Improving Problem Reduction for 0–1 Multidimensional Knapsack Problems with Valid Inequalities

Hanyu Gu*

School of Mathematical and Physical Sciences, University of Technology Sydney
15 Broadway Ultimo NSW 2007, Australia

Abstract

This paper investigates the problem reduction heuristic for the Multidimensional Knapsack Problem (MKP). The MKP formulation is first strengthened by the Global Lifted Cover Inequalities (GLCI) using the cutting plane approach. The dynamic core problem heuristic is then applied to find good solutions. The GLCI is described in the general lifting framework and several variants are introduced. A two-level core problem heuristic is also proposed to tackle large instances. Computational experiments were carried out on classic benchmark problems to demonstrate the effectiveness of this new method.

Keywords: multidimensional knapsack problem, core problem, global lifted cover inequalities, heuristic algorithm

1. Introduction

The Multidimensional Knapsack Problem (MKP) is an extension of the classic Knapsack Problem (KP) with more than one knapsack constraints. Given m knapsacks with capacities b_i, $i = 1, \ldots, m$, and n items which require resource consumption of $a_{i,j}$ units in the i-th knapsack ($i = 1, \ldots, m$), and yield c_j units of profit upon inclusion for item j, $j = 1, \ldots, n$, the goal is to find a subset of items that yields maximum profit, denoted by z^*, without exceeding the knapsack capacities. The MKP can be defined by the following Integer Linear

*Corresponding author: +61 2 9514 2281
Email address: hanyu.gu@uts.edu.au (Hanyu Gu)

Programming (ILP):

$$\text{(MKP)} \qquad z^* = \max\{c^T x : Ax \leq b, x \in \{0,1\}^n\} \tag{1}$$

where $c = [c_1, c_2, \ldots, c_n]^T$ is an n-dimensional vector of profits, $x = [x_1, x_2, \ldots, x_n]^T$ is an n-dimensional vector of 0-1 decision variables indicating whether an item is included or not, $A = [a_{i,j}]$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ is an $m \times n$ coefficient matrix of resource requirements, and $b = [b_1, b_2, \ldots, b_m]^T$ is an m-dimensional vector of resource capacities. It is further assumed that all parameters are non-negative integers.

The MKP is a well-studied, strongly NP-hard combinatorial optimisation problem, and has found applications in many practical areas involving resource allocation. An early review of the MKP was given by [1], and a comprehensive overview of practical and theoretical results can be found in the monograph on knapsack problems [2]. Excellent reviews on solution methods and practical applications can be found in [3] [4]. In spite of the tremendous progress made by commercial ILP solvers, the methods currently yielding the best results, at least for commonly used benchmark instances in [5], are mainly from the specialised algorithms [6][7][8][9][10]. The main drawback of these approaches is, however, the huge running time for the large instances in the OR-Library [11].

Among the fast heuristics aiming for satisfactory solutions, the core problem based approach has been shown to be very competitive for its simplicity and efficiency. The core concept was first presented for the classical knapsack problem in [12], and extended later for MKP in [13]. The main idea is to reduce the original problem to a core of items for which it is hard to decide whether or not they will occur in an optimal solution, whereas all variables corresponding to items outside the core are fixed to their presumably optimal values. The core problem based heuristics typically determine an approximate core by calculating some simple *efficiency measures* for each variable. Various efficiency measures were proposed and compared in [14] in terms of core size and accuracy. It concluded that the core problem heuristic, especially with the efficiency measures exploiting the dual values of the Linear Programming (LP)

relaxation of MKP, can yield highly competitive results in significantly shorter run-times. Recently a new efficiency measure based on the reduced cost of the LP relaxation of MKP was proposed in [15]. Instead of fixed core sizes commonly used in previous literature, this novel approach can adaptively change core size for each instance. Comprehensive experimentation demonstrates that this approach performs consistently well on well-designed sets of test cases.

Since the LP relaxation plays an important role in the most successful core problem based heuristics for MKP, this inspires us to investigate in this paper the effectiveness of strengthening the LP relaxation of MKP with valid inequalities, in the hope for better performance on the hard instances. Although cuts are commonly used in the branch and cut algorithms for general ILP solvers, our application allows for classes of cuts with more expensive computational costs.

