1-30-2020

Approximate and Exact Merging of Knapsack Constraints with Cover Inequalities

Fabio Vitor
University of Nebraska at Omaha, ftorresvitor@unomaha.edu

Todd Easton
Kansas State University

Recommended Citation

Fabio Vitor & Todd Easton (2020) Approximate and exact merging of knapsack constraints with cover inequalities, Optimization, DOI: 10.1080/02331934.2020.1719492

To appear in *Optimization*
Vol. 00, No. 00, Month 20XX, 1–21

RESEARCH ARTICLE

Approximate and Exact Merging of Knapsack Constraints with Cover Inequalities

Fabio Vitor*[a] and Todd Easton[b]

[a]*Department of Mathematics, University of Nebraska at Omaha, Omaha, NE 68182, USA;* [b]*Department of Industrial and Manufacturing Systems Engineering, Kansas State University, Manhattan, KS 66506, USA*

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

This paper presents both approximate and exact merged knapsack cover inequalities, a class of cutting planes for knapsack and multiple knapsack integer programs. These inequalities combine the information from knapsack constraints and cover inequalities. Approximate merged knapsack cover inequalities can be generated through a $O(n \log n)$ algorithm, where n is the number of variables. This class of inequalities can be strengthened to an exact version with a pseudo-polynomial time algorithm. Computational experiments demonstrate an average improvement of approximately 8% in solution time and 5% in the number of ticks from CPLEX when approximate merged knapsack cover inequalities are implemented as pre-processing cuts to solve some benchmark multiple knapsack problems. Furthermore, exact merged knapsack cover inequalities improve the solution time and number of ticks of some random multiple knapsack instances by 15% and 5%, respectively.

Keywords: integer programming; cover inequalities; knapsack constraints; cutting planes; inequality merging; merged knapsack cover inequalities

AMS Subject Classification: MSC 90C10; MSC 90C27; MSC 90C39; MSC 90C57

1. Introduction

Integer program (IP) is a class of mathematical models used to formulate numerous optimization problems. Integer programs have been used by practitioners to solve several real world problems from areas such as portfolio management [1, 2], power generation [3, 4], capital budgeting [5, 6], supply chain and transportation of goods [7–12], scheduling [13, 14], and health care [15–18]. Unfortunately, IPs are $\mathcal{NP}$-hard [19] and are typically solved by the branch and bound algorithm [20].

Formally, define an IP as maximize $z = c^T x$, subject to $Ax \le b$ and $x \in \mathbb{Z}^n_+$, where n and r are positive integers, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{r \times n}$, and $b \in \mathbb{R}^r$. Denote $N = \{1, ..., n\}$ to be the set of variable indices and $R = \{1, ..., r\}$ as the set of constraint indices. Let $P = \{x \in \mathbb{Z}^n_+ : Ax \le b\}$ be the set of feasible integer solutions, (z^{IP}, x^{IP}) be any feasible integer solution, and (z^{IP*}, x^{IP*}) be an optimal solution to the IP. Furthermore, denote $P^{LR} = \{x \in \mathbb{R}^n_+ : Ax \le b\}$ as the set of feasible linear relaxation solutions, (z^{LR}, x^{LR}) to be any feasible linear relaxation solution, and (z^{LR*}, x^{LR*}) to be an optimal linear relaxation solution to the IP.

The knapsack problem (KP) is a classical IP that models the idea of a hiker that has to decide which items to pack in a knapsack for a trip. Each item has

*Corresponding author. Email: ftorresvitor@unomaha.edu

an associated benefit and weight. The problem seeks to maximize the amount of benefits constrained by a total weight limit. If the knapsack is limited to other such constraints (e.g. volume and price of items), then this problem is referred to as the multiple knapsack problem (MKP) or the multidimensional knapsack problem [21]. Observe that other formulations of MKPs also exist such as MKPs with multiple objectives [22–24]. Knapsack and multiple knapsack problems have been used in numerous different applications such as production planning and inventory control [25, 26], project and portfolio selection [27], resource allocation [28], machine scheduling [29, 30], and packing problems [31]. Observe that KPs and MKPs are also $\mathcal{NP}$-hard and substantial research has been performed to more quickly solve these classes of IPs [32–36].

Formally, define a KP as maximize $z = c^T x$, subject to $a^T x \leq b$ and $x \in \{0,1\}^n$, where $c \in \mathbb{R}^n_+$, $a \in \mathbb{R}^n_+$, and $b \in \mathbb{R}_+$. Denote $P_{KP} = \{x \in \{0,1\}^n : a^T x \leq b\}$ as the set of feasible solutions of a KP. Additionally, define an MKP as maximize $z = c^T x$, subject to $Ax \leq b$ and $x \in \{0,1\}^n$, where $c \in \mathbb{R}^n_+$, $A \in \mathbb{R}^{r\times n}_+$, and $b \in \mathbb{R}^r_+$. Let $P_{MKP} = \{x \in \{0,1\}^n : Ax \leq b\}$. Without loss of generality, assume every KP has $a_j \leq b$ for all $j \in N$ and every MKP has $a_{i,j} \leq b_i$ for all $i \in R$ and $j \in N$. Otherwise, $x_j = 0$ for all cases in which $a_j > b$ and $a_{i,j} > b_i$, and these variables can be eliminated from the problem. Furthermore, denote (z^{KP}, x^{KP}), (z^{KP*}, x^{KP*}), (z^{MKP}, x^{MKP}), and (z^{MKP*}, x^{MKP*}) as any feasible integer solution and an optimal solution for KPs and MKPs, respectively.

The concept of convexity and polyhedral theory are vital to this paper. Let $S \subseteq \mathbb{R}^n$ be convex if, and only if, $\lambda x + (1-\lambda)x' \in S$ for all x and $x' \in S$, and $\lambda \in [0,1]$. If $S \subseteq \mathbb{R}^n$, then its convex hull, S^{ch}, is the intersection of all convex sets that contain S. Furthermore, define a half space as $\{x \in \mathbb{R}^n : \sum_{j=1}^n \alpha_j x_j \leq \beta\}$, where α is a nonzero vector and β is a scalar. Define a polyhedron as the intersection of a finite number of half-spaces and a polytope as a bounded polyhedron. Observe that P^{ch}, P^{LR}, P^{ch}_{KP}, and P^{ch}_{MKP} are all polyhedra.

Frequently, cutting planes are used to improve the computational time to solve IPs. Cutting planes are valid inequalities that remove some space from P^{LR} without eliminating any $x^{IP} \in P$. For instance, lifting [37], cover cuts [38, 39], disjunctive cuts [40], Chvátal Gomory cuts [41], mixed integer rounding cuts [42], superadditive cuts [43], and modular arithmetic cuts [44] are all examples of cutting planes for integer programming problems.

Formally, any inequality of the form $\sum_{j=1}^n \alpha_j x_j \leq \beta$ is valid for P^{ch} if, and only if, $\sum_{j=1}^n \alpha_j x'_j \leq \beta$ for every $x' \in P$. If an inequality is valid and there exists an $x'' \in P^{LR}$ such that $\sum_{j=1}^n \alpha_j x''_j > \beta$, then this inequality is a cutting plane. A valid inequality $\sum_{j=1}^n \alpha_j x_j \leq \beta$ weakly dominates another valid inequality $\sum_{j=1}^n \alpha'_j x_j \leq \beta'$ if $\alpha_j \geq \alpha'_j$ for all $j \in N$ and $\beta \leq \beta'$.

The theoretical usefulness of cuttings planes is measured in terms of the dimension of the cutting plane's face $F = \{x \in P^{ch} : \sum_{j=1}^n \alpha_j x_j = \beta\}$. The dimension of a convex space, P^{ch}, is defined as the maximum number of affinely independent points minus one. Let $Q = \{x^1, x^2, ..., x^q\} \in \mathbb{R}^n$ be a set of affinely independent points if, and only if, the unique solution to $\sum_{k=1}^q \lambda_k x^k = 0$ and $\sum_{k=1}^q \lambda_k = 0$ is $\lambda_k = 0$ for all $k \in \{1, ..., q\}$. If the dimension of F equals the dimension of P^{ch} minus one, then the inequality is said to be facet defining and is one of the theoretically strongest valid inequalities [45].

Since this paper presents merged knapsack cover inequalities, a new class of cutting planes for KPs and MKPs, some background information about covers, lifting, and inequality merging is useful to understand the paper. Covers represent an infeasible solution to a knapsack constraint and are used to create valid inequalities.

Formally, $C \subseteq N$ is a cover if, and only if, $\sum_{j \in C} a_j > b$. Every cover C has a corresponding valid cover inequality $\sum_{j \in C} x_j \leq |C| - 1$. If $\sum_{j \in C \setminus \{k\}} a_j \leq b$ for each $k \in C$, then the cover is minimal. For brevity, this paper assumes that all covers are minimal.

Lifting can be used to strengthen some valid inequalities [37]. Lifting begins with a valid inequality in the restricted space $P_{E,K}$ and creates a valid inequality over the entire dimensional space P. Let $E \subset N$ be an ordered set and $K = (k_1, k_2, ..., k_{|E|}) \in \mathbb{Z}^{|E|}$. Define the restricted space $P_{E,K} = \{x \in P : x_j = k_j \ \forall j \in E\}$. Lifting begins with a valid inequality $\sum_{j \in E} \alpha_j x_j + \sum_{j \in N \setminus E} \alpha_j x_j \leq \beta$ of $P^{ch}_{E,K}$ and creates a valid inequality $\sum_{j \in E} \alpha'_j x_j + \sum_{j \in N \setminus E} \alpha_j x_j \leq \beta'$ of P^{ch}.

Valid inequalities can be: up, down, or middle lifted; exactly or approximately lifted; sequentially or simultaneously lifted. Inequalities are up, down, or middle lifted when for each $j \in E$, the values of k_j are at the lower bound, upper bound, or in between the lower and upper bound, respectively [46]. Furthermore, lifting is considered exact if α' cannot be increased and β' cannot be decreased [47–49]. On the other hand, if a lifted inequality has either an α' that can be increased or a β' that can be decreased, then the corresponding inequality is approximately lifted. Finally, an inequality is sequentially lifted when $|E| = 1$ and simultaneously lifted when $|E| > 1$ [50–53].

One particular type of simultaneous lifting is referred to as inequality merging [54, 55]. Inequality merging occurs when two low dimensional valid inequalities are merged to create a new valid inequality of higher dimension. In such a case, one valid inequality is defined as the host inequality $\sum_{j \in C^1} \alpha^1_j x_j \leq \beta^1$, while the other valid inequality is defined as the donor inequality $\sum_{j \in C^2} \alpha^2_j x_j \leq \beta^2$, where $C^1 \subseteq N$, $p \in C^1$, $C^2 \subseteq \left(N \setminus C^1\right) \cup \{p\}$, and β^1, β^2, α^1_j, $\alpha^2_j \in \mathbb{Z}_+$. Thus, the host inequality replaces at least one of its variables with a set of variables from the donor inequality and the resulting merged inequality is $\sum_{j \in C^1 \setminus \{p\}} \alpha^1_j x_j + \sum_{j \in C^2} \frac{\alpha^2_j}{\beta^2} x_j \leq \beta^1$. Observe that merged inequalities may be valid and facet defining under certain conditions. Other techniques to merge inequalities not related to this paper also exist [56, 57].

