 & 7 & 13 & 10 & 12 & 14 \end{bmatrix}$$

and

$$b = \begin{bmatrix} 44 \\ 50 \\ 57 \\ 67 \end{bmatrix}.$$

Designate the first constraint as the host with $r = 1$, and assume that pseudo-costing sorts the indices according to the size of the coefficients in the host inequality. Using this pseudo-cost strategy, the host cover is $\{1, 2, 3\}$, and the corresponding inequality is $x_1 + x_2 + x_3 \leq 2$.

Sequentially merging may occur first on any of these three variables, and this example demonstrates one merge using each of the members of C^{host} as the variable x_p. First, consider merging on index 1. In order to determine the eligible indices for C_1^{donor}, it is necessary to calculate ψ_1^S. For the first merge, this follows the initial procedure described in Chapter 4. Since $a_{1,2} + a_{1,3} = 36$ and the right-hand side is 44, $\psi_1^S = (44 - 36) + 1 = 9$. Indices 2 and 3 are taboo, since they are in C^{host} and are not the merging index. Since $\psi_1^S = 9$, all eligible (non-taboo) indices in the host constraint must have coefficients that are greater than or equal to 9 in row 1. Thus, $\psi_1^S = 9$ implies that indices 4 and 5 are the only eligible indices that may join index 1 in C_1^{donor}, since $a_{1,4}$ and $a_{1,5}$ are ≥ 9. Thus $N_{\psi_1^S} = \{1, 4, 5\}$.

Observe that indices 1, 4, and 5 form a cover in row 2. Since this merge is on index 1, $C_1^{donor} = \{1, 4, 5\}$, with corresponding inequality $x_1 + x_4 + x_5 \leq 2$. Merging C^{host} with C_1^{donor} yields

$$\frac{1}{2}x_1 + x_2 + x_3 + \frac{1}{2}x_4 + \frac{1}{2}x_5 \leq 2. \tag{6.1}$$

Next, consider merging the host cover inequality on index 2 with knowledge of C_1^{donor}. The first step is to calculate ψ_2^S. Observe that this process is more complicated than in the past because there are many combinations of indices that satisfy 6.1 at equality. However, the choice of indices must also satisfy the host constraint. The calculation of ψ_2^S requires choosing terms from (6.1) such that their coefficients sum to exactly 1 in (6.1). This involves selecting the terms with the smallest possible sum of $a_{1,i}$ values from each cover.

In this instance, index 2 is not considered because it becomes the merged index. This implies that the model selects index 3 (with coefficient 1) from the $C^{rem\ host}$ indices, and $a_{1,3} = 17$. Now, 2 of the 3 indices from C_1^{donor} must be selected since each of the three terms has coefficient $\frac{1}{2}$ in (6.1). In particular, the two indices from C_1^{donor} with the smallest $a_{1,i}$ values are selected. Observe that $a_{1,1} = 21$, $a_{1,4} = 13$, and $a_{1,5} = 12$. Thus indices 4 and 5 are selected, where $a_{1,4} + a_{1,5} = 25 = a'_1$.

Accordingly, the smallest summed value of coefficients that satisfies the host constraint and satisfies (6.1) at equality is attained by selecting indices 3, 4, and 5. The sum of their coefficients $a_{1,3}$, $a_{1,4}$, and $a_{1,5}$ yield the value of 42. Thus, $\psi_2^S = (44 - 42) + 1 = 3$. Notice that all indices in the host inequality have $a_{1,i}$ values greater than or equal to 3. Thus, any of the indices satisfy the restrictions imposed by ψ_2^S.

However, it is important to exclude any indices that are taboo. Indices 1 and 3 are taboo since they are non-merging components of C^{host}. Additionally, indices 4 and 5 are also taboo since they are in C_1^{donor}. Thus $N_{\psi_2^S} = \{2, 6, 7, 8, 9, 10, 11, 12\}$. Observe that row 3 contains a cover that satisfies these conditions, where $C_2^{donor} = \{2, 6, 7, 8\}$ with corresponding inequality $x_2 + x_6 + x_7 + x_8 \leq 3$.

Merging C^{host} with C_1^{donor} and C_2^{donor} sequentially is the same as merging (6.1) with C_2^{donor}, yielding

$$\frac{1}{2}x_1 + \frac{1}{3}x_2 + x_3 + \frac{1}{2}x_4 + \frac{1}{2}x_5 + \frac{1}{3}x_6 + \frac{1}{3}x_7 + \frac{1}{3}x_8 \leq 2. \tag{6.2}$$

Finally, consider merging the host cover on index 3. As before, the first step in this process is the calculation of ψ_3^S. The calculation of ψ_3^S requires that if (6.2) is satisfied at equality, then including any index in $N_{\psi_3^S}$ must force a cover (infeasibility) in the host row.

