Open Journal of Optimization, 2015, 4, 141-155
Published Online December 2015 in SciRes. http://www.scirp.org/journal/ojop
http://dx.doi.org/10.4236/ojop.2015.44014

On Merging Cover Inequalities for Multiple Knapsack Problems

Randal Hickman[1], Todd Easton[2]

[1]Department of Mathematical Sciences, United States Military Academy, West Point, USA
[2]Industrial and Manufacturing Systems Engineering Department, Kansas State University, Manhattan, USA
Email: Randal.hickman@usma.edu, teaston@ksu.edu

Received 27 August 2015; accepted 21 December 2015; published 25 December 2015

Abstract

This paper describes methods to merge two cover inequalities and also simultaneously merge multiple cover inequalities in a multiple knapsack instance. Theoretical results provide conditions under which merged cover inequalities are valid. Polynomial time algorithms are created to find merged cover inequalities. A computational study demonstrates that merged inequalities improve the solution times for benchmark multiple knapsack instances by about 9% on average over CPLEX with default settings.

Keywords

Multiple Knapsack Problem, Cutting Plane, Cover Inequality, Inequality Merging, Pseudocost, Integer Programming

1. Introduction to Inequality Merging

An integer program (IP) is a common type of optimization problem, defined as maximize $c^{\mathrm{T}}x$ subject to $Ax \le b$ and $x \in \mathbb{Z}_+^n$ where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$ where m and n are integers both greater than or equal to 1. Define $N = \{1, \cdots, n\}$ as the set of indices of an IP.

One frequently studied IP is the 0 - 1 knapsack problem (KP), defined as maximize $\sum_{i=1}^{n} c_i x_i$ subject to $\sum_{i=1}^{n} a_i x_i \le b$, and $x_i \in \{0,1\}^n$ where c and $a \in \mathbb{R}_+^n$, $b \in \mathbb{R}_+$. The multiple knapsack (MK) problem has multiple knapsack constraints and is defined as maximize $c^{\mathrm{T}}x$ subject to $Ax \le b$ and $x \in \mathbb{Z}_+^n$ where $A \in \mathbb{R}_+^{m\times n}$, $b \in \mathbb{R}_+^m$, and $c \in \mathbb{R}^n$.

Solutions to KP and MK problems support a wide variety of real-world applications, including examples in Ahuja and Cunha [1], Chang and Lee [2], Dawande *et al.* [3], Dizdar *et al.* [4], Kellerer and Strusevich [5],

How to cite this paper: Hickman, R. and Easton, T. (2015) On Merging Cover Inequalities for Multiple Knapsack Problems. *Open Journal of Optimization*, **4**, 141-155. http://dx.doi.org/10.4236/ojop.2015.44014

Martello and Toth [6], Shachnai and Tamir [7], and Szeto and Lo [8]. This paper focuses on MK problems.

A half space is $\left\{x \in \mathbb{R}^n : \sum_{i=1}^n a_i x_i \le b\right\}$, 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*.

Let *P* be the set of feasible points of an integer program, where $P = \left\{x \in \mathbb{Z}_+^n : Ax \le b\right\}$. Define $P^{KP} = \left\{x \in \{0,1\}^n : \sum_{i=1}^n a_i x_i \le b\right\}$ and $P^{MK} = \left\{x \in \{0,1\}^n : Ax \le b\right\}$ as the feasible regions of the knapsack and multiple knapsack problems, respectively where $a \in \mathbb{R}_+^n$ and $A \in \mathbb{R}_+^{m \times n}$.

A well-known technique to improve solution times for IP problems is the generation of valid inequalities. An inequality $\sum_{i=1}^n \alpha_i x_i \le \beta$ is a valid inequality for $conv\left(P^{MK}\right)$ if every $x \in P^{MK}$ satisfies the inequality. If the valid inequality separates the linear relaxation solution from the convex hull of the IP, then it is called a cutting plane. The linear relaxation is the IP with the integrality restriction eliminated. The theoretically best cutting planes define facets of $conv\left(P^{MK}\right)$, but any cutting plane that separates the linear relaxation from $conv\left(P^{MK}\right)$ may be computationally useful. A thorough explanation of such results is in Nemhauser and Wolsey [9].

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 $i \in \{1,\cdots,m\}$ if $\sum_{j \in C} a_{i,j} > b_i$. The corresponding cover inequality is valid for $conv\left(P^{MK}\right)$ and takes the form $\sum_{j \in C} x_j \le |C| - 1$. Cover cuts have been studied extensively by Balas and Zemel [10], De Farias *et al*. [11], Louveaux and Weismantel [12], Nemhauser and Vance [13], and Park [14]. Knowledge of cover cuts is critical to this research.

Many such covers may exist and pseudo-costing strategies provide a prioritized variable ordering. Pseudo-costing strategies for integer programming problems were studied by Benichou, *et al*. in [15] and Gauthier and Ribiere in [16]. Refalo used pseudo-cost strategies to improve constraint programming in [17], and Achterberg, *et al*. developed reliability branching rules for IPs as an extension of pseudo-costing in [18].

In some instances, cover inequalities may be strengthened through lifting. Gomory introduced the technique in [19], taking a valid inequality of a restricted space and tilting it to become a valid inequality of a higher dimensional space. Substantial bodies of research have extended lifting to several categories such as exact up-lifting (Cho *et al*. [20], Gutierrez [21], Hammer *et al*. [22], and Wolsey [23]), exact simultaneous up-lifting (Easton and Hooker [24], Kubik [25], and Zemel [26]), exact sequential down and middle lifting by Wolsey [23], sequence dependent lifting (Atamtürk [27], Gu *et al*. [28]-[30], and Shebalov and Klabjan [31]), and other approximate lifting methods (Balas [32] and Weismantel [33]).