This paper presents a $O(n \log n)$ algorithm to create valid approximate merged knapsack cover inequalities. The variables of a cover inequality are merged with a set of variables from a knapsack constraint. The merged knapsack coefficient must be greater than a certain predefined value, which allows a strengthening of the cover inequality. In addition, this paper describes a technique to strengthen these approximate inequalities into an exact version. Exact merged knapsack cover inequalities are generated in pseudo-polynomial time with a dynamic programming algorithm, and are the theoretically strongest such inequalities. Computational results demonstrate an average improvement of nearly 8% in solution time and 5% in the number of ticks from CPLEX when approximate merged knapsack cover inequalities are implemented as preprocessing cuts to solve some benchmark multiple knapsack instances. Moreover, exact merged knapsack cover inequalities improve the solution time of some random multiple knapsack problems by 15% and number of ticks by 5%.

The remainder of the paper is organized as follows. Section 2 demonstrates the theory behind both approximate and exact merged knapsack cover inequalities. Section 3 provides the main computational results. Section 4 concludes the paper and presents potential topics for future work.

2. Merging Knapsack Constraints with Cover Inequalities

This section describes the theoretical and algorithmic results involving approximate and exact merged knapsack cover inequalities. Examples are provided to demonstrate this novel class of cutting planes. Preliminary results can also be found in a conference proceedings paper [58] and a thesis [59].

2.1. *Approximate Merged Knapsack Cover Inequalities*

Let $\sum_{j\in N} a_j x_j \leq b$ be a knapsack constraint, $C \subseteq N$ be a cover, $M \subseteq N$ be a set of merging indices, and $\alpha \in \mathbb{R}_+$ be a merging coefficient. Define a merged knapsack cover inequality MKC_α as $\sum_{j\in C\setminus M} x_j + \alpha \sum_{j\in M} a_j x_j \leq |C| - 1$. Observe that $\sum_{j\in C\setminus M} x_j \leq |C| - 1$ is part of the cover inequality and $\sum_{j\in M} a_j x_j$ is a portion of the knapsack constraint. If there exists an $x^{KP} \in P_{KP}$ that meet MKC_α at equality, then MKC_α is an exact merged knapsack cover inequality. If not, then MKC_α is an approximate merged knapsack cover inequality. This definition follows identically to the terms used to define when an inequality is approximately or exactly lifted.

Observe that $\alpha = 0$ results in MKC_α being weakly dominated by the cover inequality. If $\alpha > \frac{|C|-1}{\max\{a_j : j\in M\}}$, then there exists an $x^{KP} \in P_{KP}$ with $x_j = 1$ for some $j \in M$ that violates the merged knapsack cover inequality, and MKC_α is not valid. Consequently, valid merged knapsack cover inequalities must have $0 < \alpha \leq \frac{|C|-1}{\max\{a_j : j\in M\}}$ in order to be considered theoretically useful.

The approximate merging knapsack cover algorithm (AMKCA) determines valid merged knapsack cover inequalities that can be theoretically stronger than the corresponding cover inequality. This algorithm first determines a set of merging indices M and calculates a value for α such that MKC_α is valid. Algorithm 1 presents AMKCA, which requires as input a knapsack constraint $\sum_{j\in N} a_j x_j \leq b$, a cover $C \subseteq N$ such that $C = \{f_1, f_2, ..., f_{|C|}\}$, and a set $M' \subseteq C$, where $M' = \{g_1, g_2, ..., g_{|M'|}\}$. One can view M' as a set of elements in C that are merged jointly with the elements in M. That is, $M' = M \cap C$. The output to AMKCA is an $\alpha \in \mathbb{R}_+$, a set of merging indices $M = \{h_1, h_2, ..., h_{|M|}\}$, and the approximate merged knapsack cover inequality $\sum_{j\in C\setminus M} x_j + \alpha \sum_{j\in M} a_j x_j \leq |C| - 1$.

Algorithm 1 has four major steps. The first step (lines 2-12) initializes certain variables and sorts the knapsack constraint, the indices of C, and the indices of M' in descending order according to the values of a. The second step (lines 13-21) determines a set $M \subseteq N$ such that $\alpha > 0$ (Theorem 2.1). The third step (lines 22-29), which is equivalent to calculating $\alpha = \min\left\{\frac{|C|-1-q}{b-\sum_{j=|C\setminus M|-q+1}^{|C\setminus M|} a_{t_j}} : 0 \leq q \leq p\right\}$ (see Theorem 2.2), defines a value for α by efficiently utilizing the sorted knapsack coefficients to determine feasible solutions for P_{KP}. Observe that any $x^{KP} \in P_{KP}$ may decrease the current estimate for α (Theorem 2.2). The last step (lines 30-32) reports the α value, M, and MKC_α determined by AMKCA.

The most expensive step in AMKCA is to sort the coefficients of the knapsack constraint. Therefore, approximate merged knapsack cover inequalities can be generated with $O(n \log n)$ effort. To prove that the inequalities reported by AMKCA are valid and not dominated by their corresponding cover inequality, one must evaluate both steps (B) and (C) of AMKCA. Theorem 2.1 proves that AMKCA determines a set of merging indices M such that approximate merged knapsack cover inequalities are valid for some $\alpha > 0$.

Algorithm 1 : Approximate Merging Knapsack Cover Algorithm (AMKCA)

begin
(A) Initialization
  Sort $a = (a_1, a_2, ...a_n)$ such that $a_j \geq a_{j+1} \quad \forall\ j = \{1, 2, ..., |N| - 1\}$
  Sort $C = \{f_1, f_2, ..., f_{|C|}\}$ such that $a_{f_j} \geq a_{f_{j+1}} \quad \forall\ j = \{1, 2, ..., |C| - 1\}$
  Sort $M' = \{g_1, g_2, ..., g_{|M'|}\}$ such that $a_{g_j} \geq a_{g_{j+1}} \quad \forall\ j = \{1, 2, ..., |M'| - 1\}$
  $M \leftarrow \emptyset$
  $l \leftarrow |C \setminus M'|$
  $\alpha \leftarrow \infty$
  if $|M'| \leq 2$ **then**
    $p \leftarrow |C| - 2$
  else
    $p \leftarrow |C \setminus M'|$
(B) Determine M
  if $|M'| = 0$ **then**
    $\theta \leftarrow b - \sum_{j=2}^{|C|} a_{f_j}$
  else if $|M'| = 1$ **then**
    $\theta \leftarrow b - \sum_{j \in C \setminus M'} a_{f_j}$
  else
    $\theta \leftarrow 0$
  for each $j \in N \setminus (C \setminus M')$ **do**
    if $a_j > \theta$ **then** $M \leftarrow M \cup \{j\}$
(C) Calculate α
  $\theta \leftarrow b$
  for $q = 0$ **to** p **do**
    $\alpha' \leftarrow \dfrac{|C| - 1 - q}{\theta}$
    if $\alpha' < \alpha$ **then** $\alpha \leftarrow \alpha'$
    if $q < p$ **then**
      $\theta \leftarrow \theta - a_{f_l}$
      $l \leftarrow l - 1$
(D) Output
  return $\alpha \in \mathbb{R}^+$ and $M = \{h_1, h_2, ..., h_{|M|}\}$
  return $MKC_\alpha \leftarrow \sum_{j \in C \setminus M} x_j + \alpha \sum_{j \in M} a_j x_j \leq |C| - 1$
end

Theorem 2.1 *Let $\sum_{j \in N} a_j x_j \leq b$ be a knapsack constraint, $C \subseteq N$ be a cover, and $M' \subseteq C$ be a set of overlapping indices. AMKCA returns a set of merging indices M such that MKC_α is valid for P_{KP}^{ch} for some $\alpha > 0$.*

Proof. Let $\sum_{j \in N} a_j x_j \leq b$ be a knapsack constraint, $C \subseteq N$ be a cover, and $M' \subseteq C$ be a set of overlapping indices. Assume M is the set of merging indices returned by AMKCA and $\alpha = \frac{1}{b}$. To show that MKC_α is valid for P_{KP}^{ch}, let x' be any solution in P_{KP} and define $q = \sum_{j \in C \setminus M} x'_j$. Since C is a cover, then $\sum_{j \in C} x_j \leq |C| - 1$ is a valid cover inequality and it holds that $q \leq \sum_{j \in C} x'_j \leq |C| - 1$.

Assume $q \leq |C| - 2$. Applying x' to the left-hand side of MKC_α results in $q + \frac{1}{b}\big(\sum_{j \in M} a_j x'_j\big) \leq |C| - 2 + \frac{1}{b}\big(\sum_{j \in M} a_j x'_j\big)$. Since x' is feasible, $\sum_{j \in M} a_j x'_j \leq b$. Hence, $|C| - 2 + \frac{1}{b}\big(\sum_{j \in M} a_j x'_j\big) \leq |C| - 1$ and x' satisfies MKC_α.

Assume $q = |C| - 1$ and $|M'| = 1$. Thus, $x'_j = 1$ for every $j \in C \setminus M$. Since x' is feasible, $\sum_{j \in C \setminus M} a_j x'_j + \sum_{j \in M} a_j x'_j \leq b$. Therefore, $\sum_{j \in M} a_j x'_j \leq$

$b - \sum_{j \in C \setminus M} a_j$. Since M is returned from AMKCA, $a_j > \theta$ for every $j \in M$, where $\theta = b - \sum_{j \in C \setminus M'} a_{f_j}$ (lines 16-17). Thus, $x'_j = 0$ for every $j \in M$ and $\sum_{j \in C \setminus M} x'_j + \frac{1}{b}\left(\sum_{j \in M} a_j x'_j\right) = |C| - 1$.

Assume $q = |C| - 1$ and $|M'| = 0$. Since x' is feasible, $\sum_{j \in C} a_j x'_j + \sum_{j \in M} a_j x'_j \leq b$ implies that $\sum_{j \in M} a_j x'_j \leq b - \sum_{j \in C} a_j x'_j$. Because C is sorted and $q = |C| - 1$, then $\sum_{j \in C} a_j x'_j \geq \sum_{j=2}^{|C|} a_{f_j}$. Since M is returned from AMKCA, every $a_j > \theta$ where $\theta = b - \sum_{j=2}^{|C|} a_{f_j}$ (lines 14-15). Thus, $x'_j = 0$ for every $j \in M$ and $\sum_{j \in C \setminus M} x'_j + \frac{1}{b}\left(\sum_{j \in M} a_j x'_j\right) = |C| - 1$. Hence, MKC_α is valid for P_{KP}^{ch}. The cases are exhaustive and the result is shown. □

From Theorem 2.1, AMKCA returns $M = \{h_1, h_2, ..., h_{|M|}\}$ such that MKC_α is valid for some $\alpha > 0$. Theorem 2.2 proves that the α value returned by AMKCA implies that $\sum_{j \in C \setminus M} x_j + \alpha \sum_{j \in M} a_j x_j \leq |C| - 1$ is a valid merged knapsack cover inequality for P_{KP}^{ch}.