Lifted Cover Inequalities (LCI), among other valid inequalities for 0-1 knapsack polytopes, have proven useful when tackling hard 0-1 Integer Programming problems including the MKP [16][17]. Recently the Global LCI (GLCI) [18] was proposed for MKP to take into consideration multiple knapsack constraints simultaneously by solving LPs to lift the coefficients of a valid inequality. Although the GLCI may not even define a face of MKP, it can still be stronger than LCI, especially for MKP with many knapsack constraints. Nevertheless the GLCI was not evaluated for the effectiveness in the branch and bound algorithm. In this paper we apply the GLCI and its variants to MKP, and test the adaptive problem reduction heuristic in [15] on hard instances of MKP.

The paper is organised as follows. We first describe the problem reduction method introduced in [15] in Section 2. The rationale for applying valid inequalities is also discussed. The GLCI and some variants are described in the framework of general lifting principles [19] in Section 3. A two-level core problem heuristic is proposed in Section 4 to tackle large MKP instances. Computational results are presented in Section 5. The conclusion is given in Section 6.

2. Problem Reduction Heuristic Strengthened by Valid Inequalities

In the core problem reduction heuristic, an efficiency measure e is employed to rearrange the items into the order $(i_1, i_2, \ldots, i_n)$, so that

$$e(i_k) \geq e(i_{k+1}) \tag{2}$$

The items with higher efficiency values are regarded to be more likely included into the knapsacks, and the items with lower efficiency values are regarded to be more likely excluded from the knapsacks. An interval $[a_e, b_e]$ is therefore determined so that x_{i_k} is fixed to 1 if $i_k \in F_e^1 = \{i_k | 0 < k < a_e\}$, and x_{i_k} is fixed to 0 if $i_k \in F_e^0 = \{i_k | n \geq k > b_e\}$. The remaining undecided items $C_e = \{i_k | a_e \leq k \leq b_e\}$ form the reduced problem defined as

$$\begin{aligned} MKPC \quad \max \quad & z = \sum_{j \in C_e} c_j x_j + \tilde{z} \\ \text{s.t.} \quad & \sum_{j \in C_e} a_{ij} x_j \leq \tilde{b}_i, \quad i = 1, \ldots, m \\ & x_j \in \{0, 1\}, \quad j \in C_e \end{aligned}$$

with $\tilde{z} = \sum_{j \in F_e^1} c_j$ and $\tilde{b}_i = b_i - \sum_{j \in F_e^1} a_{ij}$, $i = 1, \ldots, m$.

The C_e with the smallest cardinality that leads to the optimal solution to the original MKP by solving the reduced MKPC is called the core, and the reduced MKPC is called the core problem accordingly [14].

The exact identification of the core problem requires solving the MKP to optimality. In practice only an approximate core is calculated to include hopefully the actual unknown core with high probability. The core size is a crucial parameter in most core problem heuristics, which is used to balance the accuracy of the approximate core and the computational effort required to solve the core problem. It is typically just a predefined constant in the core based heuristics [20]. Some empirical rules are also suggested in the literature which normally only depends on the number of items [14].

Recently a novel core based heuristic [15] is proposed which is based on the Lagrangian Relaxation (LR) method. The Lagrangian relaxation of MKP can

be written as

$$LR(\lambda) = \max\{(c - \lambda^T A)^T x + \lambda^T b : x \in \{0,1\}^n\}$$

where $\lambda \in \mathcal{R}_+^m$ are the Lagrangian multipliers associated with the relaxed knapsack constraints. $LR(\lambda)$ provides an upper bound for the MKP problem, and can be further strengthened by solving the Lagrangian dual problem

$$LD = \min_{\lambda} LR(\lambda) \tag{3}$$

which is a non-smooth convex optimisation problem, and can be solved by the subgradient algorithm [21].