Theoretical foundations for inequality merging were first introduced by Hickman and Easton in [34]. Although merging appears similar to lifting, it yields new cutting planes that are not attainable through straightforward applications of known lifting techniques. Their paper creates a single cutting plane by merging two inequalities. This merged inequality can be theoretically stronger than the original inequalities, and it may induce a facet under certain conditions.

This paper extends the idea of inequality merging by focusing on cover inequalities in MK problems. Information from two or more cover inequalities in an MK instance may be merged into a single cutting plane. In some instances, simultaneous merging of cover inequalities may occur across multiple rows at the same time.

The next section describes the process of cover inequality merging for MK instances and provides theoretical results and examples. The third section offers the results of a computational study that highlights the computational benefits of employing merged cover inequalities in test MK problems. The final section offers some directions for future research.

2. Theory and Examples of Merging Cover Inequalities

It is straightforward to find cover inequalities in MK instances and merging requires two covers, called host and donor. Let $C^{host} \subseteq N$ be a cover in row *r* and $C^{donor} \subseteq N$ be a cover in row *s* for some $r, s \in \{1,\cdots,m\}$. Thus, the cover inequalities $\sum_{i \in C^{host}} x_i \le |C^{host}| - 1$ and $\sum_{i \in C^{donor}} x_i \le |C^{donor}| - 1$ are valid inequalities of $conv\left(P^{MK}\right)$.

Merging the host and donor cover inequalities occurs on binary variable x_p where $\{p\} = C^{host} \bigcap C^{donor}$ or if

$C^{host} \cap C^{donor} = \varnothing$, then $p \in C^{host}$. Since x_p is bounded by 1 and $\sum_{i \in C^{donor}} \frac{1}{\left|C^{donor}\right|-1} x_i \le 1$, it follows that x_p could be replaced in host cover inequality with the C^{donor} indices with coefficients $\frac{1}{\left|C^{donor}\right|-1}$. Thus, a merged cover inequality has the form $\sum_{i \in C^{host} \setminus \{p\}} x_i + \sum_{i \in C^{donor}} \frac{1}{\left|C^{donor}\right|-1} x_i \le \left|C^{host}\right| - 1$.

If the merged inequality is valid, then this inequality includes more nonzero coefficients than either C^{host} or C^{donor}. The question remains as to whether or not the merged inequality is valid. The following theorem provides conditions for its validity.

Theorem 1. *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 $\left|C^{host} \cap C^{donor}\right| \le 1$. Define index $p \in C^{host}$ as the merging index with the restriction that if $\left|C^{host} \cap C^{donor}\right| = 1$, 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, $\sum_{i \in C^{host} \setminus \{p\}} x_i + \sum_{i \in C^{donor}} \frac{1}{\left|C^{donor}\right|-1} x_i \le \left|C^{host}\right| - 1$, is valid for $conv\left(P^{MK}\right)$.*

Proof. Let x' be any point in P_{MK}. Define $q = \sum_{i \in C^{host} \setminus \{p\}} x_i'$. If $q = \left|C^{host}\right| - 1$, 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, $\sum_{i \in C^{host} \setminus \{p\}} x_i' + \sum_{i \in C^{donor}} \frac{1}{\left|C^{donor}\right|-1} x_i' \le \left|C^{host}\right| - 1$. If $q \le \left|C^{host}\right| - 2$, then $\sum_{i \in C^{donor}} \frac{1}{\left|C^{donor}\right|-1} x_i' \le 1$ since C^{donor} is a cover. Thus, $\sum_{i \in C^{host} \setminus \{p\}} x_i' + \sum_{i \in C^{donor}} \frac{1}{\left|C^{donor}\right|-1} x_i' \le \left|C^{host}\right| - 1$ and the result follows. □

Theorem 1 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 $\psi_p = b_r - \left(\sum_{i \in C^{host}} a_{r,i} - a_{r,p}\right) + 1$. The purpose of ψ_p is to rapidly identify indices that can be used to create a donor cover from any row *s*. Define these potential donor indices as $N_{\psi_p} = \left\{i \in N : i \notin C^{host} \setminus \{p\}, a_{r,i} \ge \psi_p\right\}$. 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 shown in the following theorem.

Theorem 2. *Given 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 $a_{r,i} \ge \psi_p$ for all $i \in C^{donor}$, then $\sum_{i \in C^{host} \setminus \{p\}} x_i + \frac{1}{\left|C^{donor}\right|-1} \sum_{i \in C^{donor}} x_i \le \left|C^{host}\right| - 1$ is a valid inequality of $conv\left(P^{MK}\right)$.*

Proof. Assume $x' \in P_{MK}$, C^{host} is a cover in row *r*, C^{donor} is a cover in some row *s* and $C^{donor} \subseteq N_{\psi_p}$. Define $q = \sum_{i \in C^{host} \setminus \{p\}} x_i'$. Since C^{host} is a cover, $q \le \left|C^{host}\right| - 1$. The proof divides into two cases, $q = \left|C^{host}\right| - 1$ and $q \le \left|C^{host}\right| - 2$.