THEOREM 2.2 *Let $\sum_{j \in N} a_j x_j \leq b$ be a knapsack constraint, $C \subseteq N$ be a cover, and $M \subseteq N$ be a set of merging indices. Then, $\sum_{j \in C \setminus M} x_j + \alpha' \sum_{j \in M} a_j x_j \leq |C| - 1$ is a valid inequality of P_{KP}^{ch} for any $\alpha' \leq \alpha$, where α is returned by AMKCA.*

Proof. Let $\sum_{j \in N} a_j x_j \leq b$ be a knapsack constraint, $C \subseteq N$ be a cover, and $M \subseteq N$ be a set of merging indices. For contradiction, assume that there exists an $\alpha' \leq \alpha$ such that $\sum_{j \in C \setminus M} x_j + \alpha' \sum_{j \in M} a_j x_j \leq |C| - 1$ is not a valid inequality of P_{KP}^{ch}, where α is returned by AMKCA. Thus, there exists an $x' \in P_{KP}$ such that $\sum_{j \in C \setminus M} x'_j + \alpha' \sum_{j \in M} a_j x'_j > |C| - 1$. Define $q = \sum_{j \in C \setminus M} x'_j$ and $C \setminus M = \{t_1, t_2, ..., t_{|C \setminus M|}\}$. Since x' is feasible, $\sum_{j \in C \setminus M} a_j x'_j + \sum_{j \in M} a_j x'_j \leq b$ implies that $\sum_{j \in M} a_j x'_j \leq b - \sum_{j \in C \setminus M} a_j x'_j$. Therefore, $\sum_{j \in M} a_j x'_j \leq b - \sum_{j=|C \setminus M| - q + 1}^{|C \setminus M|} a_{t_j}$ due to the sets being sorted.

Assume $q \leq |C| - 2$. Since $MKC_{\alpha'}$ is not a valid inequality, $\alpha' \sum_{j \in M} a_j x'_j > |C| - 1 - q$. Therefore, $\alpha' > \frac{|C| - 1 - q}{\sum_{j \in M} a_j x'_j}$ implies that $\alpha' > \frac{|C| - 1 - q}{\left(b - \sum_{j=|C \setminus M| - q + 1}^{|C \setminus M|} a_{t_j}\right)}$. However, AMKCA requires $\alpha \leq \frac{|C| - 1 - q}{\left(b - \sum_{j=|C \setminus M| - q + 1}^{|C \setminus M|} a_{t_j}\right)}$ (lines 22-29). Thus, $\alpha' > \alpha$ contradicts α being returned by AMKCA.

Assume $q \geq |C| - 1$. If $q \geq |C|$, x' violates the cover inequality and contradicts C being a cover. If $q = |C| - 1$, then $\alpha' \sum_{j \in M} a_j x'_j > (|C| - 1) - (|C| - 1)$ implies that $\alpha' > 0$ and $x'_j = 1$ for some $j \in M$. Since $\alpha' > 0$, AMKCA returns an $\alpha > 0$. Thus, $\min\{a_{h_j} : j \in \{1, 2, ...|M|\}\} > b - \sum_{j=2}^{|C|} a_{f_j}$ for $|M'| = 0$ or $\min\{a_{h_j} : j \in \{1, 2, ...|M|\}\} > b - \sum_{j \in C \setminus M} a_{f_j}$ for $|M'| = 1$ (lines 14-17). Both cases contradict the feasibility of x' and the result follows. □

The following example demonstrates the implementation of AMKCA to generate valid approximate merged knapsack cover inequalities.

EXAMPLE 2.1 *Consider the knapsack constraint described in (1), where $x_j \in \{0, 1\}$ for every $j \in \{1, 2, ..., 11\}$. Let $C = \{5, 6, 7, 8, 9\}$ be a cover and $M' = \emptyset$ for this example.*

$$30x_1 + 25x_2 + 20x_3 + 15x_4 + 12x_5 + 11x_6 + 11x_7 + 10x_8 + 10x_9 + 5x_{10} + x_{11} \leq 44 \quad (1)$$

Since the knapsack constraint in (1) and the cover C are given in a sorted order, and because $M' = \emptyset$, the first steps in AMKCA initializes $M = \emptyset$, $l = 5 - 0 = 5$, $\alpha = \infty$ and $p = 5 - 2 = 3$ (lines 2-12). The subsequent step in AMKCA calculates $\theta = b - \sum_{j=2}^{|C|} a_{f_j}$, where $C = \{f_1, f_2, ..., f_{|C|}\}$ (lines 13-21). Consequently, $\theta = 2$ and M contains every $j \in N \setminus C$ such that $a_j > \theta = 2$. Therefore, $M = \{1, 2, 3, 4, 10\}$. Observe that including the index corresponding to variable x_{11} in M violates the conditions described in Theorem 2.1. The following step in AMKCA calculates a value for α such that $\alpha = \min\left\{\frac{|C|-1-q}{\theta}\right\}$ for every $q = \{0, ..., |C| - 2\}$ (lines 22-29). Thus, $\alpha = \min\left\{\frac{4}{44}, \frac{3}{34}, \frac{2}{24}, \frac{1}{13}\right\} = \frac{1}{13}$. The algorithm terminates (lines 30-32) and reports $\alpha = \frac{1}{13}$, $M = \{1, 2, 3, 4, 10\}$, and the valid approximate merged knapsack cover inequality $MKC_{\frac{1}{13}}$ presented in (2).

$$x_5 + x_6 + x_7 + x_8 + x_9 + \frac{1}{13}(30x_1 + 25x_2 + 20x_3 + 15x_4 + 5x_{10}) \leq 4 \qquad (2)$$

Observe that $MKC_{\frac{1}{13}}$ is also a cutting plane. To show that, consider the linear relaxation solution $x^{LR} = (0, 0, 0, \frac{2}{15}, 0, 1, 1, 1, 1, 0, 0) \in P^{LR}$. Applying x^{LR} to $MKC_{\frac{1}{13}}$ results in $\sum_{j \in C \setminus M} x_j^{LR} + \frac{1}{13}\sum_{j \in M} a_j x_j^{LR} = \frac{54}{13}$. Since $\frac{54}{13} > 4$, $MKC_{\frac{1}{13}}$ cuts off the corresponding x^{LR}.

2.2. *Exact Merged Knapsack Cover Inequalities*

Since the MKC_α determined by AMKCA is approximate, an obvious question is whether these inequalities can be strengthened. This paper also presents a technique to determine an $\alpha^* \in \mathbb{R}_+$ such that α^* cannot be increased and the inequality remains valid. These type of inequalities are referred to as exact merged knapsack cover inequalities.

Observe that AMKCA determines an α value such that $\alpha = \min\left\{\frac{|C|-1-q}{\theta_q}\right\}$ for every $q = \{0, ..., |C| - 2\}$, where θ_q is dependent on the value of q. Exact merged knapsack cover inequalities have an α^* such that there exists an $x^{KP} \in P_{KP}$ that meets MKC_{α^*} at equality. Thus, one can determine the value of α^* by maximizing $z = \sum_{j \in M} a_j x_j$, subject to $\sum_{j \in M} a_j x_j \leq \theta_q$ and $x_j \in \{0, 1\}$ for all $j \in M$. If $\alpha^* = \min\left\{\frac{|C|-1-q}{z_q}\right\}$ for every $q = \{0, ..., |C|-2\}$, where z_q also depends on the value of q, then an optimal solution to the aforementioned maximization problem satisfies MKC_{α^*} at equality, and MKC_{α^*} is an exact merged knapsack cover inequality. Since C is assumed to be minimal, and $q \leq |C| - 2$, then $z_q > 0$ for each of these subproblems.

The exact merging knapsack cover algorithm (EMKCA), described in Algorithm 2, uses dynamic programming to determine the largest possible value of α^*. This is achieved by tracking an array $d = (d_0, d_1, ..., d_b)$, where $d_k \in \{0, 1\}$ for each $k \in \{0, ..., b\}$ (lines 12-15). If the sum of the coefficients of any combination of feasible solutions with indices in M equals k, then $d_k = 1$. If not, $d_k = 0$ for all $k \in \{0, ..., b\}$. Once d is computed, θ is updated by the same method as in AMKCA and z is determined as the maximum index $k \leq \theta$ such that $d_k = 1$ (lines 16-29).

The input to EMKCA is a knapsack constraint $\sum_{j \in N} a_j x_j \leq b$, a cover $C \subseteq N$, and a set of merging indices $M \subseteq N$. The set M must satisfy the conditions of Theorem 2.1. Therefore, running steps (A) and (B) of AMKCA is sufficient to identify a suitable $M = \{h_1, h_2, ..., h_{|M|}\}$. The output to EMKCA is an $\alpha^* \in \mathbb{R}_+$

and the exact merged knapsack cover inequality $\sum_{j \in C \setminus M} x_j + \alpha^* \sum_{j \in M} a_j x_j \leq |C| - 1$ (lines 30-32).

Algorithm 2 : Exact Merging Knapsack Cover Algorithm (EMKCA)

1: **begin**
2: (A) INITIALIZATION
3:   $d_0 \leftarrow 1$
4:   $d_k \leftarrow 0$ for all $k \in \{1, ..., b\}$
5:   $\theta \leftarrow b$
6:   $l \leftarrow |C \setminus M|$
7:   $\alpha^* \leftarrow \infty$
8:   **if** $|C \cap M| \leq 2$ **then**
9:    $p \leftarrow |C| - 2$
10:   **else**
11:    $p \leftarrow |C \setminus M|$
12: (B) COMPUTE d
13:   **for** $q = 1$ **to** $|M|$ **do**
14:    **for** $r = b - a_{h_q}$ **to** 0 **do**
15:     **if** $d_r = 1$ **then** $d_{r+a_{h_q}} \leftarrow 1$
16: (C) CALCULATE α^*
17:   **for** $q = 0$ **to** p **do**
18:    $t \leftarrow \theta$
19:    $flag \leftarrow 0$
20:    **while** $flag = 0$ **do**
21:     **if** $d_t = 1$ **then**
22:      $z \leftarrow t$
23:      $flag \leftarrow 1$
24:     $t \leftarrow t - 1$
25:    $\alpha' \leftarrow \frac{|C| - 1 - q}{z}$
26:    **if** $\alpha' < \alpha^*$ **then** $\alpha^* \leftarrow \alpha'$
27:    **if** $q < p$ **then**
28:     $\theta \leftarrow \theta - a_{f_l}$
29:     $l \leftarrow l - 1$
30: (D) OUTPUT
31:   **return** $\alpha^* \in \mathbb{R}^+$
32:   **return** $MKC_{\alpha^*} \leftarrow \sum_{j \in C \setminus M} x_j + \alpha^* \sum_{j \in M} a_j x_j \leq |C| - 1$
33: **end**

Computational complexity of EMKCA is clearly $O(bn + n \log n)$. Since the running time is a function of input data, EMKCA is a pseudo-polynomial time algorithm that solves quickly if the right-hand side b is bounded by a constant. The following theorem formally proves that EMKCA generates valid exact merged knapsack cover inequalities.