While there may exist many methods for (6.2) to be satisfied at equality, calculating ψ_3^S only considers the following case. Since $C^{rem\ host} = \emptyset$, the coefficients of merged terms from both C_2^{donor} and C_1^{donor} in (6.2) must sum to 1. Additionally, the terms from both C_2^{donor} and C_1^{donor} are selected that minimize the sum of $a_{1,i}$ values.

Achieving a weight of 1 from both C_2^{donor} and C_1^{donor} satisfies (6.2) at equality yielding a minimized sum of $a_{1,i}$ values of 48. This corresponds to indices 4 and 5 contributing $13 + 12 = 25 = a_1'$ from the C_1^{donor} terms and indices 6, 7, and 8 contributing $8 + 8 + 7 = 23 = a_2'$ from the C_2^{donor} terms. Thus, $\psi_3^S = (44 - 48) + 1 = -3$. Since all coefficients in the host row are non-negative, all indices satisfy the eligibility requirements associated with ψ_3^S.

However, it is again important to consider which indices are taboo. Indices 1 and 2 are taboo from C^{host}, indices 4 and 5 are taboo from C_1^{donor}, and indices 6, 7, and 8 are taboo from C_2^{donor}. Thus $N_{\psi_3^S} = \{3, 9, 10, 11, 12\}$. Notice that row 4 contains a 5-element cover with these indices. Thus $C_3^{donor} = \{3, 9, 10, 11, 12\}$ with corresponding inequality $x_3 + x_9 + x_{10} + x_{11} + x_{12} \leq 4$. Sequentially merging C^{host} with C_1^{donor}, C_2^{donor}, and C_3^{donor} is the same as merging (6.2) with C_3^{donor}, yielding

$$\frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 + \frac{1}{2}x_4 + \frac{1}{2}x_5 + \frac{1}{3}x_6 + \frac{1}{3}x_7 + \frac{1}{3}x_8 + \frac{1}{4}x_9 + \frac{1}{4}x_{10} + \frac{1}{4}x_{11} + \frac{1}{4}x_{12} \leq 2. \quad (6.3)$$

The SCMA is used to verify the validity of (6.3). The SCMA begins by creating Table 6.1, which shows the minimum possible sum of $a_{1,i}$ coefficients depending on how many indices associated with each fractional coefficient in (6.3) are selected. If (6.3) does not remove a valid point from the host constraint in row 1 of the A matrix, then (6.3) is valid.

Depth	C_1^{donor}		C_2^{donor}		C_3^{donor}	
i	$T_{1,i}^{\alpha}$	$T_{1,i}^{a}$	$T_{2,i}^{\alpha}$	$T_{2,i}^{a}$	$T_{3,i}^{\alpha}$	$T_{3,i}^{a}$
0	0	0	0	0	0	0
1	$\frac{1}{2}$	12	$\frac{1}{3}$	7	$\frac{1}{4}$	3
2	1	25	$\frac{2}{3}$	15	$\frac{2}{4}$	7
3	$\frac{3}{2}$	46	1	23	$\frac{3}{4}$	13
4			$\frac{4}{3}$	42	1	20
5					$\frac{5}{4}$	37

Table 6.1: The T^{α} and T^{a} Table for Sequential Merging from the SCMA

The SCMA uses the data in Table 6.1 to verify the validity of (6.3). Observe that (6.3) is invalid if there is an instance where the sum of $T_{i,counter_i}^{\alpha}$ values across all indices $i \in C^{host}$ is strictly greater than 2, but the corresponding sum of $T_{i,counter_i}^{a}$ values are ≤ 44 since $b_r = 44$. All such combinations of $T_{i,counter_i}^{\alpha}$ and $T_{i,counter_i}^{a}$ values are considered exhaustively.

Consider one instance that uses Table 6.1 as an example. Let $counter_3 = 3$, $counter_2 = 2$ and $counter_1 = 2$. Then $T_{3,3}^{\alpha} = \frac{3}{4}$ and $T_{3,3}^{a} = 13$, $T_{2,2}^{\alpha} = \frac{2}{3}$ and $T_{2,2}^{a} = 15$, and $T_{1,2}^{\alpha} = 1$ and $T_{1,2}^{a} = 25$. Observe that $T_{3,3}^{\alpha} + T_{2,2}^{\alpha} + T_{1,2}^{\alpha} = \frac{3}{4} + \frac{2}{3} + 1 = \frac{29}{12} > 2$. In this instance,

$T^a_{3,3} + T^a_{2,2} + T^a_{1,2} = 13 + 15 + 25 = 53 > 44$. If 53 was less than $b_r = 44$, then the SCMA would report that (6.3) is invalid and terminate. Clearly, this instance did not yield an invalid circumstance. The SCMA now moves to the next instance for inspection, which is $counter_3 = 4$, $counter_2 = 2$ and $counter_1 = 2$. This process repeats exhaustively until all such combinations have been inspected or an invalid instance is discovered.