First, assume $q = \left|C^{host}\right| - 1$. Thus, $x_i = 1$ for all $i \in C^{host} \setminus \{p\}$. Since $x' \in P_{MK}$, $\sum_{i \in C^{host} \setminus \{p\}} a_{r,i} + \sum_{i \notin C^{host} \setminus \{p\}} a_{r,i} x_i' \le b_i$. Thus, $\sum_{i \notin C^{host} \setminus \{p\}} a_{r,i} x_i' \le b_i - \sum_{i \in C^{host} \setminus \{p\}} a_{r,i} = \psi_p - 1$. Every $i \in C^{donor}$ has the property that $a_{r,i} \ge \psi_p$ and so $x_i' = 0$ for all $i \in C^{donor}$. Consequently, $\sum_{i \in C^{host} \setminus \{p\}} x_i' + \frac{1}{\left|C^{donor}\right|-1} \sum_{i \in C^{donor}} x_i' \le \left|C^{host}\right| - 1$.

Second, assume $q \le \left|C^{host}\right| - 2$. Since C^{donor} is a cover in row *s*, $\sum_{i \in C^{donor}} x_i' \le \left|C^{donor}\right| - 1$. Thus, $\frac{\sum_{i \in C^{donor}} x_i'}{\left|C^{donor}\right|-1} \le 1$.

Consequently, $\sum_{i\in C^{host}\setminus\{p\}} x_i' + \frac{1}{\left|C^{donor}\right|-1}\sum_{i\in C^{donor}} x_i' \le \left|C^{host}\right|-1$.

These two cases are exhaustive. Therefore every $x' \in P_{MK}$ satisfies $\sum_{i\in C^{host}\setminus\{p\}} x_i' + \frac{1}{\left|C^{donor}\right|-1}\sum_{i\in C^{donor}} x_i' \le \left|C^{host}\right|-1$ and this merged inequality is valid for $conv(P_{MK})$. □

To identify valid merged cover inequalities, the user must identify a host cover, C^{host} and a merged index $p \in C^{host}$. Some selections for C^{host} and a merged variable x_p may not allow a candidate donor inequality to exist. The Reducing ψ_p Algorithm changes C^{host} to increase the likelihood of the existence of an appropriate donor cover.

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 and a new merging variable. These are denoted by C'^{host} and $x_{p'}$, respectively.

Reducing ψ_p Algorithm

a. Initialization:

$\psi_p \leftarrow b_r - \left(\sum_{i\in C^{host}} a_{r,i} - a_{r,p}\right) + 1$

$C'^{host} \leftarrow C^{host} \setminus \{p\}$

$a_{total} \leftarrow \sum_{i\in C'^{host}} a_{r,i}$

b. Main Step:

For each $q \in N \setminus C'^{host}$

If $\tau\psi_p \le a_{r,q} \le \psi_p - 1$, 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 ψ_p

If the Reducing ψ_p Algorithm terminates successfully, then C'^{host} is a cover because 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 overlapped variable $x_{p'}$, and $\psi_{p'} = b_r - \left(\sum_{i\in C'^{host}} a_{r,i} - a_{r,p'}\right) + 1$. 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 a merged cutting plane inequality.

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 may be used iteratively 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 indices with small *a* coefficients to enter C'^{host}. When this happens, the size of C'^{host} may become undesirably large or fail to generate a cover. Including too many variables in the host cover results in fewer candidate indices in N_{ψ_p}.

High values of τ may allow few (or zero) new candidate indices for inclusion 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. Given

this sensitivity to τ, a careful selection of τ is required. 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 $O\left(\left|C^{host}\right|\right)$. The main step could search through all other indices, so it performs in $O\left(\left|N \setminus C^{host}\right|\right)$ effort. Thus, the algorithm runs in $O\left(\left|N\right|\right)$, which is linear for a fixed τ.

2.1. Merging over Multiple Donor Covers 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 from 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 $\sum_{i \in C^{host} \setminus \{p\}} x_i + \sum_{j \in N_{\psi_p}} \alpha_j x_j \le \left|C^{host}\right| - 1$ is likely to be valid for any $\alpha_j \le \dfrac{1}{\left|C_j\right| - 1}$ 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 $\alpha_j = \dfrac{1}{\left|C_j\right| - 1}$ 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 $c^q_{\min} = \min_{Q \subset C^{host} \setminus \{p\}, |Q| = q} \left\{\sum_{i \in Q} a_{r,i}\right\}$, $\alpha^q_{\min} = \min_{D \subseteq N_{\psi_p}} \left\{\left|D\right| : \sum_{i \in D} \alpha_i > \left|C^{host}\right| - q - 1\right\}$ and $a^q_{\min} = \min_{D' \subseteq N_{\psi_p}, |D'| = \alpha^q_{\min}} \left\{\sum_{i \in D'} a_{r,i}\right\}$.

Theorem 3. *Let C^{host} be a cover from a MK constraint with $p \in C^{host}$, corresponding value ψ_p and associated set N_{ψ_p}. Then the inequality $\sum_{i \in C^{host} \setminus \{p\}} x_i + \sum_{j \in N_{\psi_p}} \alpha_j x_j \le \left|C^{host}\right| - 1$ is valid for $conv\left(P^{MK}\right)$ for any $\alpha_j \le \dfrac{1}{\left|C_j\right| - 1}$ 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*

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

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

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

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

Assume 2) is true. Let $q = \sum_{i \in C^{host} \setminus \{p\}} x'_i$. Thus $\sum_{i \in C^{host} \setminus \{p\}} a_i x'_i \ge a^q_{\min}$. By 2) $a^q_{\min} + c^q_{\min} > b$ and thus $\sum_{j \in N_{\psi_p}} \alpha_j x'_j < \alpha^q_{\min} \le q$. Thus, $\sum_{i \in C^{host} \setminus \{p\}} x'_i + \sum_{j \in N_{\psi_p}} \alpha_j x'_j \le \left|C^{host}\right| - 1$. □

An immediate result of Theorem 3 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 stronger coefficients for each merging variable. The input to the Donor Coefficient Strengthening Algorithm (DCSA) is a MK instance, a host cover C^{host} from row *r* and an index $p \in C^{host}$.