THEOREM 2.3 *Let $\sum_{j \in N} a_j x_j \leq b$ be a knapsack constraint, $C \subseteq N$ be a cover, $M \subseteq N$ be a set of merging indices, and α^* be returned by EMKCA. Thus, $\sum_{j \in C \setminus M} x_j + \alpha' \sum_{j \in M} a_j x_j \leq |C| - 1$ is a valid inequality of P_{KP}^{ch} for any $\alpha' \leq \alpha^*$ and is not a valid inequality of P_{KP}^{ch} for any $\alpha' > \alpha^*$.*

Proof. Let $\sum_{j\in N} a_j x_j \le b$ be a knapsack constraint, $C \subseteq N$ be a cover, $M \subseteq N$ be a set of merging indices, and α^* be returned by EMKCA. Assume $\alpha' \le \alpha^*$ and let $x' \in P_{KP}$. Define $q' = \sum_{j\in C\setminus M} x'_j$ and $z' = \sum_{j\in M} a_j x'_j$. From Algorithm 2, $d_{z'} = 1$ (lines 12-15 and lines 21-22). Thus, $\alpha^* \le \frac{|C|-1-q'}{z'}$ (line 25). Consequently, $\sum_{j\in C\setminus M} x'_j + \alpha' \sum_{j\in M} a_j x'_j \le q' + \frac{|C|-1-q'}{z'} z' \le |C| - 1$. Therefore, $\sum_{j\in C\setminus M} x'_j + \alpha' \sum_{j\in M} a_j x'_j \le |C| - 1$ is valid for P^{ch}_{KP}.

Assume $\alpha' > \alpha^*$. Let q'' and z'' be the values of q and z, respectively, that generated α^* in EMKCA. Clearly, $d_{z''} = 1$ and there exists an $x'' \in P_{KP}$ such that $\sum_{j\in C\setminus M} x''_j = q''$ and $\sum_{j\in M} a_j x''_j = z''$. Applying x'' to $MKC_{\alpha'}$ results in $q'' + \alpha' z'' > q'' + \alpha^* z'' = q'' + \frac{|C|-1-q''}{z''} z'' = |C| - 1$. Thus, x'' violates $MKC_{\alpha'}$ and $MKC_{\alpha'}$ is not a valid inequality of P^{ch}_{KP}. □

Theorem 2.3 proves that merged knapsack cover inequalities reported by EMKCA are exact because the x'' from the proof demonstrates that $MKC_{\alpha'}$ supports P^{ch}_{KP}. Example 2.2 demonstrates the implementation of EMKCA.

EXAMPLE 2.2 *Consider the knapsack constraint presented in (1), the same cover $C = \{5,6,7,8,9\}$ from Example 2.1, and the set of merging indices $M = \{1,2,3,4,10\}$ returned by AMKCA.*

EMKCA initializes $d_0 = 1$, $\theta = 44$, $l = 5$, $\alpha^* = \infty$, and $p = 5 - 2 = 3$ (lines 2-11). Step (B) of EMKCA runs $|M|$ times and defines the combination of all possible integers that can be achieved by points restricted to indices in M (lines 12-15). Table 1 summarizes the final array d. Therefore, any combination of points from $\sum_{j\in C\setminus M} x_j$ results in $\sum_{j\in M} a_j x_j$ being equal to an integer represented by any index of d where $d_k = 1$ for all $k \in \{0, 1, ..., b\}$.

Table 1. Final array d of Example 2.2.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
1	0	0	0	0	**1**	0	0	0	0	**1**	0	0	0	0
30	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**
1	0	0	0	0	**1**	0	0	0	0	**1**	0	0	0	0

The algorithm continues and calculates a value for α^* such that $\alpha^* = \min\left\{\frac{|C|-1-q}{z}\right\}$ for every $q = \{0, ..., |C| - 2\}$ (lines 16-29). Therefore, $\alpha^* = \min\left\{\frac{4}{40}, \frac{3}{30}, \frac{2}{20}, \frac{1}{5}\right\} = \frac{1}{10}$ and EMKCA reports $\alpha^* = \frac{1}{10}$ along with the valid exact merged knapsack cover inequality $MKC_{\frac{1}{10}}$ presented in (3) (lines 30-32).

$$x_5 + x_6 + x_7 + x_8 + x_9 + \frac{1}{10}(30x_1 + 25x_2 + 20x_3 + 15x_4 + 5x_{10}) \le 4 \qquad (3)$$

Observe that $MKC_{\frac{1}{10}}$ is exact while $MKC_{\frac{1}{13}}$ is approximate. For instance, let $x^{KP} = (0,0,1,0,0,0,0,1,1,0,1) \in P_{KP}$. Applying x^{KP} to $MKC_{\frac{1}{10}}$ and $MKC_{\frac{1}{13}}$ results in $MKC_{\frac{1}{10}}$ being equal to 4 while $MKC_{\frac{1}{13}}$ being equal to $\frac{46}{13}$. Thus, the corresponding solution x^{KP} meets $MKC_{\frac{1}{10}}$ at equality, but not $MKC_{\frac{1}{13}}$. The reader can enumerate every $x^{KP} \in P_{KP}$ and see that the argument holds for all cases.

Clearly, $MKC_{\frac{1}{10}}$ dominates $MKC_{\frac{1}{13}}$. In addition, $MKC_{\frac{1}{10}}$ is facet defining while $MKC_{\frac{1}{13}}$ is not a facet defining inequality. Table 2 presents 11 affinely independent points that meet $MKC_{\frac{1}{10}}$ at equality as part of the facet defining proof for $MKC_{\frac{1}{10}}$. Furthermore, $MKC_{\frac{1}{10}}$ is also a cutting plane because $\sum_{j\in C\setminus M} x_j^{LR} + \frac{1}{10}\sum_{j\in M} a_j x_j^{LR} = \frac{21}{5} > 4$, where $x^{LR} = (0,0,0,\frac{2}{15},0,1,1,1,1,0,0) \in P^{LR}$ from Example 2.1. Observe that $\frac{21}{5} > \frac{54}{13}$ and $MKC_{\frac{1}{10}}$ also cuts off x^{LR} by more than $MKC_{\frac{1}{13}}$.

Table 2. Affinely independent points for $MKC_{\frac{1}{10}}$.

x_1	0	0	0	0	0	**1**	0	0	0	0	0
x_2	0	0	0	0	0	0	0	**1**	**1**	0	0
x_3	0	0	0	0	0	0	**1**	0	0	0	0
x_4	0	0	0	0	0	0	0	**1**	0	**1**	0
x_5	0	**1**	**1**	**1**	**1**	0	0	0	0	0	0
x_6	**1**	0	**1**	**1**	**1**	0	0	0	0	0	**1**
x_7	**1**	**1**	0	**1**	**1**	0	0	0	0	0	**1**
x_8	**1**	**1**	**1**	0	**1**	0	**1**	0	0	**1**	**1**
x_9	**1**	**1**	**1**	**1**	0	**1**	**1**	0	**1**	**1**	**1**
x_{10}	0	0	0	0	0	0	0	0	**1**	**1**	0
x_{11}	0	0	0	0	0	0	0	0	0	0	**1**

Sequence independent lifting [49] is similar to both approximate and exact merged knapsack cover inequalities because the lifted coefficients are correlated to the size of the knapsack coefficients. For instance, consider the same knapsack constraint and cover from Examples 2.1 and 2.2. The resulting sequence independent lifted inequality is given in (4). Observe that $MKC_{\frac{1}{10}}$ weakly dominates the sequence independent lifted inequality in this case. Moreover, the sequence independent lifted inequality in (4) is not facet defining while $MKC_{\frac{1}{10}}$ is a facet defining inequality.

$$x_5 + x_6 + x_7 + x_8 + x_9 + \frac{25}{9}x_1 + \frac{20}{9}x_2 + \frac{16}{9}x_3 + \frac{11}{9}x_4 + \frac{1}{3}x_{10} \le 4 \tag{4}$$

To demonstrate EMKCA when $M' \neq \emptyset$, consider the same knapsack constraint described in (1) and cover $C = \{5,6,7,8,9\}$. Let $M' = \{9\}$ and the inequality presented in (5) is returned by AMKCA while the inequality presented in (6) is returned by EMKCA. In this case, EMKCA's inequality weakly dominates AMKCA's inequality as well.

$$x_5 + x_6 + x_7 + x_8 + \frac{1}{12}(30x_1 + 25x_2 + 20x_3 + 15x_4 + 10x_9 + 5x_{10} + x_{11}) \le 4 \tag{5}$$

$$x_5 + x_6 + x_7 + x_8 + \frac{1}{11}(30x_1 + 25x_2 + 20x_3 + 15x_4 + 10x_9 + 5x_{10} + x_{11}) \le 4 \tag{6}$$

3. Computational Study

This paper also presents a computational study that evaluates the real effectiveness of both AMKCA and EMKCA to solve MKPs. Multiple knapsack instances were first solved in CPLEX [60], a high performance mathematical programming solver, at default settings. Thus, approximate and exact merged knapsack cover inequalities were added as preprocessing cuts, and the problems were solved again with these newly developed cutting planes.

The computational study was performed on an Intel® Core^{TM} i7-6700 3.4GHz processor with 32 GB of RAM. Both AMKCA and EMKCA were implemented in C++ using Microsoft Visual Studio and CPLEX Version 12.7. Computational experiments also stored node files in the hard drive instead of RAM in order to avoid running out of memory.

Computational experiments evaluated approximate and exact merged knapsack cover inequalities with and without overlapping variables between C and M. Therefore, computational experiments tested these inequalities with $|M'| = 0$, $|M'| = 1$, and $|M'| = 2$. Observe that only one inequality was implemented at a time for each multiple knapsack instance.

This study implemented these inequalities with minimal covers as discussed in Section 1. Furthermore, one cover inequality was generated for each constraint of the multiple knapsack instance. The cover inequality of each knapsack constraint $i \in R$ was generated by selecting variables $x_j^{LR^*}$ with the greatest ratios $r_j^{LR^*} = \frac{d_j^{LR^*}}{a_{i,j}} + x_j^{LR^*}$ for all $j \in N$, where $d_j^{LR^*}$ is the reduced cost of variable $x_j^{LR^*}$.

Additionally, the knapsack constraint $i \in R$ selected to generate the newly created inequalities for each instance was determined as the i^{th} constraint with the $\max\left\{\sum_{j \in C_i} x_j^{LR^*}\right\}$. Observe that the computational time to generate cover inequalities, approximate and exact merged knapsack cover inequalities, and perform other preprocessing operations were not included in the final results. However, these times were less than 0.01 seconds for every implementation and would not have impacted the final results.

Improvement in solution time for each instance is computed by $\Delta_{time} = \left(\frac{s_{\text{CPLEX}} - s_{\text{MKCA}}}{s_{\text{CPLEX}}}\right) \times 100\%$ and improvement in the number of ticks is calculated as $\Delta_{ticks} = \left(\frac{t_{\text{CPLEX}} - t_{\text{MKCA}}}{t_{\text{CPLEX}}}\right) \times 100\%$, where s_{MKCA} corresponds to the solution time and t_{MKCA} is the number of ticks when solved with either approximate or exact merged knapsack cover inequalities, and s_{CPLEX} is the solution time and t_{CPLEX} represents the number of ticks when solved with CPLEX at default settings. When Δ_{time} and Δ_{ticks} of all instances from a corresponding problem set are averaged, it creates $\overline{\Delta_{time}}$ and $\overline{\Delta_{ticks}}$, respectively.

3.1. *Computational Experiments for AMKCA*

This paper first describes the results obtained with the implementation of AMKCA. Multiple knapsack instances solved in this study are benchmark problems from the OR-Library [61]. These problems set are named as *mknapcb1*, *mknapcb2*, ..., *mknapcb9* and each problem set contains 30 instances with: 100, 250, and 500 variables; 5, 10, and 30 constraints. These MKPs are suggested by Chu and Beasley [32] and take the form of maximize $z = \sum_{j \in N} c_j x_j$, subject to $\sum_{j \in N} a_{i,j} x_j \leq b_i$ for all $i \in R$ and $x_j \in \{0, 1\}$ for all $j \in N$.