An exhaustive inspection using Table 6.1 verifies the validity of (6.3). Now, it is important to demonstrate the usefulness of this merged inequality by demonstrating that it is a cutting plane.

The linear relaxation point that maximizes the value achieved by the left-hand side of (6.3) is the point (0, 0, 0, 0, 0.5793, 1, 0.5315, 1, 0.6851, 1, 1, 1) with a right-hand side of 2.0547. This linear relaxation point is valid in all 4 MK constraints, but it is removed by (6.3). Thus (6.3) is a cutting plane. One criticism may argue that it is extremely unlikely to consider such unusual decimal values and 4 indices in that point are zero. Consequently, more reasonable points were pursued that also demonstrate the usefulness of the cutting plane in (6.3). Many such points exist, and a more reasonable example is the point ($\frac{1}{100}$, $\frac{1}{100}$, $\frac{1}{100}$, $\frac{1}{10}$, $\frac{4}{10}$, $\frac{3}{4}$, $\frac{3}{4}$, $\frac{3}{4}$, 1, 1, 1, 1). This point has a right-hand side of 2.011 in (6.3), it provides coefficients for each index in (6.3), and all coefficients are reasonable. Furthermore, this point is valid in all 4 MK constraints but it is removed by (6.3). Thus this point provides evidence that (6.3) is a cutting plane. This demonstrates that the merged inequality has a strong potential to be computationally useful.

In this example, a non-zero coefficient for every variable in the MK instance was attained. The existence of the C^{host} cover inequality is nearly impossible to distinguish in (6.3) after completing the sequential merging process. The cover inequality was formerly $x_1 + x_2 + x_3 \leq 2$, but those indices now appear as $\frac{1}{2}x_1 + \frac{1}{3}x_2 + \frac{1}{4}x_3 \leq 2$. Examining any donor cover inequality also results in similar phenomenon. For instance, C_2^{donor} originally appeared as $x_2 + x_6 + x_7 + x_8 \leq 3$, but sequentially merging transforms the C_2^{donor} indices into

$\frac{1}{3}x_2 + \frac{1}{3}x_6 + \frac{1}{3}x_7 + \frac{1}{3}x_8 \leq 2$. It is thus apparent that sequentially merged inequalities on multiple covers and variables is not an extension of prior work in this dissertation or other known cutting plane generation technique. Consequently, sequential merging on multiple covers and variables yields a new class of cutting planes.

The coefficients obtained for sequentially merged indices may not be the strongest possible values like those achieved through simultaneous merging and the DCSA in Chapter 4. However, the primary benefit of sequentially merging on multiple variables is that this technique typically yields more non-zero coefficients in the obtained inequality than any of the other merging techniques discussed in this dissertation. If a single merge was performed in this example, the attained inequality would have been (6.1), including non-zero coefficients on variables $x_1, ..., x_5$. Instead, sequentially merging on multiple variables yielded (6.3) with non-zero coefficients on variables $x_1, ..., x_{12}$.

Thus sequential merging on multiple variables may combine information from multiple donor inequalities into a single merged cutting plane. This technique may avoid the diminishing returns observed in Chapter 5 when more than 3 merged cutting planes were introduced into the constraint matrix and basis. Iterative use of sequential merging on multiple variables could potentially condense information from higher numbers of single-variable merged inequalities into 2 or 3 sequentially merged inequalities. This is an obvious extension to consider as future research.

Lastly, sequential merging on multiple variables has an intuitive appeal for sparse matrices in multiple knapsack problems. Many practical applications of MK problems contain extremely sparse A matrices because of the structure of the knapsack constraints. Some recent examples include work by Salman, et. al. [66], Kalagnanam, et. al. [53], and Dawande, et. al. [23]. In such instances, it may be easier to find valid donor cover inequalities for sequential merging on multiple variables.

Chapter 7

Conclusions and Future Research

This dissertation introduces a previously undiscovered category of cutting planes for integer programming problems called inequality merging. The dissertation provides the theoretical foundations for inequality merging through several theorems. The document also includes numerous algorithms to assist the construction of merged inequalities. A computational study validates the benefit of implementing merged inequalities for multiple knapsack problems, reducing average computational time by about 9% over baseline CPLEX settings when recommended implementation procedures are followed.

The theory presented in this document is an introduction to a new class of inequalities, and inequality merging has a rich potential for additional theoretical discoveries and implementation techniques that may solve integer programming problems more quickly. While MK instances provide the structure for many of the results in this document, the technique can be applied to many other types of IPs through straightforward extension and substitution. Two theoretical extensions are presented in detail in this dissertation, known as simultaneous merging over multiple donor cover inequalities and sequential inequality merging on multiple covers and variables. Both are demonstrated through example MK instances. Several other possibilities are recommended as areas for future research.