Let $\bar{\lambda}$ be the optimal multipliers to LD. The modified profit of each item in $LR(\bar{\lambda})$ is

$$r_i = c_i - \bar{\lambda}^T a_i, \qquad i = 1, \ldots, n$$

where a_i is the i-th column of A. The set of optimal solutions of $LR(\bar{\lambda})$ is

$$S = \{x \in \{0,1\}^n : (2x_i - 1)r_i \geq 0,\ i = 1, \ldots, n\}$$

Based on the observation that x_i tends to be 1 in the optimal solution of MKP if the modified profit r_i takes large positive values, while it tends to be 0 if it takes large negative values, the efficiency measure is chosen as

$$e(i) = r_i, \qquad i = 1, \ldots, n \tag{4}$$

The approximate core is identified as follows. Let $r_{max} = \max\{|r_i| : i = 1, \ldots, n\}$. Given $\epsilon \in \mathcal{R}_+$, the core interval is defined as

$$a_e(\epsilon) = \max\{k | r_{i_k} \geq \epsilon r_{max}\} + 1, \quad b_e(\epsilon) = \min\{k | - r_{i_k} \geq \epsilon r_{max}\} - 1 \tag{5}$$

Therefore the set of variables fixed to 1 is

$$F_e^1(\epsilon) = \{i_k | 0 < k < a_e(\epsilon)\} = \{k | r_k \geq \epsilon r_{max}\}, \tag{6}$$

the set of core variables is

$$C_e(\epsilon) = \{i_k | a_e \leq k \leq b_e(\epsilon)\} = \{i : |r_i| < \epsilon r_{max}\}, \tag{7}$$

and the set of variables fixed to 0 is

$$F_e^0(\epsilon) = \{i_k | n \geq k > b_e\} = \{k | - r_k \geq \epsilon r_{max}\} \tag{8}$$

One unique feature of this core identification approach is that the core size is not pre-determined and can dynamically adapt to the characteristics of each instance. This Dynamically reduced Core Heuristic (DCH) has been comprehensively tested [15] on problems featuring varied coefficient correlation structures and constraint slackness levels. It was found that, by setting $\epsilon = 0.15$, DCH compared well with other problem reduction heuristics in terms of solution quality and estimated core problem sizes, and showed robust effectiveness as problem difficulty increased.

It is well-known [22] that the set of optimal multipliers of LD coincides with the set of optimal solutions of the dual of the LP relaxation of MKP

$$\text{(MKP-LP)} \qquad \bar{z} = \max\{c^T x : Ax \leq b, x \in [0,1]^n\} \tag{9}$$

Accordingly, the efficiency measure (4) is just the reduced cost of each item, which can be efficiently calculated even for large problems using LP solvers.

Since the LP relaxation can be weak for hard problems, ϵ may have to be large that results in a large core. Intuitively, if the LP can be strengthened by valid inequalities, the identification of the core may be more accurate. Here we will consider the ideal case that the polytope of MKP-LP, $P = \{x \in [0,1]^n : Ax \leq b\}$, is the complete description of the convex hull of the MKP polytope $P_I = Conv\{x \in \{0,1\}^n : Ax \leq b\}$.

Theorem 1. *If $P = P_I$, the core problem defined by DCH according to (6),(7),(8) solves the original MKP problem for any $\epsilon r_{max} > 0$.*

Proof. Given the optimal multipliers $\bar{\lambda}$, we can define, from the definition of $LR(\bar{\lambda})$ and LD, a valid inequality of $[0,1]^n$

$$(c - \bar{\lambda}^T A)^T x + \bar{\lambda}^T b \leq LD \tag{10}$$

Then we have

$$Conv(S) = \{x \in [0,1]^n | (c - \bar{\lambda}^T A)^T x + \bar{\lambda}^T b = LD\} \tag{11}$$

which shows that $Conv(S)$ is a face of $[0,1]^n$ defined by (10).