Donor Coeffcient Strengthening Algorithm

a. Initialization:

$\psi_p \leftarrow b_r - \left(\sum_{i \in C^{host}} a_{r,i} - a_{r,p}\right) + 1$

$N_{\psi_p} \leftarrow \left\{ i \in N \setminus \left(C^{host} \setminus \{p\} \right) : a_{r,i} \geq \psi_p \right\}$

Let $A'_{r'}$ be N_{ψ_p} sorted according to the a coefficients in row $r' \in \{1, \cdots, m\}$

b. Main Step

For each $i \in N_{\psi_p}$

Set $\alpha_i \leftarrow 0$

For each $r' \in \{1, \cdots, m\}$

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

$coversize \leftarrow 1$

$j \leftarrow 1$

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

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 $\alpha_i < \dfrac{1}{coversize - 1}$, Then

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

end *for*

end *for*

c. Output

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

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 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 3, then it is a valid inequality.

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

2.2. Inequality Merging Example

The following example demonstrates the theoretical concepts discussed earlier. 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, $r=1$ and let the host cover be $C^{host}=\{1,2,3,4,5\}$. If the merging index is $p=5$, then $\psi_5 = b-(a_{1,1}+a_{1,2}+a_{1,3}+a_{1,4})+1=(79-70)+1=10$. 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 $N_{\psi_5}=\{5,6,7,8,9\}$.

No subset of N_{ψ_5} is a cover. The Reducing ψ_p Algorithm is used to change the host cover to create a smaller ψ. 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 $C'^{host}=\{1,2,3,4,11,12\}$. 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 a host and donor cover that can be merged. Merging the host with the donor on x_{12} yields (1), a valid inequality according to Theorem 2.

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

The following arguments demonstrate Theorems 1 and 2 in practice. Verifying the validity of (1) requires that $C^{host}\setminus\{12\}\cup\{i\}$ is a cover for some constraint for every $i\in C^{donor}$. The sum of the $C^{host}\setminus\{12\}$ $a_{1,i}$ coefficients is 76. Clearly $C^{host}\setminus\{12\}\cup\{i\}$ is a cover in the first knapsack as long as $a_{1,i}\ge 4$ and $i\notin C^{host}\setminus\{12\}$. Since all candidate donor indices have $a_{1,i}\ge 4=\psi_{12}$, (1) 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 $\{5,6,7,9,10,12\}$ and $\{5,6,7,8,9,10,13\}$. Accordingly, we could merge each of these cover inequalities with the host cover inequality yielding the following valid merged inequalities

$$x_1+x_2+x_3+x_4+\frac{1}{5}x_5+\frac{1}{5}x_6+\frac{1}{5}x_7+\frac{1}{5}x_9+\frac{1}{5}x_{10}+x_{11}+\frac{1}{5}x_{12}\le 5 \tag{2}$$

$$x_1+x_2+x_3+x_4+\frac{1}{6}x_5+\frac{1}{6}x_6+\frac{1}{6}x_7+\frac{1}{6}x_8+\frac{1}{6}x_9+\frac{1}{6}x_{10}+x_{11}+\frac{1}{6}x_{13}\le 5. \tag{3}$$

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 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, 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

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

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 $\{4,5,9,10,11,12,14\}$. One such cover in the second constraint is $\{4,9,10,11,12,14\}$, which results in the following merged constraint

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

The authors believe that 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 $N_{\psi_{12}}=\{5,6,7,8,9,10,12,13\}$. For each index in $N_{\psi_{12}}$, 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 1**. The smallest covers are listed in the order in which DCSA adds indices to the cover.

Thus the simultaneous merged inequality is

$$x_1+x_2+x_3+x_4+\frac{1}{4}x_5+\frac{1}{4}x_6+\frac{1}{4}x_7+\frac{1}{4}x_8+\frac{1}{4}x_9+\frac{1}{4}x_{10}+x_{11}+\frac{1}{4}x_{12}+\frac{1}{4}x_{13}\le 5. \quad (6)$$

Observe that this new inequality dominates all of the previous inequalities. Furthermore, to achieve this inequality all rows are necessary. For instance, the smallest cover in row 3 containing index 6 has 6 indices and thus row two is necessary. Similarly, the smallest cover in row 2 containing index 7 has 6 indices and thus row 3 is necessary.

Table 1. Applying DCSA to find strongest coefficients.

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	$\frac{1}{4}$

To argue validity of (6), consider Theorem 3. Since $\sum_{i\in N_{\psi 12}} \alpha_i = 2$, 1) is not satisfied. For 2), observe that $c^1_{\min} = a_{1,11} = 6$ and $c^2_{\min} = a_{1,11} + a_{1,4} = 22$. Continuing this process yields $c^3_{\min} = 38$, $c^4_{\min} = 56$, and $c^5_{\min} = 76$. Determining the values for $\alpha_{\min}$ yield that $\alpha^1_{\min}$, $\alpha^2_{\min}$, and $\alpha^3_{\min}$ do not exist as $\sum_{i\in N_{\psi 12}} \alpha_j = 2$. 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 $1 = \left|C^{host}\right| - 4 - 1$. Since $\left|C^{host}\right| - 5 - 1 = 0$, $\alpha^5_{\min} = 1$.