7.1 Conclusions

The initial theoretical results for inequality merging are introduced in Chapter 3. Theorem 1 expresses conditions for validity of merged inequalities in general IP instances. When the host inequality is restricted to a cover inequality, verification of validity may be achieved in polynomial time using the MOCIA. Merging two valid inequalities typically generates a theoretically stronger valid inequality, and Theorem 2 provides conditions under which a valid merged inequality in a general IP instance may be facet defining. Corollary 1 extends this result to a MK instance that merges inequalities associated with two minimal covers.

An example problem in Chapter 3 demonstrates the basic theoretical results, yielding a facet-inducing merged inequality using the general theory. The resulting merged inequality is shown to be unattainable through any straightforward implementation of previously known cutting plane techniques or lifting procedures. Thus, the general inequality lifting theory yields a new category of cutting planes for IPs.

Inequality merging is readily employed for situations like multiple knapsack problems, when it is relatively easy to find valid inequalities such as cover inequalities. Consequently, theoretical extensions and a computational study focus on the MK. These results merge a host cover inequality with 1 or more donor inequalities in a MK instance. Theorem 3 provides conditions for validity when two such cover inequalities are merged. To assist the process of merging two cover inequalities, Chapter 4 introduces the ψ_p threshold as a technique to determine which indices are eligible members of C^{donor} when merging occurs with C^{host} on x_p. N_{ψ_p} represents this set of eligible indices. Theorem 4 provides conditions under which the proper use of the ψ_p threshold yields a valid merged inequality in a MK instance.

The ψ_p threshold is used to generate donor cover inequalities in applied computational problems. However, an unfortunate selection of indices in C^{host} or a poor choice of x_p may result in a scenario that fails to find C^{donor} because ψ_p is too large. Chapter 4 provides the Reducing ψ_p Algorithm to address this possibility while implementing inequality merging in practice. Successful iterative application of this algorithm incrementally reduces ψ_p until it is possible to merge C^{host} and C^{donor} on x_p yielding a valid merged inequality in a MK instance.

The first extension considered in detail in this dissertation is the idea of merging over multiple donor cover inequalities simultaneously. This technique determines the strongest (largest) coefficient for each eligible donor index in N_{ψ_p}, thus theoretically strengthening the merged inequality. Theorem 5 provides conditions under which such a merged inequality is valid. Chapter 4 also presents the DCSA, which is the corresponding implementation algorithm for simultaneous merging across multiple donor inequalities. The DCSA requires cubic effort, but it runs quickly in practice. The DCSA is used to implement simultaneous merging across multiple donor inequalities as part of the computational study in this dissertation.

A thorough computational study validates the benefits of inequality merging in internationally recognized benchmark MK instances available through the $OR - Library$. The computational study considers a variety of implementation strategies, including the pseudo-costing method to construct C^{host}, the number of merged cutting planes added to the constraint matrix and basis, the strategy of overlapping rows when multiple cuts are added, and solutions both with and without simultaneous merging across multiple donor covers.

Substantial experimentation on smaller MK instances motivated important conclusions that defined the "best practices" for implementing inequality merging methods. The best performance usually occurred when 3 or fewer merged cuts were generated, and it was consistently preferred to overlap rows when multiple merged cutting planes were generated.

Overlapping may include merging on all rows simultaneously or on a randomly selected subset of rows considered for donor cover inequalities. Both polices supported improved results in many instances. Each of the pseudo-costing strategies worked in different scenarios, depending on the geometry of the problem.

Cover inequality merging offered excellent average improvements in computational time for MK instances. Compared against the CPLEX 12.5 solver with baseline settings, inequality merging is shown to reduce computational time by about 9% on average when the recommended implementation strategies are followed. The computational improvement percentages were shown to be consistent on both small and larger MK instances, but the computational time savings become more significant in practice for some of the larger MK problems. In some interesting cases, inequality merging in large MK instances reduced computational time by more than 30%.

Finally, a second theoretical extension is considered in detail in Chapter 6. Sequentially merging on multiple covers and variables provides a merged cutting plane construction technique that increases the number of merged indices with non-zero coefficients. The process sequentially merges on multiple indices in C^{host}, combining information from multiple donor covers into a single merged cutting plane. The SCMA implementation algorithm provides a quick check to verify the validity of a sequentially merged inequality, and Theorem 6 guarantees that a sequentially merged inequality is valid when the SCMA reports validity. This extension may have strong computational benefits because it combines information from multiple single-variable merges into one merged cutting plane.

7.2 Future Research

Inequality merging offers the potential for significant future research, both computational and theoretical in nature. This dissertation provides the theoretical foundations and implementation algorithms for several types of inequality merging, while numerous extensions are likely to continue to improve our understanding of merged cutting planes for IPs. Several such ideas are considered in the paragraphs below.