Each $a_{i,j}$ is integer, randomly generated, and uniformly distributed between an lb and ub, where $lb = 0$ and $ub = 1,000$. Right-hand side values were calculated as $b_i = \left\lfloor \delta \sum_{j \in N} a_{i,j} \right\rfloor$, where δ represents the tightness ratio. Cost coefficients were generated as $c_j = \sum_{i \in R} a_{i,j} + \lfloor 500\gamma_j \rfloor$, where γ_j is a uniform random number between 0 and 1. Observe that each problem set from the OR-Library has 10 instances with $\delta = 0.25$, 10 instances with $\delta = 0.50$, and 10 instances with $\delta = 0.75$.

This paper solved the following problems set from the OR-Library: *mknapcb1*, *mknapcb2*, *mknapcb3*, *mknapcb4*, *mknapcb5*, and *mknapcb7*. Results for *mknapcb6*, *mknapcb8*, and *mknapcb9* are not reported in this paper because numerous instances in each of these problems set could not be solved to optimality within a time limit of 24 hours. On average, instances in *mknapcb1*, *mknapcb2*, and *mknapcb4* were solved in less than one minute, instances in *mknapcb7* were solved in less than 20 minutes, instances in *mknapcb3* were solved in less than 1.5 hours, and instances in *mknapcb5* were solved within 3.5 hours. Since problems set *mknapcb3* and *mknapcb5* appear to be more significant, the solution time and number of ticks for each instance of these sets are presented.

Table 3. Solution time in seconds (s_{CPLEX} and s_{MKCA}) and number of ticks (t_{CPLEX} and t_{MKCA}) obtained when AMKCA is tested with instances in problem set *mknapcb3* from the OR-Library (500 variables and 5 constraints).

δ	#	z^{MKP*}	Time				Ticks			
			CPLEX	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$	**CPLEX**	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$
0.25	1	120,148	36,934	28,368	34,018	25,977	963,584	651,123	605,318	551,337
	2	117,879	533	493	272	380	43,538	41,037	30,669	33,835
	3	121,131	3,795	3,823	3,887	4,711	193,531	224,182	224,300	239,284
	4	120,804	2,822	2,701	1,746	3,007	195,990	192,826	176,250	189,761
	5	122,319	651	622	639	586	67,145	65,835	68,847	65,573
	6	122,024	983	963	868	905	150,098	117,044	110,274	111,720
	7	119,127	57,114	55,726	55,172	52,208	607,606	669,035	782,955	596,255
	8	120,568	366	418	404	373	80,258	79,576	77,837	75,227
	9	121,586	8,394	8,653	8,333	7,796	602,556	670,157	632,483	481,723
	10	120,717	6,954	7,291	4,949	7,251	319,905	346,621	240,408	356,204
	Average		**11,855**	**10,906**	**11,029**	**10,319**	**322,421**	**305,744**	**294,934**	**270,092**
0.50	11	218,428	281	230	222	221	73,447	64,293	62,036	61,718
	12	221,202	243	177	177	216	64,651	44,336	45,425	50,654
	13	217,542	2,240	983	989	699	2,049,860	387,531	345,191	154,428
	14	223,560	1,139	1,104	1,028	1,043	177,378	175,958	172,310	173,538
	15	218,966	53	44	46	44	19,209	15,880	16,270	15,923
	16	220,530	425	343	310	313	117,202	98,356	92,540	92,072
	17	219,989	155	128	129	131	49,432	42,629	43,148	44,074
	18	218,215	326	274	102	96	41,195	45,074	29,803	28,503
	19	216,976	364	350	358	350	102,715	104,596	107,187	103,102
	20	219,719	392	760	477	404	114,352	134,245	124,088	119,136
	Average		**562**	**439**	**384**	**352**	**280,944**	**111,290**	**103,800**	**84,315**
0.75	21	295,828	8	8	8	8	2,284	2,529	2,595	2,476
	22	308,086	91	75	73	92	28,000	23,586	23,327	27,662
	23	299,796	38	32	34	33	11,555	9,991	10,659	10,247
	24	306,480	214	167	163	163	60,935	50,357	48,948	49,501
	25	300,342	50	53	54	53	13,553	14,425	14,889	14,535
	26	302,571	101	48	95	57	27,565	13,905	26,959	16,463
	27	301,339	849	1,139	1,098	1,129	42,530	52,504	53,088	53,363
	28	306,454	62	18	29	45	15,435	4,532	6,118	11,775
	29	302,828	94	54	66	70	24,493	14,201	16,740	19,819
	30	299,910	185	163	171	172	49,818	46,547	48,909	48,801
	Average		**169**	**176**	**179**	**182**	**27,617**	**23,258**	**25,223**	**25,464**

Tables 3 and 4 provide the solution time and number of ticks for instances in problems set *mknapcb3* and *mknapcb5*. Let # denote which particular instance refers to the data set. Table 5 presents $\overline{\Delta_{time}}$ and $\overline{\Delta_{ticks}}$ for all 10 instances in each problem set tested in this study with $\delta = 0.25$, $\delta = 0.50$, and $\delta = 0.75$. Recall that $|N|$ is the total number of variables and $|R|$ is the total number of constraints in each problem set. For *mknapcb3*, the average percentage improvement in solution time is 9.9% when $|M'| = 0$, 15.5% when $|M'| = 1$, and 13.3% when $|M'| = 2$. Number of ticks are improved on average by 11.3% when $|M'| = 0$, 12.2% when $|M'| = 1$, and 12.4% when $|M'| = 2$. For *mknapcb5*, solution time is improved on average by 1.9%, 3.6%, and 2.5% when $|M'| = 0$, $|M'| = 1$, and $|M'| = 2$, respectively. Furthermore, the average percentage improvement in the number of ticks is 1.1% for $|M'| = 0$, 1.9% for $|M'| = 1$, and 0.8% for $|M'| = 2$.

Table 4. Solution time in seconds (s_{CPLEX} and s_{MKCA}) and number of ticks (t_{CPLEX} and t_{MKCA}) obtained when AMKCA is tested with instances in problem set *mknapcb5* from the OR-Library (250 variables and 10 constraints).

δ	#	z^{MKP*}	Time				Ticks			
			CPLEX	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$	**CPLEX**	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$
0.25	1	59,187	13,356	14,426	14,727	14,825	1,655,035	1,799,932	1,837,299	1,849,424
	2	58,781	5,383	4,870	3,918	4,761	293,903	290,739	236,981	284,223
	3	58,097	4,388	4,871	4,652	4,926	278,116	288,010	287,053	296,163
	4	61,000	20,413	20,112	21,916	22,867	2,594,848	2,483,525	2,615,185	2,694,179
	5	58,092	56,474	59,365	61,138	61,057	6,995,591	7,343,748	7,565,223	7,597,400
	6	58,824	7,528	6,544	6,777	6,399	416,930	339,391	334,522	324,025
	7	58,704	6,667	5,321	5,417	5,449	278,173	259,731	260,490	260,758
	8	58,936	40,499	47,902	49,937	47,827	5,609,246	6,662,269	6,930,348	6,638,090
	9	59,387	7,179	7,036	7,055	7,058	623,819	620,053	621,679	621,956
	10	59,208	14,683	15,931	15,323	16,652	1,329,861	1,432,828	1,377,768	1,506,192
	Average		**17,657**	**18,638**	**19,086**	**19,182**	**2,007,552**	**2,152,023**	**2,206,655**	**2,207,241**
0.50	11	110,913	10,336	13,086	11,860	12,515	840,879	1,054,597	954,854	1,008,154
	12	108,717	13,784	11,822	11,787	11,196	1,212,435	1,099,852	1,155,044	1,077,073
	13	108,932	8,859	7,670	7,669	7,579	1,014,826	890,074	889,910	879,614
	14	110,086	33,651	33,616	33,292	33,597	4,115,762	4,094,506	4,030,001	4,076,060
	15	108,485	5,942	5,124	5,246	5,175	457,495	402,188	411,568	406,124
	16	110,845	14,029	13,703	13,510	13,302	1,375,114	1,326,986	1,318,684	1,333,611
	17	106,077	16,051	13,796	14,370	14,166	1,639,047	1,440,350	1,446,479	1,446,363
	18	106,686	9,434	10,270	10,225	10,239	1,077,533	1,162,989	1,179,287	1,180,920
	19	109,829	9,530	9,282	9,268	9,326	1,021,183	1,005,362	1,003,806	1,010,045
	20	106,723	7,214	6,343	6,340	6,432	377,800	312,434	329,835	330,633
	Average		**12,883**	**12,471**	**12,357**	**12,353**	**1,313,208**	**1,278,934**	**1,271,947**	**1,274,860**
0.75	21	151,809	3,537	4,088	4,201	4,231	247,678	269,718	269,878	275,707
	22	148,772	8,590	9,419	9,127	9,140	649,820	751,309	749,183	749,183
	23	151,909	3,201	3,009	2,948	2,711	195,756	181,553	181,720	181,087
	24	151,324	2,311	2,261	2,201	2,151	161,687	154,049	157,484	157,484
	25	151,966	8,243	6,182	5,887	5,771	511,072	318,843	322,124	320,081
	26	152,109	1,619	1,527	1,521	1,531	136,431	127,645	127,110	127,988
	27	153,131	248	226	193	204	23,155	22,117	20,460	20,658
	28	153,578	9,572	10,220	10,042	10,024	1,163,710	1,254,598	1,232,946	1,230,807
	29	149,160	1,984	1,515	1,570	1,584	125,201	116,879	118,869	117,710
	30	149,704	1,747	2,005	1,716	1,872	129,039	159,085	146,597	152,603
	Average		**4,105**	**4,045**	**3,941**	**3,922**	**334,355**	**335,580**	**332,637**	**333,331**

When considering the overall average described in Table 5, solution time is improved on average by 8.0% when $|M'| = 0$, 8.3% when $|M'| = 1$, and 8.5% when $|M'| = 2$. Improvement in the number of ticks are 4.9% for $|M'| = 0$, 4.5% for $|M'| = 1$, and 5.1% for $|M'| = 2$. Observe that on average, computational results of approximate merged knapsack cover inequalities implemented with $|M'| = 0$, $|M'| = 1$, and $|M'| = 2$ are somewhat similar for the cases reported in this paper, and no conclusions can be made on which overlapping strategy is more efficient.

3.2. *Computational Experiments for EMKCA*

Exact merged knapsack cover inequalities were generated by EMKCA and also tested with benchmark instances from the OR-Library. In such a case, EMKCA's input was the same knapsack constraint and cover inequality used by AMKCA, and the set of merging indices M returned by AMKCA. Surprisingly, each benchmark instance solved with EMKCA had $\alpha^* = \alpha$. Consequently, AMKCA generated exact merged knapsack cover inequalities for all the cases.

To investigate why EMKCA did not have any $\alpha^* > \alpha$, the authors tracked the array d from EMKCA and noticed that d had the majority, if not all, of its indices

Table 5. $\overline{\Delta_{time}}$ and $\overline{\Delta_{ticks}}$ of all 10 instances for each problem set tested from the OR-Library.