Since $\alpha^1_{\min}$, $\alpha^2_{\min}$, and $\alpha^3_{\min}$ 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 2) of Theorem 3 checks $c^4_{\min} + a^4_{\min} = 56 + 38 = 94 > 79$ and $c^5_{\min} + a^5_{\min} = 76 + 5 = 81 > 79$. Thus, (6) meets condition 2) of Theorem 3 and it is valid. As a note, observe that checking $q = \left|C^{host}\right| - 1$ is always true by the definition of N_{ψ_p}.

The final benefit of this example demonstrates that merging cover inequalities are not an immediate extension of known methods. There are similarities between inequality merging and some categories of lifting. Any type of sequential lifting has integer coefficients [35], and sequence independent lifting would require all non-cover coefficients in this example to be 0 [30]. Thus neither of these methods generate (6). While simultaneous lifting could theoretically generate (6) [21], it would require starting with the trivial cutting plane $x_1 + x_2 + x_3 + x_4 + x_{11} \le 5$ and furthermore have a perfect guess of proper weights. Consequently, inequality merging yields inequalities similar to (6), which are extremely unlikely to be produced by lifting techniques.

The general inequality merging presented by Hickman and Easton in [34] did not merge multiple donor covers simultaneously, and it could not obtain (6). Inequality merging is also fundamentally different from other popular cutting plane generation techniques such as C-G cuts (Chvátal [36] and Gomory [37]), disjunctive cuts (Balas and Perregaard [38]), Gomory cuts (Gomory [37]), or superadditive cuts (Gomory and Johnson [39] and Wolsey [40]). Theoretically, these methods could generate (6), 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.

A single call to DCSA creates (6) and requires $O\left(nm^2\right)$ 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. The next section provides the results of a computational study, demonstrating the practical effectiveness of inequality merging on benchmark multiple knapsack problems.

3. Computational Study

This computational study compares solution times for multiple knapsack problems both with and without the use of merged inequalities. The instances chosen for this study are the MK instances from the *OR-Library* [41], developed by Chu and Beasley in 1998 [42]. The majority of these instances are either trivially solved or too computationally intensive for an optimal solution. Thus, this study focuses on medium sized instances contained in files mknapcb2 ($m = 5$ and $n = 250$) and mknapcb5 ($m = 10$ and $n = 250$).

Each file contains 30 instances divided into groups of 10 based upon a tightness ratio, which is equal to $s_i = \dfrac{b_i}{\sum_{j=1}^{n} a_{i,j}}$. The tightness ratio is approximately equal for all constraints and is 0.25 for the first 10 instances, 0.5 for the second ten instances, and 0.75 for the final ten instances. For this computational study, the first ten instances are only considered. When the tightness ratio is 0.5 or higher, C^{host} tends to include too many variables. Since the variables in C^{host} are prohibited from being in N_{ψ_p}, higher tightness ratios reduce the size of N_{ψ_p}, which decreases the likelihood of finding a suitable donor cover in any row.

The study considers a variety of implementation strategies including the number of merged inequalities added, the possibility of overlapping rows when multiple cuts are added, the option to use the Donor Coefficient Strengthening Algorithm when constructing merged inequalities, and different pseudocosting techniques. The psuedocosting techniques provide an order for selecting indices for cover inequalities. Three options are considered: sorting on the reduced costs, sorting on the *a* coefficient values, and sorting on equal weights for both reduced costs and *a* coefficient values. More details of these methods and computational results are described in [43].

The experimentation compares computational effort to solve the MK instances with and without the inclusion

of merged cover inequalities. CPLEX 12.5 [44] solves all of the instances at default settings, but writing node files out to memory is used for the larger instances. All results are obtained using a PC with an i7-4770 processor at 3.4 GHz with 8 GB of RAM.

3.1. Computational Results

The computational study considered the variations of each implementation strategy by testing both small and large instances. Solving all 10 smaller instances required from 10 to 15 minutes. Solving all 10 larger instances typically needed 1 to 2 days. Instead of reporting the time in seconds, the data below compares computational ticks in CPLEX, as this is more accurate. It should be noted that the time in seconds was highly correlated to ticks. The overall improvement in time was plus or minus two percent of the percent improvement in ticks.

Ticks provide a more accurate comparison between the experimental runs because the computational time in seconds is subject to variability on different computers. Fischetti, *et al.* argue the benefit of using ticks in [45]. Ju, *et al*. use a similar process to report their computational results [46]. Since the two categories of MK test problems 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.

3.1.1. Computation Results for Smaller Problems

Problems from the smaller MK instances (file mknapcb2) offered an excellent opportunity for extensive experimentation with each of the implementation strategies. **Table 2** and **Table 3** show the best known results

Table 2. Changing implementation strategies for smaller MK problems, 1 - 3 Cuts.

# Merged	Overlap	Pseudo-Costing Strategy			Total Ticks	Percent
Cuts	Rows	Red. Costs	Balanced	*a* Values	(10 probs.)	Improv.
Baseline	Baseline	0	0	0	81,497	Baseline
1	N/A	1	0	0	**70,895**	13.0%
1	N/A	0	0	1	**69,669**	14.5%
1	N/A	0	1	0	75,868	6.9%
2	Yes	2	0	0	**72,376**	11.2%
2	Yes	0	0	2	78,840	3.3%
2	Yes	0	2	0	**71,668**	12.1%
2	Yes	1	0	1	**71,305**	12.5%
2	Yes	0	1	1	**67,634**	17.0%
2	Yes	1	1	0	76,272	6.4%
2	No	2	0	0	81,022	0.6%
2	No	0	0	2	76,956	5.6%
2	No	0	2	0	77,947	4.4%
2	No	1	0	1	**64,417**	21.0%
2	No	0	1	1	**72,356**	11.2%
2	No	1	1	0	79,088	3.0%
3	Yes	3	0	0	74,985	8.0%
3	Yes	0	0	3	**72,123**	11.5%
3	Yes	0	3	0	**67,794**	16.8%
3	Yes	1	1	1	**72,593**	10.9%
3	No	3	0	0	80,178	1.6%
3	No	0	0	3	75,445	7.4%
3	No	0	3	0	77,490	4.9%
3	No	1	1	1	77,448	5.0%
Merged Average					**74,099**	**9.1%**

Table 3. Changing implementation strategies for smaller MK problems, 4 - 5 Cuts.