7.2.1 Topics for Theoretical Extensions for Inequality Merging

The techniques described in this dissertation describe a new technique to create valid cutting planes for integer programming problems. This research provides a strong foundation, which should inspire future theoretical extensions. A few of these topics are described in this section.

The general theory for inequality merging applies to general integer programming problems. However, the work in this dissertation assumes that the coefficient of the merging variable x_p in C^1 or C^{host} is 1. This naturally applies to cover inequalities, but candidate merging inequalities in general IPs may have higher integral values or even fractional values. Future research may extend the theory to more general instances of this type.

A method to prove validity for sequential merging on multiple variables was shown in Chapter 6. Future research may provide conditions under which this technique can yield facet-defining merged inequalities. It may also be possible to implement the sequential merging process across multiple variables several times for the same MK instance. Thus, iterative merging of merged inequalities would combine the information from many single-variable merged cutting planes into a few sequentially merged inequalities.

During each iteration of sequential merging, it may also be possible to merge across multiple donor inequalities simultaneously. This may combine some of the ideas presented as simultaneous merging over multiple donors (from Chapter 4) with sequential merging on multiple variables (from Chapter 6) into a single technique. Such a methodology may have computational appeal because it could theoretically strengthen merged coefficients, while yielding a single merged cutting plane with increased numbers of merged variables that have non-zero coefficients.

Another possibility is to simultaneously merge two or more inequalities over multiple variables x_{p_1}, x_{p_2}, ..., x_{p_q} that occur in the host cover and all donor covers across multiple constraints. This should require the development of a new type of ψ that takes information from ψ_{p_1}, ψ_{p_2}, ..., ψ_{p_q}.

These represent rich opportunities for theoretical extensions that apply the general theory to more categories of merged covers. In addition to the obvious applications of integer programming, theoretical extensions for inequality merging may also apply to research concerning conflict hypergraphs and other similar graph structures. Thus, there may be some similarities or applications to hypergraph research by Atamtürk et al. [6], Easton et al. [32], and Hooker and Easton [49].

7.2.2 Computational Studies and Algorithm Contributions as Future Research

In addition to the possible theoretical extensions, future research may include new computational studies or new algorithm improvements to implement inequality merging in practice.

One important contribution will be a better understanding of the properties that make some rows in a MK instance better choices as host inequalities than other rows. During the computational study, it was observed that significant variability in computational time may exist when solving the same MK problems using the same implementation strategies but different host inequalities were selected for multiple trials. While the current method selects a host inequality row at random, another implementation technique may be discovered that selects a host inequality row more deterministically.

The implementation algorithms in this dissertation are performed as preprocessing, but some of them require significant computational effort. Of note, the SCMA validity check shown in Chapter 6 requires exponential effort with a worst case of $O(2^n)$, and the DCSA from Chapter 4 requires cubic effort. Although both perform rapidly in practice, it may be possible to improve one or both algorithms to perform their functions more efficiently.

The computational study in this research was performed on the first 10 problems of each file provided by Chu and Beasley in [19] with a tightness ratio of 0.25. Other test problems exist in the same files with different tightness ratios. Future research should consider if varying the tightness ratios tends to motivate different levels of computational improvement when merged cover inequalities are added to the MK instance.

Another possibility for future computational studies is to truncate some variables by removing the smallest a values from each row of the original A matrix. It is less likely that these indices would contribute to a cover inequality in any given row, and truncating the A matrix would tend to speed up computations by generating a smaller matrix for algebraic computations. However, it is possible that indices with small a values may have some of the highest reduced costs. If this happens, truncating the A matrix could eliminate indices with the potential to serve as very strong terms in a host cover inequality. Some experimentation with this concept suggested that eliminating about 10% of the smallest coefficients facilitated faster results in some instances, but a more complete study of this concept could be considered as future research.

These ideas are excellent candidates for computational extensions and algorithm improvements for future research. Some of the largest, most difficult IP problems may provide some of the best candidates for excellent computational improvements through the use of inequality merging. These computational extensions may further reduce computational requirements for large integer programming problems.

Bibliography

[1] Achterberg, T., T. Koch, and A. Martin, "Branching rules revisited," *Operations Research Letters*, Vol. 33, No. 1, (2005), pp. 42-54.

[2] Achterberg, T., and T. Berthold, "Hybrid branching," *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Lecture Notes in Computer Science*, Vol. 5547 (2009), pp. 309-311.

[3] Ahuja, R. and C. Cunha, "Very large-scale neighborhood search for the k-constraint multiple knapsack problem," *Journal of Heuristics, Special Issue: Supply Chain and Distribution Management*, Vol. 11, (2005), pp. 465-481.

[4] Anstreicher, K., N. Brixius, J-P. Goux, and J. Linderoth, "Solving large quadratic assignment problems on computational grids," *Mathematical Programming*, Vol. 91, No. 3, (2002), pp. 563-588.