Let L be the face of P defined by the set of optimal LP solutions. Since $P = P_I$, all the extreme points of L are binary vectors. It is known from [23][24] that there exists $\hat{x} \in Conv(S) \cap L$. Therefore (10) also defines a proper face of L, which has the set of extreme points $O \subset S$. This means that at least one of the optimal solutions to $LR(\bar{\lambda})$ is the optimal solution of MKP. Since every member in S is a feasible solution to the core problem with $\epsilon r_{max} > 0$, the core problem solves the original MKP problem . □

Based on the insight from Theorem 1, we designed the Cut strengthened Core problem Heuristic (CCH) as detailed in Algorithm 1. First, the cutting plane approach [19] is employed to strengthen the LP relaxation. By solving the MKP-LP at the k-th iteration, an optimal continuous solution x^k is obtained. Then an attempt is made to generate a cut (valid inequality) that separates x^k from the convex hull of MKP polytope. If successful, the cut is added to MKP-LP and the process is repeated. The equivalence of optimisation and separation [25] implies that it is strongly NP-hard to derive a complete description of the 0-1 MKP polytope. In practice the cut generation is usually terminated when it enters the tailing-off phase. Hopefully, the strengthened LP can improve the accuracy of the core identification, which leads to good solutions of MKP.

Algorithm 1: Cut strengthened Core problem Heuristic (CCH)

Input: MKP, ϵ

Output: a feasible solution of MKP

```
k = 1;
repeat
    solve the cut strengthened MKP-LP and the optimal solution is x^k;
    generate cuts for x^k with separation algorithms;
    if no cut then break;
    add cuts to MKP;
until enough cuts or time limit reached ;
Identify the core problem for the strengthened MKP according to DCH using ε;
solve the reduced problem MKPC and return the solution for MKP;
```

3. Variants of Global LCI

In this section we describe the separation algorithm for GLCI [18], which has been shown to be effective for hard benchmark problems of MKP. Some variants of GLCI are also introduced to generate stronger valid inequalities. Similar to LCI, GLCI applies lifting procedures on the cover inequalities to generate valid inequalities of the form

$$\sum_{j\in C\setminus D} x_j + \sum_{j\in N\setminus C} \alpha_j x_j + \sum_{j\in D} \beta_j x_j \leq |C\setminus D| + \sum_{j\in D} \beta_j - 1, \tag{12}$$

where $C \subseteq N = \{1,\ldots,n\}$ is a cover of a single knapsack constraint in the MKP, α_j, $j \in N \setminus C$, is computed from the uplifting procedure, and β_j, $j \in D \subset C$, is computed from the downlifting procedure. C is a cover of the i-th knapsack constraint of MKP if $\sum_{j\in C} a_{ij} > b_i$. C is minimal if no subset of C is also a cover.

The details of the GLCI Separation Algorithm (GSA) are given in Algorithm 2. The items are sorted in the non-increasing order of their values in the current MKP-LP solution $\bar{x}$ in step 1. From step 2 to step 7 a minimal cover C is identified for the current row i along with the set of variables for downlifting D, and the sets of variables for uplifting, i.e., L_1, L_f and L_0. These steps are similar to the LCI separation heuristic in [16]. In step 8 the variables in $L_1 \cup L_f$ are uplifted sequentially according to the general uplifting procedure [19]. Let $S(U,V) = P_I \cap \{x \in \{0,1\}^n | x_j = 0, \forall j \in N\setminus(U\cup V), x_j = 1, \forall j \in V\}$.

Proposition 1. *Given $\sum_{j\in U} \pi_j x_j \leq \pi_0$ is valid for $S(U,V)$. For $i \in N\setminus(U\cup V)$, $\alpha_i x_i + \sum_{j\in U} \pi_j x_j \leq \pi_0$ is valid for $S(U\cup\{i\},V)$ for any α_i satisfying*

$$\begin{cases} \alpha_i \leq \pi_0 - \zeta & if\ S(U,V\cup\{i\}) \neq \emptyset \\ \alpha_i < \infty & otherwise \end{cases} \tag{13}$$

where

$$\zeta = \max\{\sum_{j\in U} \pi_j x_j | x_i = 1,\ x \in S(U\cup\{i\},V)\} \tag{14}$$

In step 8, starting from the valid cover inequality for $S(C \setminus D, D)$

$$\sum_{j \in C \setminus D} x_j \leq |C \setminus D| - 1, \tag{15}$$

the variables in $L_1 \cup L_f$ are sequentially uplifted to obtain a valid inequality for $S((C \setminus D) \cup L_1 \cup L_f, D)$, which is

$$\sum_{j \in C \setminus D} x_j + \sum_{j \in L_1 \cup L_f} \alpha_j x_j \leq |C \setminus D| - 1. \tag{16}$$

If the optimisation problem in equation (14) becomes infeasible when lifting x_j, we simply set $\alpha_j = |C \setminus D|$.