# Merged	Overlap	Pseudo-Costing Strategy			Total Ticks	Percent
Cuts	Rows	Red. Costs	Balanced	*a* Values	(10 probs.)	Improv.
Baseline	Baseline	0	0	0	81,497	Baseline
4	Yes	4	0	0	80,230	1.6%
4	Yes	0	0	4	79,756	2.1%
4	Yes	0	4	0	80,494	1.2%
4	Yes	1	2	1	73,751	9.5%
4	No	4	0	0	80,606	1.1%
4	No	0	0	4	81,981	−0.6%
4	No	0	4	0	**72,744**	10.7%
4	No	1	2	1	74,820	8.2%
5	Yes	5	0	0	82,279	−1.0%
5	Yes	0	0	5	**72,882**	10.6%
5	Yes	0	5	0	82,423	−1.1%
5	Yes	1	3	1	76,820	5.7%
5	No	5	0	0	77,944	4.4%
5	No	0	0	5	78,817	3.3%
5	No	0	5	0	83,201	−2.1%
5	No	1	3	1	75,256	7.7%
Merged Average					**78,375**	**3.8%**

from these experiments on the smaller MK instances. Since there are 5 rows in the smaller test problems, each implementation strategy was tested with the inclusion of 1 - 5 merged inequalities. **Table 2** shows the results for iterations with 1, 2, or 3 merged inequalities added. **Table 3** shows the results with 4 or 5 merged inequalities added.

Observe that inequality merging outperformed the baseline CPLEX computational ticks for all strategies in **Table 2** with 1, 2, or 3 added inequalities, and inequality merging also outperformed the baseline CPLEX by about 9% on average. The 4 and 5 cut strategies from **Table 3** outperformed baseline CPLEX by about 4%. This demonstrates that adding more merged inequalities 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.

Table 4 aggregates results from **Table 2** and **Table 3**, and it reports the average results based upon different pseudo-costing strategies. Observe that many of the experimental runs in **Table 2** and **Table 3** 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 4**. 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 generated.

Merged inequalities almost always improved the computational time, regardless of the overlapping strategy. 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 3. 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 DCSA is employed, the algorithm may also search all candidate rows including those that had previously generated a host inequality. Thus, all future experimentation overlaps rows.

3.1.2. Computational Results 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 this section. Solving all 10 MK

instances required from 1 to 2 days to solve. **Table 5** shows the best known results for the large MK problems when the recommended implementation strategies are followed.

Table 5 shows that inequality merging continues to provide an average improvement of about 9% over the baseline CPLEX computational effort even on challenging instances. This is roughly the same level of average improvement observed in the smaller MK instances. Notice that 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.

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 4** 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 6 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 yields 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 DCSA and other instances that consider only specified randomly-selected donor inequalities that define single overlaps. The two best results include both overlapping strategies and DCSA facilitated the single best percentage improvement in problem 1. It is clear that each strategy yields strong results in specific instances, and neither overlapping strategy dominates the other.

Table 4. Average ticks of pseudo-costing strategies from **Table 2** and **Table 3**.

	Pseudo-Costing Strategy			
	All Reduced Costs	All Balanced	All *a* Values	Mixture of Strategies
Average Ticks	77,835	76,625	76,274	73,569
% Improvement	4.5%	6.0%	6.4%	9.8%

Table 5. Changing implementation strategies for larger MK problems, 1 - 3 Cuts.

# Merged	Pseudo-Costing Strategy			Total Ticks	Percent
Cuts Added	Red. Costs	Balanced	*a* Values	(10 problems)	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%
Merged Average				**28,260,350**	**8.8%**

Table 6. Best merging performance by problem for $m = 10$ and $n = 250$.

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%			

These larger problems are excellent representatives of difficult, real-world problems. Thus, the observed reductions in computational requirements validated the theoretical advancements in this research as effective methods to help decrease computational effort for modern MK problems.

4. Conclusion and Future Work

This paper provides the theoretical foundations needed to build merged cover inequalities in MK instances. The theorems generate conditions for validity, using the ψ_p term to identify candidate merging indices and simultaneously merging on all rows. Two algorithms support the newly-discovered theory, including an algorithm to reduce the size of ψ_p and a second algorithm to find the strongest coefficients for each candidate index during simultaneous merging.

The computational study validates inequality merging as an effective technique that reduces computational time for multiple knapsack problems. Preferred implementation strategies should generate 1, 2, or 3 cuts and overlap the rows. These 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 IPs.

Three ideas present themselves as excellent candidates for future research extensions. In this paper, inequality merging occurs on a single variable. The theory may be extended to merge on multiple variables. Since this paper focuses on cover inequalities and MK instances, another theoretical extension may merge other classes of cutting planes in general IPs.