[5] Atamtürk, A. and V. Narayanan, "Lifting for conic mixed-integer programming," *Mathematical Programming*, Vol. 126, No. 2, (2011), pp. 351-363.

[6] Atamtürk, A., G. Nemhauser and M. Savelsbergh, "Conflict graphs in solving integer programming problems," *European Journal of Operational Research*, Vol. 121, No. 1, (2000), pp. 40-55.

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

[8] Balas, E., "Facets of the knapsack polytope," *Mathematical Programming*, Vol. 8, No. 1, (1975), pp. 146-164.

[9] Balas, E. and M. Perregaard, "A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer gomory cuts for 0-1 programming," *Mathematical Programming*, Vol. 94, Nos. 2-3, (2003), pp. 221-245.

[10] Balas, E and E. Zemel, "Facets of the knapsack polytope from minimal covers," *SIAM Journal of Applied Mathematics*, Vol. 34, No. 1, (1978), pp. 119-148.

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

[12] Balas, E. and S. Ng, "On the set covering polytope. II, Lifting the facets with coefficients in 0,1,2," *Mathematical Programming*, Vol. 45, Nos. 1-3, (1989), pp. 1-20.

[13] Bartlett, M., A. Frisch, Y. Hamadi, I. Miguel, S.A. Tarim, and C. Unsworth, "The temporal knapsack problem and its solution," *Integration of AI and OR Techniques in*

Constraint Programming for Combinatorial Optimization Problems, Conference Proceedings, CPAIOR 2005, R. Barták and M. Milano (eds), Spinger Lecture Notes in Computer Science, Vol. 3524, (2005), pp. 34-48.

[14] Benichou, M., J.M. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and O. Vincent. "Experiments in mixed-integer linear programming," *Mathematical Programming*, Vol. 1 (1971), pp. 76-94.

[15] Bonami, P., M. Kilinc, and J. Linderoth, "Algorithms and software for convex mixed integer nonlinear programs," *The IMA Volumes in Mathematics and its Applications*, Vol. 154, (2012), pp. 1-39.

[16] Chang, P. and J. Lee, "A fuzzy DEA and knapsack formulation integrated model for project selection," *Computers and Operations Research*, Vol. 39, No. 1, (2012), pp. 112-125.

[17] Cho, C., D. Padberg, and M. Rao, "On the uncapacitated plant location problem. II. Facets and lifting theorems," *Mathematics of Operations Research*, Vol. 8, No. 4, (1983), pp. 590-612.

[18] Chvátal, V., "Edmonds polytopes and a hierarchy of combinatorial problems," *Discrete Mathematics*, Vol. 4, No. 4, (1973), pp. 305-337.

[19] Chu, P.C. and J.E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, Vol. 4, (1998), pp. 63-86.

[20] Chu, Y. and Q. Xia, “A hybrid algorithm for a class of resource constrained scheduling problems,” *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Conference Proceedings, CPAIOR 2005, R. Barták and M. Milano (eds), Spinger Lecture Notes in Computer Science, Vol. 3524, (2005), pp. 110-124.

[21] IBM ILOG CPLEX Optimizer, Version 12.5.1, see http://www-01.ibm.com/software/info/ilog/. (2013).

[22] De Farias Jr, I., E. Johnson, and G. Nemhauser, “Facets of the complementarity knapsack polytope,” *Mathematics of Operations Research*, Vol. 27, No. 1, (2002), pp. 210-227.

[23] Dawande, M., J. Kalagnanam, P. Keskinocak, R. Ravi, and F.S. Salman, “Approximation algorithms for the multiple knapsack problem with assignment restrictions,” *Journal of Combinatorial Optimization*, Vol. 4, No. 2, (2000), pp. 171-186.

[24] Demirel, N.Ö. and H. Gökeen, “A mixed integer programming model for remanufacturing in reverse logistics environment,” *The International Journal of Advanced Manufacturing Technology*, Vol. 39, Nos. 11-12, (2008), pp. 1197-1206.

[25] Dey, S. and L. Wolsey, “Two row mixed-integer cuts via lifting,” *Mathematical Programming* Vol. 124, Nos. 1-2, (2010), pp. 143-174.

[26] Dey, S. and J-P. Richard, "Sequential-merge facets for two-dimensional group problems," *Proceedings 12th Conference on Integer Programming and Combinatorial Optimization*, Fischetti, M. and D.P. Williamson, eds., (2007), pp. 30-42.

[27] Dizdar, D., A. Gershkov, and B. Moldovanu, "Revenue maximization in the dynamic knapsack problem," *Theoretical Economics*, Vol. 6, No. 2, (2011), pp. 157-184.

[28] Dowsland, K.A. and J.M. Thompson, "Solving a nurse scheduling problem with knapsacks, networks and tabu search," *Journal of the Operational Research Society*, Vol. 51, No. 7, (2000), pp. 825-833.