Name	$\|N\|$	$\|R\|$	δ	Time			Ticks		
				$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$
mknapcb1	100	5	0.25	4.7%	-0.4%	3.4%	-1.1%	-3.9%	-2.7%
			0.50	4.6%	-2.1%	6.2%	-0.5%	2.3%	1.7%
			0.75	10.7%	12.9%	12.7%	-1.8%	-2.3%	1.7%
			Average	**6.7%**	**3.5%**	**7.4%**	**-1.1%**	**-1.3%**	**1.2%**
mknapcb2	250	5	0.25	18.7%	12.9%	12.7%	-1.8%	-2.3%	4.5%
			0.50	10.8%	13.7%	13.2%	11.0%	11.6%	11.1%
			0.75	-3.8%	-6.5%	-8.4%	-7.5%	-10.3%	-11.4%
			Average	**8.6%**	**6.8%**	**7.7%**	**5.2%**	**2.4%**	**3.7%**
mknapcb3	500	5	0.25	2.1%	12.9%	5.5%	1.9%	7.9%	8.9%
			0.50	8.6%	22.1%	24.1%	14.4%	19.5%	20.8%
			0.75	19.1%	11.5%	10.2%	17.5%	9.2%	7.5%
			Average	**9.9%**	**15.5%**	**13.3%**	**11.3%**	**12.2%**	**12.4%**
mknapcb4	100	10	0.25	11.1%	10.3%	10.0%	4.9%	4.1%	3.9%
			0.50	10.0%	9.6%	9.5%	9.3%	8.8%	9.5%
			0.75	3.4%	4.1%	3.6%	3.5%	3.1%	3.9%
			Average	**8.1%**	**8.0%**	**7.7%**	**5.9%**	**5.3%**	**5.8%**
mknapcb5	250	10	0.25	-0.5%	-0.2%	-2.8%	-1.3%	-0.4%	-3.0%
			0.50	3.7%	4.7%	4.7%	3.5%	3.6%	3.6%
			0.75	2.4%	6.2%	5.8%	1.2%	2.6%	2.0%
			Average	**1.9%**	**3.6%**	**2.5%**	**1.1%**	**1.9%**	**0.8%**
mknapcb7	100	30	0.25	17.3%	15.9%	17.2%	8.1%	7.3%	9.1%
			0.50	12.9%	13.1%	11.8%	6.1%	5.2%	4.8%
			0.75	8.4%	7.7%	7.8%	7.5%	6.3%	6.2%
			Average	**12.9%**	**12.2%**	**12.3%**	**7.3%**	**6.3%**	**6.7%**
Overall Average				**8.0%**	**8.3%**	**8.5%**	**4.9%**	**4.5%**	**5.1%**

marked with a 1. Hunsaker and Tovey [62] show that when knapsack instances are randomly generated, there exists a set $G \subseteq N$ with probability approaching 1 such that $\sum_{j\in G} a_j x_j = b$. Consequently, many random knapsack instances eliminate gaps in d and so $\alpha^* = \alpha$.

To identify instances in which EMKCA produces inequalities that dominate the inequalities generated by AMKCA, observe that if all $a_{i,j}$ are random between 1 and some $a_{\max}$, then several small and one large number can be combined to create numerous combinations between 1 and $2 \times a_{\max}$. Therefore, EMKCA may be useful if $a_{i,j}$ is between some $a_{\min}$ and $a_{\max}$, where $a_{\min}$ is at least 50% of $a_{\max}$. Since there does not exist any publicly available benchmark instances that follow this criterion, new instances were randomly generated to test EMKCA. This paper attempted two sets of problems where $a_{\min}$ is 50% and 60% of $a_{\max}$. These instances follow the same form proposed by Chu and Beasley [32] but instead, they have constraint coefficients from $2,500$ to $5,000$ ($lb = 2,500$ and $ub = 5,000$) and from $3,000$ to $5,000$ ($lb = 3,000$ and $ub = 5,000$). Additional experiments where $a_{\min}$ is 90% of $a_{\max}$ are also provided in Vitor [59].

Since the knapsack coefficients of the newly generated random instances are greater than the knapsack coefficients of the instances from the OR-Library, problems of the same size (100, 250, and 500 variables vs. 5, 10, and 30 constraints) are computationally challenging and could not be solved within 24 hours. Consequently, only the following variations of variables and constraints were tested: 40×5, 60×5, 80×5, and 100×5. Similar to the OR-Library, 30 instances were generated for each of these problems set where 10 instances have $\delta = 0.25$, 10 instances have $\delta = 0.50$, and 10 instances have $\delta = 0.75$.

On average, instances with 40 variables and 5 constraints were solved in less than 30 seconds, 60 variables and 5 constraints were solved within 15 minutes, 80 variables and 5 constraints were solved in less than 30 minutes, and instances with 100 variables and 5 constraints were solved within 2 hours. For simplicity, only instances that are more significant (80×5 and 100×5) have the solution time and number of ticks reported in this paper. In addition, only the solution time and number of ticks from MKPs where $lb = 3,000$ and $ub = 5,000$ are presented since the time and number of ticks required to solve instances where $lb = 2,500$ and $ub = 5,000$ are in the same order of magnitude.

Table 6 describes the results for instances with 80 variables and 5 constraints, and Table 7 presents the results for instances with 100 variables and 5 constraints when $lb = 3,000$ and $ub = 5,000$. Table 8 shows $\overline{\Delta_{time}}$ and $\overline{\Delta_{ticks}}$ for MKPs where $lb = 2,500$ and $ub = 5,000$, while Table 9 describes the results for MKPs with $lb = 3,000$ and $ub = 5,000$.

Table 6. Solution time in seconds (s_{CPLEX} and s_{MKCA}) and number of ticks (t_{CPLEX} and t_{MKCA}) obtained when EMKCA is tested with the newly generated random instances (80 variables and 5 constraints, $lb = 3,000$ and $ub = 5,000$).

δ	#	z^{MKP*}	Time				Ticks			
			CPLEX	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$	**CPLEX**	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$
0.25	1	306,856	13	14	12	12	1,483	1,591	1,429	1,416
	2	305,174	412	308	317	330	34,013	25,872	29,003	27,606
	3	302,659	274	153	147	179	16,094	14,492	14,314	15,909
	4	305,681	240	64	62	75	10,452	7,941	8,043	8,721
	5	305,412	1,119	873	784	778	137,329	123,051	120,288	122,359
	6	304,329	444	328	331	320	31,470	28,846	28,561	29,076
	7	307,051	345	188	185	188	21,002	17,116	17,206	17,617
	8	305,596	252	98	102	95	11,907	9,628	10,240	9,758
	9	304,410	414	337	282	268	28,012	22,456	23,102	22,454
	10	307,926	280	85	83	82	11,312	9,087	9,249	9,035
	Average		**379**	**245**	**230**	**233**	**30,307**	**26,008**	**26,144**	**26,395**
0.50	11	612,457	11	8	7	11	1,151	1,037	1,062	1,269
	12	609,312	804	635	526	522	79,464	90,415	75,540	76,624
	13	604,381	869	555	521	528	86,548	73,147	74,140	73,818
	14	610,098	4,782	3,039	3,165	3,326	849,785	654,599	662,967	708,887
	15	610,287	828	495	647	655	83,604	63,987	72,838	70,892
	16	607,745	1,367	959	901	866	176,228	141,850	145,215	142,257
	17	610,020	1,538	1,035	1,081	1,028	233,426	190,440	202,157	180,326
	18	608,220	73	49	47	48	3,933	4,392	4,362	4,443
	19	614,610	381	243	244	223	28,805	22,464	21,971	20,657
	20	615,434	2,980	1,083	1,081	1,090	521,019	187,990	192,793	190,608
	Average		**1,363**	**810**	**822**	**830**	**206,396**	**143,032**	**145,305**	**146,978**
0.75	21	915,422	228	89	89	87	13,058	10,553	10,486	10,417
	22	911,537	403	440	505	447	28,973	35,164	38,479	35,391
	23	904,455	249	95	97	94	9,328	10,778	11,034	10,737
	24	913,181	561	449	442	451	47,547	42,785	41,207	41,421
	25	913115	1,957	1,146	1,142	1,166	332,707	227,867	230,703	234,761
	26	909,672	169	63	61	61	6,777	7,291	7,451	7,506
	27	917,076	1,575	1,164	1,371	1,419	236,014	187,904	201,840	200,749
	28	912,923	423	290	418	252	31,820	23,991	28,066	20,614
	29	909,901	437	342	326	341	33,535	32,020	31,301	32,783
	30	919,204	596	360	364	384	44,869	37,674	35,485	40,203
	Average		**660**	**444**	**482**	**470**	**78,463**	**61,603**	**63,605**	**63,458**

Overall, the average percentage improvement in solution time for MKPs where $lb = 2,500$ and $ub = 5,000$ (Table 8) is 12.8% when $|M'| = 0$, 13.2% when $|M'| = 1$, and 15.6% when $|M'| = 2$. Furthermore, the number of ticks are reduced on average by 5.1% when $|M'| = 0$, 5.3% when $|M'| = 1$, and 7.1% when $|M'| = 2$. For MKPs where $lb = 3,000$ and $ub = 5,000$ (Table 9), solution time is improved on average by 16.3%, 16.1%, and 14.6% when $|M'| = 0$, $|M'| = 1$, and $|M'| = 2$, respectively.

Moreover, the average percentage improvement in the number of ticks is 5.1%, 4.3%, and 3.8% when $|M'| = 0$, $|M'| = 1$, and $|M'| = 2$, respectively.

Table 7. Solution time in seconds (s_{CPLEX} and s_{MKCA}) and number of ticks (t_{CPLEX} and t_{MKCA}) obtained when EMKCA is tested with the newly generated random instances (100 variables and 5 constraints, $lb = 3,000$ and $ub = 5,000$).