All of the computational analysis in this research was performed on the first 10 problems of each file provided by Chu and Beasley [42] with a tightness ratio of 0.25. Other test problems exist in the same files with different tightness ratios, and future research should consider if varying tightness ratios tend to motivate different levels of computational improvement when merged cover inequalities are added to the MK instance.

References

[1] Ahuja, R. and Cunha, C. (2005) Very Large-Scale Neighborhood Search for the K-Constraint Multiple Knapsack Problem. *Journal of Heuristics, Special Issue*: *Supply Chain and Distribution Management*, **11**, 465-481. http://dx.doi.org/10.1007/s10732-005-2634-9

[2] Chang, P. and Lee, J. (2012) A Fuzzy DEA and Knapsack Formulation Integrated Model for Project Selection. *Computers and Operations Research*, **39**, 112-125. http://dx.doi.org/10.1016/j.cor.2010.10.021

[3] Dawande, M., Kalagnanam, J., Keskinocak, P., Ravi, R. and Salman, F.S. (2000) Approximation Algorithms for the Multiple Knapsack Problem with Assignment Restrictions. *Journal of Combinatorial Optimization*, **4**, 171-186.

http://dx.doi.org/10.1023/A:1009894503716

[4] Dizdar, D., Gershkov, A. and Moldovanu, B. (2011) Revenue Maximization in the Dynamic Knapsack Problem. *Theoretical Economics*, **6**, 157-184. http://dx.doi.org/10.3982/TE700

[5] Kellerer, H. and Strusevich, V.A. (2010) Fully Polynomial Approximation Schemes for a Symmetric Quadratic Knapsack Problem and its Scheduling Applications. *Algorithmica*, **57**, 769-795. http://dx.doi.org/10.1007/s00453-008-9248-1

[6] Martello, S. and Toth, P. (1987) Algorithms for Knapsack Problems. *Annals of Discrete Mathematics*, **31**, 213-257. http://dx.doi.org/10.1016/s0304-0208(08)73237-7

[7] Shachnai, H. and Tamir, T. (2001) On Two Class-Constrained Versions of the Multiple Knapsack Problem. *Algorithmica*, **29**, 442-467. http://dx.doi.org/10.1007/s004530010057

[8] Szeto, K.Y. and Lo, M.H. (2004) An Application of Adaptive Genetic Algorithm in Financial Knapsack Problem. In: *Innovations in Applied Artificial Intelligence*, *Lecture Notes in Computer Science*, Springer, Berlin/Heidelberg, 1220-1228. http://dx.doi.org/10.1007/978-3-540-24677-0_125

[9] Nemhauser, G. and Wolsey, L. (1988) Integer and Combinatorial Optimization. John Wiley and Sons, New York. http://dx.doi.org/10.1002/9781118627372

[10] Balas, E and Zemel, E. (1978) Facets of the Knapsack Polytope from Minimal Covers. *SIAM Journal of Applied Mathematics*, **34**, 119-148. http://dx.doi.org/10.1137/0134010

[11] De Farias Jr., I., Johnson, E. and Nemhauser, G. (2002) Facets of the Complementarity Knapsack Polytope. *Mathematics of Operations Research*, **27**, 210-227. http://dx.doi.org/10.1287/moor.27.1.210.335

[12] Louveaux, Q. and Weismantel, R. (2010) Polyhedral Properties for the Intersection of Two Knapsacks. *Mathematical Programming Series A*, **113**, 15-37. http://dx.doi.org/10.1007/s10107-006-0045-9

[13] Nemhauser, G. and Vance, P. (1994) Lifted Cover Facets of the 0-1 Knapsack Polytope with GUB Constraints. *Operations Research Letters*, **16**, 255-263. http://dx.doi.org/10.1016/0167-6377(94)90038-8

[14] Park, K. (1997) Lifting Cover Inequalities for the Precedence-Constrained Knapsack Problem. *Discrete Applied Mathematics*, **72**, 219-241. http://dx.doi.org/10.1016/0166-218X(95)00113-6

[15] Benichou, M., Gauthier, J.M., Girodet, P., Hentges, G., Ribiere, G. and Vincent. O. (1971) Experiments in Mixed-Integer Linear Programming. *Mathematical Programming*, **1**, 76-94. http://dx.doi.org/10.1007/BF01584074

[16] Gauthier, J.M. and Ribiere, G. (1977) Experiments in Mixed-Integer Linear Programming Using Pseudo-Costs. *Mathematical Programming*, **12**, 26-47. http://dx.doi.org/10.1007/BF01593767

[17] Refalo, P. (2004) Impact-Based Search Strategies for Constraint Programming. In: Wallace, M., Ed., *Principles and Practice of Constraint Programming—CP* 2004, Lecture Notes in Computer Science, Vol. 3258, Springer, Berlin, 557-571. http://dx.doi.org/10.1007/978-3-540-30201-8_41

[18] Achterberg, T., Koch, T. and Martin, A. (2005) Branching Rules Revisited. *Operations Research Letters*, **33**, 42-54. http://dx.doi.org/10.1016/j.orl.2004.04.002

[19] Gomory, R. (1969) Some Polyhedra Related to Combinatorial Problems. *Linear Algebra and Its Applications*, **2**, 451-558. http://dx.doi.org/10.1016/0024-3795(69)90017-2

[20] Cho, C., Padberg, D. and Rao, M. (1983) On the Uncapacitated Plant Location Problem. II. Facets and Lifting Theorems. *Mathematics of Operations Research*, **8**, 590-612. http://dx.doi.org/10.1287/moor.8.4.590

[21] Easton, T. and Gutierrez, T. (2015) Sequential Lifting of General Integer Variables. *Industrial Engineering and Management*, **4**, 158.