The violation of the GLCI inequality (12) can be calculated as

$$\begin{aligned} v &= \sum_{j \in C \setminus D} \bar{x}_j + \sum_{j \in L_1 \cup L_f \cup L_0} \alpha_j \bar{x}_j + \sum_{j \in D} \beta_j \bar{x}_j - |C \setminus D| - \sum_{j \in D} \beta_j + 1 \\ &= \sum_{j \in C \setminus D} \bar{x}_j + \sum_{j \in L_1 \cup L_f} \alpha_j \bar{x}_j - |C \setminus D| + 1 \end{aligned} \tag{17}$$

Since downlifting variables in D and uplifting variables in L_0 do not contribute in the violation, we can check if the GLCI inequality (12) will be a cut in step 9. If a cut can be produced, the variables in D are downlifted in step 10 according to the general downlifting procedure [19].

Proposition 2. *Given $\sum_{j \in U} \pi_j x_j \leq \pi_0$ is valid for $S(U, V)$. For $i \in V$, if $S(U \cup \{i\}, V \setminus \{i\}) \neq \emptyset$, then $\gamma_i x_i + \sum_{j \in U} \pi_j x_j \leq \pi_0 + \gamma_i$ is valid for $S(U \cup \{i\}, V \setminus \{i\})$ for any $\gamma_i \geq \zeta - \pi_0$, where*

$$\zeta = \max\{\sum_{j \in U} \pi_j x_j | x_i = 0,\, x \in S(U \cup \{i\}, V \setminus \{i\})\} \tag{18}$$

Starting from (16), the variables in D are sequentially downlifted resulting in a valid inequality for $S((C \cup L_1 \cup L_f, \emptyset)$, which is

$$\sum_{j \in C \setminus D} x_j + \sum_{j \in L_1 \cup L_f} \alpha_j x_j + \sum_{j \in D} \beta_j x_j \leq |C \setminus D| + \sum_{j \in D} \beta_j - 1. \tag{19}$$

Next, the variables in L_0 are uplifted according to Proposition 1 in step 11 to

derive the GLCI in (12).

Algorithm 2: General GLCI Separation Algorithm (GSA)

Input: Given the current LP solution $\bar{x}$ of the strengthened MKP

Output: cuts for MKP separating $\bar{x}$

1 sort the items in the order such that $\bar{x}_{s_k} \geq \bar{x}_{s_{k+1}}$, $k = 1, \ldots, n-1$;

2 **for** *i from 1 to m* **do**

3 find u that $\sum_{k=1}^{u} a_{i,s_k} > b_i$ and $\sum_{k=1}^{u-1} a_{i,s_k} \leq b_i$;

4 $C = \{s_1, \ldots, s_u\}$, $L_1 = \emptyset$;

5 **for** *k from u to* 1 **do**

6 **if** $\sum_{j \in C \setminus (L_1 \cup \{s_k\})} a_{i,j} > b_i$ **then** $L_1 = L_1 \cup \{s_k\}$;

7 let $C = C \setminus L_1$, $D = \{j | \bar{x}_j = 1,\ j \in C\}$, $L_f = \{j | 0 < \bar{x}_j < 1,\ j \notin C \cup L_1\}$, and $L_0 = \{j | \bar{x}_j = 0,\ j \in N\}$;

8 calculate uplifting coefficient α_j in (12) for $j \in L_1 \cup L_f$ sequentially in the non-increasing order of $\bar{x}_j$;

9 **if** *constraint violation* $v > 0$ *according to (17)* **then**

10 calculate downlifting coefficient β_j in (12) for $j \in D$;

11 calculate uplifting coefficient α_j in (12) for $j \in L_0$;

12 store the cut;

13 return stored cuts;

Calculating ζ in Proposition 1 and 2 exactly can be very expensive because a sequence of ILPs need to be solved. The original GLCI [18] instead takes the LP relaxation values of (14) and (18) to significantly reduce the computation time for lifting. We denote the GSA with this approximation scheme by GSA-LP.