δ	#	z^{MKP*}	**Time**				**Ticks**			
			CPLEX	$\lvert M'\rvert=0$	$\lvert M'\rvert=1$	$\lvert M'\rvert=2$	**CPLEX**	$\lvert M'\rvert=0$	$\lvert M'\rvert=1$	$\lvert M'\rvert=2$
	1	409,485	1,000	1,152	1,244	1,168	132,061	151,905	139,139	132,295
	2	405,262	19,677	21,848	22,516	23,147	3,458,479	3,378,492	3,552,690	3,586,408
	3	407,570	935	846	830	857	67,511	62,109	60,525	61,998
	4	409,966	3,680	4,266	4,375	4,356	784,194	838,738	822,419	827,682
0.25	5	406,789	9,693	9,561	9,361	9,771	1,820,472	1,764,526	1,771,318	1,818,242
	6	408,716	25,470	26,976	26,657	25,768	4,835,167	5,090,715	4,981,446	4,716,287
	7	406,266	5,672	5,141	5,113	4,852	898,976	850,390	852,056	793,220
	8	409,210	1,770	1,636	1,692	1,587	230,707	221,115	235,992	223,520
	9	410,335	10,871	12,015	12,290	12,014	2,426,846	2,256,937	2,330,600	2,276,982
	10	410,206	6,139	5,527	5,535	5,638	1,027,278	958,257	947,974	945,826
	Average		**8,491**	**8,897**	**8,961**	**8,916**	**1,568,169**	**1,557,318**	**1,569,416**	**1,538,246**
	11	810,884	1,264	1,078	1,117	1,095	178,991	158,193	171,134	163,621
	12	812,400	7,355	6,518	6,645	6,528	1,644,317	1,479,536	1,494,633	1,481,837
	13	814,371	279	300	274	274	20,521	23,309	20,489	20,295
	14	817,131	3,633	3,923	4,418	4,446	785,714	760,945	836,401	826,337
0.50	15	814,120	5,402	6,087	7,089	7,106	1,179,329	1,202,764	1,498,645	1,469,905
	16	805,030	869	863	853	849	94,711	91,784	91,954	91,581
	17	814,805	5,494	3,719	3,914	3,968	1,012,339	653,912	687,046	699,573
	18	817,795	2,680	2,511	2,503	2,544	429,913	409,150	412,954	413,551
	19	820,410	2,896	3,503	3,602	3,693	611,374	711,224	728,024	730,709
	20	809,899	2,433	2,458	2,473	2,517	407,233	439,357	446,258	430,681
	Average		**3,231**	**3,096**	**3,289**	**3,302**	**636,444**	**593,017**	**638,754**	**632,809**
	21	1,215,759	2,024	1,530	1,587	1,524	428,431	240,878	254,296	239,567
	22	1,212,670	2,725	3,049	2,741	2,725	511,120	542,932	518,563	511,920
	23	1,224,965	1,061	950	980	931	134,792	125,789	132,225	126,773
	24	1,213,710	2,757	3,089	3,078	3,139	587,512	622,368	615,261	621,154
0.75	25	1,221,249	821	748	779	811	98,047	85,380	86,961	92,394
	26	1,225,155	6,412	5,926	5,880	5,967	1,255,618	1,179,438	1,170,891	1,209,091
	27	1,225,627	3,379	3,591	3,758	3,773	774,516	739,747	753,734	781,725
	28	1,218,795	485	478	465	476	32,451	30,950	30,095	30,810
	29	1,218,137	1,648	1,572	1,655	1,771	253,518	256,369	283,357	309,545
	30	1,218,358	987	910	881	873	179,770	171,268	169,532	168,094
	Average		**2,230**	**2,184**	**2,180**	**2,199**	**425,578**	**399,512**	**401,492**	**409,107**

Table 10 describes the percentage improvement in α^* for all newly generated random instances tested with EMKCA. Table 10 was obtained by implementing both EMKCA and AMKCA on each instance and comparing the returned merging coefficients, α^* and α. Percentage improvement in α^* is defined for each instance as $\Delta_{\alpha^*} = \left(\frac{\alpha^*-\alpha}{\alpha}\right) \times 100\%$, and $\overline{\Delta_{\alpha^*}}$ is defined similarly as $\overline{\Delta_{time}}$ and $\overline{\Delta_{ticks}}$. Overall, implementing EMKCA instead of AMKCA results in an α^* that is on average 8.5%, 3.3%, and 1.8% greater than the α from AMKCA for $|M'| = 0$, $|M'| = 1$, and $|M'| = 2$, respectively.

When combining both sets of newly generated random multiple knapsack instances ($lb = 2,500$ and $ub = 5,000$ plus $lb = 3,000$ and $ub = 5,000$), the average percentage improvement in solution time is 14.6% when $|M'| = 0$, 14.7% when $|M'| = 1$, and 15.1% when $|M'| = 2$. Furthermore, the average percentage improvement in the number of ticks is 5.1%, 4.8%, and 5.5% when $|M'| = 0$, $|M'| = 1$, and $|M'| = 2$, respectively. Since improvement in solution time and number of ticks for each of the three overlapping strategies is somewhat similar, conclusions on which strategy is more efficient cannot be determined based only on the experiments performed for this paper. On the other hand, improvements in α^* appears to be more significant when $|M'| = 0$ than when $|M'| = 1$ and $|M'| = 2$. Consequently, one can infer that even a small improvement in α^* can result in substantial improvements in solution time when implementing EMKCA against CPLEX at default settings.

Table 8. $\overline{\Delta_{time}}$ and $\overline{\Delta_{ticks}}$ of all 10 problems from the newly generated random instances with $lb = 2,500$ and $ub = 5,000$.

$\|N\|$	$\|R\|$	δ	Time			Ticks		
			$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$
40	5	0.25	12.6%	4.8%	10.2%	-0.6%	-1.9%	-1.6%
		0.50	12.0%	11.9%	10.4%	5.8%	6.3%	5.8%
		0.75	21.0%	23.8%	22.5%	-5.4%	-4.1%	2.1%
		Average	**15.2%**	**13.5%**	**14.4%**	**-0.1%**	**0.1%**	**2.1%**
60	5	0.25	1.3%	2.7%	9.2%	-1.6%	-1.8%	1.4%
		0.50	-7.6%	-5.1%	16.7%	-2.6%	-4.3%	12.3%
		0.75	10.5%	11.0%	12.2%	3.2%	3.2%	5.5%
		Average	**1.4%**	**2.9%**	**12.7%**	**-0.3%**	**-1.0%**	**6.4%**
80	5	0.25	38.4%	38.4%	38.5%	18.8%	19.5%	17.5%
		0.50	41.6%	42.6%	41.6%	17.9%	17.9%	16.5%
		0.75	32.1%	33.6%	31.1%	13.2%	12.5%	11.2%
		Average	**37.3%**	**38.2%**	**37.1%**	**16.7%**	**16.6%**	**15.1%**
100	5	0.25	-4.7%	-4.5%	-5.3%	0.4%	1.3%	-1.7%
		0.50	5.5%	5.9%	7.5%	17.6%	19.0%	18.9%
		0.75	-9.0%	-7.5%	-7.2%	-6.0%	-3.9%	-3.3%
		Average	**-2.7%**	**-2.0%**	**-1.7%**	**4.0%**	**5.5%**	**4.6%**
Overall Average			**12.8%**	**13.2%**	**15.6%**	**5.1%**	**5.3%**	**7.1%**

Table 9. $\overline{\Delta_{time}}$ and $\overline{\Delta_{ticks}}$ of all 10 problems from the newly generated random instances with $lb = 3,000$ and $ub = 5,000$.

$\|N\|$	$\|R\|$	δ	Time			Ticks		
			$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$
40	5	0.25	14.1%	24.1%	23.2%	-0.6%	1.9%	7.2%
		0.50	0.9%	7.4%	5.9%	-4.2%	-0.7%	-0.9%
		0.75	27.6%	28.0%	24.4%	9.6%	9.9%	5.3%
		Average	**14.2%**	**19.8%**	**17.8%**	**1.6%**	**3.7%**	**3.9%**
60	5	0.25	22.7%	15.6%	13.5%	7.8%	5.0%	2.1%
		0.50	5.9%	7.1%	7.8%	-2.9%	-1.7%	-2.3%
		0.75	12.1%	5.3%	-4.9%	0.4%	-5.7%	-10.9%
		Average	**13.6%**	**9.3%**	**5.5%**	**1.8%**	**-0.8%**	**-3.7%**
80	5	0.25	38.0%	41.6%	39.8%	14.7%	14.2%	13.4%
		0.50	35.5%	36.5%	33.2%	17.0%	16.9%	16.1%
		0.75	35.6%	30.1%	34.5%	8.2%	5.6%	7.7%
		Average	**36.4%**	**36.1%**	**35.9%**	**13.3%**	**12.2%**	**12.4%**
100	5	0.25	-2.1%	-3.7%	-2.4%	0.9%	1.2%	3.0%
		0.50	1.5%	-1.8%	-2.3%	2.8%	-1.0%	0.2%
		0.75	3.5%	3.4%	2.6%	7.0%	5.8%	4.2%
		Average	**1.0%**	**-0.7%**	**-0.7%**	**3.6%**	**2.0%**	**2.5%**
Overall Average			**16.3%**	**16.1%**	**14.6%**	**5.1%**	**4.3%**	**3.8%**

In summary, approximate merged knapsack cover inequalities improved the solution time and number of ticks by nearly 8% and 5%, respectively, while exact merged knapsack cover inequalities improved the solution time by 15% and the number of ticks by 5%. Based on the computational results presented in this paper, both AMKCA and EMKCA help reduce the effort to solve multiple knapsack instances. The authors recommend implementing EMKCA to solve MKPs in which the knapsack coefficients are restricted to the cases where the minimum knapsack coefficient is at least 50% of the maximum knapsack coefficient. Otherwise, AMKCA should be implemented.

Table 10. $\overline{\Delta_{\alpha^*}}$ of all 10 problems from the newly generated random instances with $lb = 2,500$ and $ub = 5,000$, and $lb = 3,000$ and $ub = 5,000$.

$\|N\|$	$\|R\|$	δ	$lb = 2,500$ and $ub = 5,000$			$lb = 3,000$ and $ub = 5,000$		
			$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$	$\|M'\|=0$	$\|M'\|=1$	$\|M'\|=2$
40	5	0.25	9.4%	5.4%	4.2%	15.8%	4.9%	1.3%
		0.50	3.5%	6.7%	3.2%	11.4%	6.1%	3.4%
		0.75	4.7%	9.6%	2.5%	14.8%	4.5%	0.9%
		Average	**5.9%**	**7.2%**	**3.3%**	**14.0%**	**5.1%**	**1.9%**
60	5	0.25	5.8%	3.0%	2.5%	10.5%	4.1%	2.5%
		0.50	8.3%	0.5%	3.9%	11.1%	2.6%	0.8%
		0.75	4.9%	6.2%	1.0%	9.9%	3.5%	1.9%
		Average	**6.3%**	**3.2%**	**2.5%**	**10.5%**	**3.4%**	**1.7%**
80	5	0.25	1.8%	0.5%	3.1%	10.2%	0.5%	0.2%
		0.50	2.6%	1.2%	2.8%	11.2%	0.8%	0.8%
		0.75	4.0%	2.0%	0.7%	10.2%	1.4%	0.3%
		Average	**2.8%**	**1.3%**	**2.2%**	**10.5%**	**0.9%**	**0.4%**
100	5	0.25	4.3%	3.2%	1.9%	19.8%	5.5%	1.3%
		0.50	7.2%	0.1%	0.1%	9.8%	2.3%	2.0%
		0.75	3.5%	0.4%	0.5%	8.8%	4.3%	1.8%
		Average	**5.0%**	**1.2%**	**0.8%**	**12.8%**	**4.0%**	**1.7%**
Overall Average			**5.0%**	**3.2%**	**2.2%**	**12.0%**	**3.4%**	**1.4%**

4. Conclusions and Future Research

Merged knapsack cover inequalities are a new class of cutting planes for multiple knapsack problems. These inequalities merge the variables of a knapsack constraint with the variables of a cover inequality. This paper presents an algorithm that requires $O(n \log n)$ effort, where n is the number of variables, to generate valid approximate merged knapsack cover inequalities. Furthermore, a dynamic programming technique, which runs in pseudo-polynomial time, improves these approximate inequalities into an exact version.

Computational experiments tested the effectiveness of both approximate and exact merged knapsack cover inequalities versus CPLEX at default settings. In the studied benchmark multiple knapsack instances, approximate merged knapsack cover inequalities improved the solution time on average by nearly 8% and the number of ticks by 5%. Moreover, exact merged knapsack cover inequalities improved the solution time and number of ticks of the newly generated random instances by 15% and 5%, respectively.