[22] Hammer, P.L., Johnson, E.L. and Peled, U.N. (1975) Facets of Regular 0-1 Polytopes. *Mathematical Programming*, **8**, 179-206. http://dx.doi.org/10.1007/BF01580442

[23] Wolsey, L. (1975) Faces for a Linear Inequality in 0-1 Variables. *Mathematical Programming*, **8**, 165-178. http://dx.doi.org/10.1007/BF01580441

[24] Easton, T. and Hooker, K. (2008) Simultaneously Lifting Sets of Binary Variables into Cover Inequalities for Knapsack Polytopes. *Discrete Optimization*, **5**, 254-261. http://dx.doi.org/10.1016/j.disopt.2007.05.003

[25] Kubik, L. (2009) Simultaneously Lifting Multiple Sets in Binary Knapsack Integer Programs. MS Thesis, Department of Industrial and Manufacturing Systems Engineering, Kansas State University, Manhattan.

[26] Zemel, E. (1978) Lifting the Facets of 0-1 Polytopes. *Mathematical Programming*, **15**, 268-277. http://dx.doi.org/10.1007/BF01609032

[27] Atamtürk, A. (2003) On the Facets of the Mixed-Integer Knapsack Polyhedron. *Mathematical Programming*, **98**, 145-175. http://dx.doi.org/10.1007/s10107-003-0400-z

[28] Gu, Z., Nemhauser, G. and Savelsbergh, M. (1998) Lifted Cover Inequalities for 0-1 Integer Programs: Computation. *INFORMS Journal on Computing*, **10**, 427-437. http://dx.doi.org/10.1287/ijoc.10.4.427

[29] Gu, Z., Nemhauser, G. and Savelsbergh, M. (1999) Lifted Cover Inequalities for 0-1 Integer Programs: Complexity. *INFORMS Journal on Computing*, **11**, 117-123. http://dx.doi.org/10.1287/ijoc.11.1.117

[30] Gu, Z., Nemhauser, G. and Savelsbergh, M. (2000) Sequence Independent Lifting in Mixed Integer Programming. *Journal of Combinatorial Optimization*, **4**, 109-129. http://dx.doi.org/10.1023/A:1009841107478

[31] Shebalov, S. and Klabjan, D. (2006) Sequence Independent Lifting for Mixed Integer Programs with Variable Upper Bounds. *Mathematical Programming*, **105**, 523-561. http://dx.doi.org/10.1007/s10107-005-0664-6

[32] Balas, E. (1975) Facets of the Knapsack Polytope. *Mathematical Programming*, **8**, 146-164. http://dx.doi.org/10.1007/BF01580440

[33] Weismantel, R. (1997) On the 0/1 Knapsack Polytope. *Mathematical Programming*, **77**, 49-68. http://dx.doi.org/10.1007/BF02614517

[34] Hickman, R. and Easton, T. (2015) Merging Valid Inequalities over the Multiple Knapsack Polyhedron. *International Journal of Operations Research*, **24**, 214-227. http://dx.doi.org/10.1504/IJOR.2015.071495

[35] Wolsey, L. (1976) Facets and Strong Valid Inequalities for Integer Programs. *Operations Research*, **24**, 367-372. http://dx.doi.org/10.1287/opre.24.2.367

[36] Chvátal, V. (1973) Edmonds Polytopes and a Hierarchy of Combinatorial Problems. *Discrete Mathematics*, **4**, 305-337. http://dx.doi.org/10.1016/0012-365X(73)90167-2

[37] Gomory, R. (1958) Outline of an Algorithm for Integer Solutions to Linear Programs. *Bulletin of the American Mathematical Society*, **64**, 275-278. http://dx.doi.org/10.1090/S0002-9904-1958-10224-4

[38] Balas, E. and Perregaard, M. (2003) A Precise Correspondence Between Lift-and-Project Cuts, Simple Disjunctive Cuts, and Mixed Integer Gomory Cuts for 0-1 Programming. *Mathematical Programming*, **94**, 221-245. http://dx.doi.org/10.1007/s10107-002-0317-y

[39] Gomory, R. and Johnson, E.L. (1972) Some Continuous Functions Related to Corner Polyhedra 1. *Mathematical Programming*, **3**, 23-85. http://dx.doi.org/10.1007/BF01584976

[40] Wolsey, L. (1977) Valid Inequalities and Superadditivity of 0/1 Integer Programs. *Mathematics of Operations Research*, **2**, 66-77. http://dx.doi.org/10.1287/moor.2.1.66

[41] OR Library Webpage (2013). http://people.brunel.ac.uk/~mastjjb/jeb/info.html

[42] Chu, P.C. and Beasley, J.E. (1998) A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*, **4**, 63-86. http://dx.doi.org/10.1023/A:1009642405419

[43] Hickman, R. (2014) Generating Cutting Planes through Inequality Merging for Integer Programming Problems. PhD Dissertation, Department of Industrial and Manufacturing Systems Engineering, Kansas State University, Manhattan.

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

[45] Fischetti, M., Lodi, A., Monaci, M., Salvagnin, D. and Tramontani, A. (2013) Tree Search Stabilization by Random Sampling. Technical Report OR/13/5, DEI, University of Bologna, Bologna.

[46] Ju, L., Huynh, B.K., Chakraborty, S. and Roychoudhury, A. (2009) Context-Sensitive Timing Analysis of Esterel Programs. *Proceedings of the* 46*th Annual Design Automation Conference*, San Francisco, 26-31 July 2009, 870-873. http://dx.doi.org/10.1145/1629911.1630132