[29] Dudzinski, K. and S. Walukiewicz, "Exact methods for the knapsack problem and its generalizations," *European Journal of Operational Research*, Vol. 28, No. 1, (1987), pp. 3-21.

[30] Easton, K., G.L. Nemhauser, and M.A. Trick, "Solving the traveling tournament problem: a combined integer and constraint programming approach," *PATAT'2002*, E. Burke and P. Causmaecker (eds), Springer Lecture Notes in Computer Science, Vol. 2740, (2003), pp. 63-77.

[31] Easton, T. and K. Hooker, "Simultaneously lifting sets of binary variables into cover inequalities for knapsack polytopes," *Discrete Optimization*, Special Issue: In Memory of George B. Dantzig, Vol. 5, No. 2, (2008), pp. 254-261.

[32] Easton, T., K. Hooker and E. K. Lee, "Facets of the independent set polytope," *Mathematical Programming, Series B*, Vol. 98, Nos. 1-3, (2003), pp. 177-199.

[33] Fischetti, M., A. Lodi, M. Monaci, D. Salvagnin, and A. Tramontani, “Tree search stabilization by random sampling,” *Technical Report OR/13/5*, DEI, University of Bologna (2013).

[34] Gauthier, J.M. and G. Ribiere. “Experiments in mixed-integer linear programming using pseudo-costs,” *Mathematical Programming*, Vol. 12, (1977), pp. 26-47.

[35] Gomory, R., “Outline of an algorithm for integer solutions to linear programs,” *Bulletin of the American Mathematical Society*, Vol. 64, No. 5, (1958), pp. 275-278.

[36] Gomory, R., “Some polyhedra related to combinatorial problems,” *Linear Algebra and its Applications*, Vol. 2, No. 4, (1969), pp. 451-558.

[37] Gomory, R. and E.L. Johnson, “Some continuous functions related to corner polyhedra 1,” *Mathematical Programming*, Vol. 3, No. 1, (1972), pp. 23-85.

[38] Gu, Z., G. Nemhauser, and M. Savelsbergh, “Lifted cover inequalities for 0-1 integer programs: computation,” *INFORMS Journal on Computing*, Vol. 10, No. 4, (1998), pp. 427-437.

[39] Gu, Z., G. Nemhauser, and M. Savelsbergh, “Lifted cover inequalities for 0-1 integer programs: complexity,” *INFORMS Journal on Computing*, Vol. 11, No. 1, (1999), pp. 117-123.

[40] Gu, Z., G. Nemhauser, and M. Savelsbergh, “Sequence independent lifting in mixed integer programming,” *Journal of Combinatorial Optimization*, Vol. 4, No. 1, (2000), pp. 109-129.

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

[42] Hammer, P.L., E.L. Johnson, and U.N. Peled, "Facets of regular 0-1 polytopes," *Mathematical Programming*, Vol. 8, No. 1, (1975), pp. 179-206.

[43] Hane, C., C. Barnhart, E. Johnson, R. Marsten, G. Nemhauser, and G. Sigismondi, "The fleet assignment problem: solving a large-scale integer program," *Mathematical Programming*, Vol. 70, Nos. 1-3, (1995), pp. 211-232.

[44] Hansen, J. and T. Lidén, "Group construction for airline cabin crew: comparing constraint programming with branch and price," *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Conference Proceedings, CPAIOR 2005, R. Barták and M. Milano (eds), Spinger Lecture Notes in Computer Science, Vol. 3524, (2005), pp. 228-242.

[45] Hemmecke, R., S. Onn, and R. Weismantel, "A polynomial oracle-time algorithm for convex integer minimization," *Mathematical Programming*, Vol. 126, No. 1, (2011), pp. 97-117.

[46] Hickman, R. and T. Easton, "Merging valid inequalities over the multiple knapsack polyhedron," To Appear In: *International Journal of Operations Research.*

[47] Hochbaum, D. "A non-linear knapsack problem," *Operations Research Letters*, Vol. 17, No. 3, (1995), pp. 103-110.

[48] Hooker, J.N., “A search-infer-and-relax framework for integrating solution methods,” *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Conference Proceedings, CPAIOR 2005, R. Barták and M. Milano (eds), Spinger Lecture Notes in Computer Science, Vol. 3524, (2005), pp. 243-257.

[49] Hooker, K. and T. Easton, “Using hyperstars to create facial defining inequalities of general binary integer programs,” *The International Journal of Operations Research*, Vol. 4, No. 3, (2007), pp. 138-145.

[50] Hunsaker, B. and C. Tovey, “Simple lifted cover inequalities and hard knapsack problems,” *Technical Report: Industrial Engineering, University of Pittsburgh*, Pittsburgh, PA, (2004), pp. 1-13.