Future research topics include identifying new classes of inequalities that can be merged to create strong cutting planes and perform computational experiments to show the effectiveness of these new inequalities. Examining new overlapping strategies other than the ones presented in this paper is also a potential future research topic. Computationally comparing the methods described in this paper to other algorithms to solve multiple knapsack problems [33, 35, 49] would also be beneficial to merged knapsack cover inequalities. Finally, one could view merging as a new type of lifting, called inequality lifting. Exploring methods to quickly perform inequality lifting could create new and useful classes of cutting planes.

References

[1] Pinto RLV, Rustem B. Solving a mixed-integer multiobjective bond portfoliomodel involving logical conditions. Ann Oper Res. 1998;81(0):497–514.

[2] Bertsimas D, Darnell C, Soucy R. Portfolio construction through mixed-integer programming at Grantham, Mayo, Van Otterloo and Company. Interfaces. 1999;29(1):49–66.

[3] Carrion M, Arroyo JM. A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. IEEE Transactions on Power Systems. 2006;21(3):1371–1378.

[4] Zhan Y, Zheng QP. A multistage decision-dependent stochastic bilevel programming approach for power generation investment expansion planning. IISE Transactions. 2018;50(8):720–734.

[5] Finn FJ. Integer programming, linear programming and capital budgeting. Abacus. 1973;9(2):180–192.

[6] Iwamura K, Liu B. Dependent-chance integer programming applied to capital budgeting. J Oper Res Soc Jpn. 1999;42(2):117–127.

[7] Arunapuram S, Mathur K, Solow D. Vehicle routing and scheduling with full truckloads. Transport Sci. 2003;37(2):170–182.

[8] Ruiz R, Maroto C, Alcaraz J. A decision support system for a real vehicle routing problem. Eur J Oper Res. 2004;153(3):593–606.

[9] Sinha AK, Davich T, Krishnamurthy A. Optimisation of production and subcontracting strategies. International Journal of Production Research. 2016;54(8):2377–2393.

[10] Gifford T, Opicka T, Sinha A, Brink DV, Gifford A, Randall R. Dispatch optimization in bulk tanker transport operations. INFORMS Journal on Applied Analytics. 2018; 48(5):403–421.

[11] Albashabsheh NT, Heier Stamm JL. Optimization of lignocellulosic biomass-to-biofuel supply chains with mobile pelleting. Transportation Research Part E: Logistics and Transportation Review. 2019;122(1):545–562.

[12] Delli U, Sinha AK. Parallel computation framework for optimizing trailer routes in bulk transportation. Journal of Industrial Engineering International. 2019;:1–11.

[13] Subramanian R, Scheff RP, Quillinan JD, Wiper DS, Marsten RE. Coldstart: Fleet assignment at Delta Air lines. Interfaces. 1994;24(1):104–120.

[14] Easton K, Nemhauser G, Trick M. Solving the travelling tournament problem: A combined integer programming and constraint programming approach. In: Burke E, Causmaecker PD, editors. Practice and theory of automated timetabling iv. Vol. 2740 of Lecture Notes in Computer Science; Berlin Heidelberg: Springer; 2003. p. 100–109.

[15] Lee EK, Fox T, Crocker I. Integer programming applied to intensity-modulated radiation therapy treatment planning. Ann Oper Res. 2003;119(1):165–181.

[16] Lee EK, Zaider M. Mixed integer programming approaches to treatment planning for brachytherapy application to permanent prostate implants. Ann Oper Res. 2003; 119(1):147–163.

[17] Muggy L, Heier Stamm JL. Dynamic, robust models to quantify the impact of decentralization in post-disaster health care facility location decisions. Operations Research for Health Care. 2017;12(1):43–59.

[18] Heier Stamm JL, Serban N, Swann J, Wortley P. Quantifying and explaining accessibility with application to the 2009 H1N1 vaccination campaign. Health Care Management Science. 2017;20(1):76–93.

[19] Karp RM. Reducibility among combinatorial problems. In: Miller RE, Thatcher JW, Bohlinger JD, editors. Complexity of computer computations. The IBM Research Symposia Series; New York: Plenum; 1972. p. 85–103.

[20] Land AH, Doig AG. An automatic method of solving discrete programming problems. Econometrica. 1960;28(3):497–520.

[21] Puchinger J, Raidl GR, Pferschy U. The multidimensional knapsack problem: Structure and algorithms. INFORMS J Comput. 2010;22(2):250–265.

[22] Florios K, Mavrotas G, Diakoulaki D. Solving multiobjective, multiconstraint knapsack problems using mathematical programming and evolutionary algorithms. Eur J Oper Res. 2010;203(1):14–21.

[23] Lust T, Teghem J. The multiobjective multidimensional knapsack problem: a survey and a new approach. International Transactions in Operational Research. 2012; 19(4):495–520.

[24] Mavrotas G, Florios K, Figueira JR. An improved version of a core based algorithm for the multi-objective multi-dimensional knapsack problem: A computational study and comparison with meta-heuristics. Appl Math Comput. 2015;270(1):25–43.

[25] Dawande M, Kalagnanam J, Keskinocak P, Salman FS, Ravi R. Approximation algorithms for the multiple knapsack problem with assignment restrictions. J Comb Optim. 2000;4(2):171–186.

[26] MBretthauer K, Shetty B, Syam S, Vokurka RJ. Production and inventory management under multiple resource constraints. Math Comput Model. 2006;44(1-2):85–95.

[27] Chang P, Lee J. A fuzzy DEA and knapsack formulation integrated model for project selection. Comput Oper Res. 2012;39(1):112–125.

[28] Babaioff M, Immorlica N, Kempe D, Kleinberg R. A knapsack secretary problem with applications. In: Charikar M, Jansen K, Reingold O, Rolim JD, editors. Approximation, randomization, and combinatorial optimization. Vol. 4627 of Lecture Notes in Computer Science; Berlin Heidelberg: Springer; 2007. p. 16–28.

[29] Kolliopoulos SG, Steiner G. Partially ordered knapsack and applications to scheduling. Discrete Appl Math. 2007;155(8):889–897.

[30] Kellerer H, Strusevich VA. Fully polynomial approximation schemes for a symmetric quadratic knapsack problem and its scheduling applications. Algorithmica. 2010; 57(4):769–795.

[31] Tsai J, Wang P, Lin M. A global optimization approach for solving three-dimensional open dimension rectangular packing problems. Optimization. 2014;64(12):2601–2618.

[32] Chu PC, Beasley JE. A genetic algorithm for the multidimensional knapsack problem. J Heuristics. 1998;4(1):63–86.

[33] Vasquez M, Vimont Y. Improved results on the 0-1 multidimensional knapsack problem. Eur J Oper Res. 2005;165(1):70–81.

[34] Ahuja RK, Cunha CB. Very large-scale neighborhood search for the k-constraint multiple knapsack problem. J Heuristics. 2005;11(5-6):465–481.

[35] Boussier S, Vasquez M, Vimont Y, Hanafi S, Michelon P. A multi-level search strategy for the 0-1 multidimensional knapsack problem. Discrete Appl Math. 2010;158(2):97–109.

[36] Zeng B, Richard JP. A polyhedral study on 0-1 knapsack problems with disjoint cardinality constraints: Strong valid inequalities by sequence-independent lifting. Discrete Optim. 2011;8(2):259–276.

[37] Gomory RE. Some polyhedra related to combinatorial problems. Linear Algebra Appl. 1969;2(4):451–558.

[38] Balas E, Zemel E. Facets of the knapsack polytope from minimal covers. SIAM J Appl Math. 1978;34(1):119–148.

[39] Zemel E. Easily computable facets of the knapsack polytope. Math Oper Res. 1989; 14(4):760–764.

[40] Balas E. Disjunctive programming. In: Hammer P, Johnson E, Korte B, editors. Discrete optimization II. Vol. 5 of Annals of Discrete Mathematics; Amsterdam: North-Holland Publishing Company; 1979. p. 3–51.

[41] Chvátal V. Edmonds polytopes and a hierarchy of combinatorial problems. Discrete Math. 1973;4(4):305–337.

[42] Gomory RE. Outline of an algorithm for integer solutions to linear programs. B Am Math Soc. 1958;64(5):275–278.

[43] Wolsey LA. Valid inequalities and superadditivity for 0-1 integer programs. Math Oper Res. 1977;2(1):66–77.

[44] Gomory RE. On the relation between integer and non-integer solutions to linear programs. P Natl Acad Sci USA. 1965;53(2):260–265.

[45] Nemhauser GL, Wolsey LA. Integer and combinatorial optimization. New York: John Wiley and Sons; 1999.
[46] Easton T, Gutierrez T. Sequential lifting of general integer variables for integer programs. Ind Eng Manage. 2015;4(2):7pp.
[47] Balas E. Facets of the knapsack polytope. Math Program. 1975;8(1):146–164.
[48] Zemel E. Lifting the facets of zero-one polytopes. Math Program. 1978;15(1):268–277.
[49] Gu Z, Nemhauser GL, Savelsbergh MWP. Sequence independent lifting in mixed integer programming. J Comb Optim. 2000;4(1):109–129.
[50] Hammer PL, Johnson EL, Peled UN. Facet of regular 0-1 polytopes. Math Program. 1975;8(1):179–206.
[51] Cho DC, Padberg MW, Rao MR. On the uncapacitated plant location problem II: Facets and lifting theorems. Math Oper Res. 1983;8(4):590–612.
[52] Atamtürk A. On the facets of the mixedinteger knapsack polyhedron. Math Program. 2003;98(1):145–175.
[53] Easton T, Hooker K. Simultaneously lifting sets of binary variables into cover inequalities for knapsack polytopes. Discrete Optim. 2008;5(2):254–261.
[54] Hickman R, Easton T. Merging valid inequalities over the multiple knapsack polyhedron. International Journal of Operational Research. 2015;24(2):214–227.
[55] Hickman R, Easton T. On merging cover inequalities for multiple knapsack problems. Open Journal of Optimization. 2015;4(4):141–155.
[56] Dey SS, Richard JP. Sequential-merge facets for two-dimensional group problems. In: Fischetti M, Williamson DP, editors. Integer programming and combinatorial optimization. Vol. 4513 of Lecture Notes in Computer Science; Berlin Heidelberg: Springer; 2007. p. 30–42.
[57] Dey SS, Wolsey LA. Two row mixed-integer cuts via lifting. Math Program. 2010; 124(1):143–174.
[58] Vitor F, Easton T. Merged knapsack cover inequalities for the multiple knapsack problem. In: Yang H, Kong Z, Sarder M, editors. Proceedings of the 2016 Industrial and Systems Engineering Research Conference; may. Institute of Industrial and Systems Engineers; 2016. p. 607–612.
[59] Vitor FT. Improving the solution time of integer programs by merging knapsack constraints with cover inequalities [master's thesis]. Kansas State University; 2015.
[60] IBM. CPLEX Optimizer. Version 12.7. 2016; Available from: https://www.ibm.com/analytics/cplex-optimizer.
[61] Beasley JE. OR-Library: distributing test problems by electronic mail. The Journal of the Operational Research Society. 1990;41(11):1069–1072.
[62] Hunsaker B, Tovey CA. Simple lifted cover inequalities and hard knapsack problems. Discrete Optim. 2005;2(3):219–228.