[51] INFORMS Analytics webpage, see https://www.informs.org/Sites/Getting-Started-With-Analytics/What-Analytics-Is (2014).

[52] Ju, L., B.K. Huynh, S. Chakraborty, and A. Roychoudhury, “Context-sensitive timing analysis of esterel programs,” *Design Automation Conference, Conference Publication*, 46th ACM/IEEE, (2009), pp. 870-873.

[53] Kalagnanam, J., M. Dawande, M. Trumbo, and H.S. Lee, “The surplus inventory matching problem in the process industry,” *Operations Research*, Vol. 48, No. 4, (2000), pp. 505-516.

[54] Kellerer, H. and V.A. Strusevich, "Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications," *Algorithmica*, Vol. 57, No. 4, (2010), pp. 769-795.

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

[56] Land, A. H., and A. G. Doig, "An automatic method of solving discrete programming problems," *Econometrica*, Vol. 28, No. 3, (1960), pp. 497-520.

[57] Luedtke, J., S. Ahmed, and G. Nemhauser, "An integer programming approach for linear programs with probabilistic constraints," *Mathematical Programming*, Vol. 122, No. 2, (2010), pp. 247-272.

[58] Martello, S. and P. Toth, "Algorithms for knapsack problems," *Annals of Discrete Mathematics*, Vol. 31, (1987), pp. 213-258.

[59] Melachrinoudis, E. and G. Kozanidis, "A mixed integer knapsack model for allocating funds to highway safety improvements," *Transportation Research Part A: Policy and Practice*, Vol. 36, No. 9, (2002) pp. 789-803.

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

[61] Nemhauser, G. and P. Vance, "Lifted cover facets of the 0-1 knapsack polytope with GUB constraints," *Operations Research Letters*, Vol. 16, No. 5, (1994), pp. 255-263.

[62] OR Library webpage, see http://people.brunel.ac.uk/ mastjjb/jeb/info.html (2013).

[63] Park, K., "Lifting cover inequalities for the precedence-constrained knapsack problem," *Discrete Applied Mathematics*, Vol. 72, No. 3, (1997), pp. 219-241.

[64] Pajunas, A., E. Matto, M. Trick, and L. Zuluaga, "Optimizing highway transportation at the United States postal service," *Interfaces*, Vol. 37, No. 6, (2007), pp. 515-525.

[65] Refalo, P., "Impact-based search strategies for constraint programming," *Principles and Practice of Constraint Programming – CP 2004, Lecture Notes in Computer Science*, Vol. 3258, (2004) pp. 557-571.

[66] Salman, F.S., J. Kalagnanam, S. Murthy, and A. Davenport, "Cooperative strategies for solving the bicriteria sparse multiple knapsack problem," *Journal of Heuristics*, Vol. 8, No. 2, (2002), pp. 215-239.

[67] Seda, M., "Resource-constrained project scheduling problem as a sequence of multiple-knapsack problems," *Proceedings of the 10th WSEAS International Conference on Computers*, (2006), pp. 80-86.

[68] Shachnai, H., and T. Tamir, "On two class-constrained versions of the multiple knapsack problem," *Algorithmica*, Vol. 29, No. 3, (2001), pp. 442-467.

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

[70] Szeto, K.Y. and M.H. Lo, "An application of adaptive genetic algorithm in financial knapsack problem," *Innovations in Applied Artificial Intelligence, Lecture Notes in Computer Science*, Springer: Berlin/Heidelberg, (2004), pp. 1220-1228.

[71] Trick, M. "Formulations and reformulations in integer programming," *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Conference Proceedings, CPAIOR 2005, R. Barták and M. Milano (eds), Spinger Lecture Notes in Computer Science, Vol. 3524, (2005), pp. 366-379.

[72] Weismantel, R., "On the 0/1 knapsack polytope," *Mathematical Programming*, Vol. 77, No. 3, (1997), pp. 49-68.

[73] Williams, H.P., *Model Building in Mathematical Programming*,Wiley, New York, (1999).

[74] Wolsey, L., "Faces for a linear inequality in 0-1 variables," *Mathematical Programming*, Vol. 8, No. 1, (1975), pp. 165-178.

[75] Wolsey, L., "Facets and strong valid inequalities for integer programs," *Operations Research*, Vol. 24, No. 2, (1976), pp. 367-372.

[76] Wolsey, L., "Valid inequalities and superadditivity of 0/1 integer programs," *Mathematics of Operations Research*, Vol. 2, No. 1, (1977), pp. 66-77.

[77] Zemel, E., "Lifting the facets of 0-1 polytopes" *Mathematical Programming*, Vol. 15, No. 1, (1978), pp. 268-277.

[78] Zemel, E., “Easily computable facets of the knapsack polytope,” *Mathematics of Operations Research*, Vol. 14, No. 4, (1989), pp. 760-764.