Since our CCH can accommodate more computational burden for cut generation compared with the branch and bound approach, we can employ tighter approximation for the calculation of ζ. Based on the observation from (17) that only lifting on $L_1 \cup L_f$ may increase the violation of the GLCI cut, we can calculate the uplifting coefficients in step 8 of Algorithm 2 exactly, while approximating the other lifting coefficients with LP values. The GSA with this approximation scheme is denoted by GSA-IP.

Inspired by the extended cover inequality [26], we also consider the variant of GLCI that the lifted coefficients can be either 1 or 0.

Proposition 3. *Given $\sum_{j\in U}\pi_j x_j \le \pi_0$ is valid for $S(U,V)$, $\pi_j \in \mathcal{Z}$, $\forall j \in U$, $\pi_0 \in \mathcal{Z}$. For $i \in N \setminus (U \cup V)$, let*

$$B(U,V,i) = \{x \in \{0,1\}^n | x \in S(U, V \cup \{i\}), \sum_{j\in U} \pi_j x_j \ge \pi_0\} \tag{20}$$

If $S(U \cup \{i\}, V) \neq \emptyset$, then $\alpha_i x_i + \sum_{j\in U}\pi_j x_j \le \pi_0$ is valid for $S(U \cup \{i\}, V)$, where $\alpha_i = 1$ if $B(U,V,i) = \emptyset$, and 0 otherwise.

Similar to GSA-IP, we can approximate the uplifting coefficients in step 8 of Algorithm 2 by solving the feasibility problem (20), while approximating the other lifting coefficients with LP values. The GSA with this approximation scheme is denoted by GSA-UP1.

If the lifting on $j \in L_1 \cup L_f$ fails, i.e., $\alpha_j = 0$, we can record the reason of failure as the vector of excess capacities $u \in \mathcal{Z}^m$ which is

$$u_i = b_i - \sum_{j\in N} a_{ij} x^*_j, \quad i = 1, \ldots, m, \tag{21}$$

where x^* is the optimal solution to (14) in GSA-IP, or any feasible solution of (20) in GSA-UP1.

Proposition 4. *If lifting on $j \in L_1 \cup L_f$ fails in step 8 with the reason u, any subsequent lifting on $v \in L_1 \cup L_f$ will also fail if $a_v \le u$.*

When lifting $v \in L_1 \cup L_f$, we check a_v against all the stored reasons of failure for the current row of MKP.

4. Two-level Core Problem Heuristic

For large MKP problems the GLCI cuts are very weak in terms of closing the integrality gaps as can be seen in Section 5. The identified core problem is also too big to be solved efficiently by an ILP solver. Therefore we propose a Two-level Core problem Heuristic (TCH) to tackle these problems. At the first level the core problem is identified by DCH without adding any cuts, and is denoted by MKPC-1. We then apply CCH on MKPC-1 at the second level to

generate a good feasible solution of MKP. The size of the core problem at level two is further reduced at the risk of more non-optimal decisions. Hopefully the added cuts can help to improve the accuracy of the twicely reduced core. The details of TCH is given in Algorithm 3.

Algorithm 3: Two-level Core problem Heuristic (TCH)

Input: ϵ_1, ϵ_2, MKP

Output: A feasible solution of MKP

1 Identify the first level core problem MKPC-1 of MKP by DCH using ϵ_1 ;

2 solve MKPC-1 using CCH with ϵ_2, and the solution is $\hat{x}$;

3 combine the fixed decisions of MKP-1 with $\hat{x}$ and return it as a feasible solution of MKP;

5. Computational Results

We first compare our implementations of GLCI and its variants to evaluate the performance of the various valid inequalities and separation routine. The effects of GLCIs on the core reduction heuristics are investigated thereafter. All experiments are carried out on the widely used Chu and Beasley MKP test set in [11], and the Cho test set in [27]. The Chu and Beasley test set contains classes of randomly generated instances for each combination of $n \in \{100, 250, 500\}$ items, $m \in \{5, 10, 25\}$ constraints, and tightness ratios $\alpha \in \{0.25, 0.5, 0.75\}$. Ten different instances are available for each class, i.e., for each combination of n, m, and α. The Cho test set also contains classes of randomly generated instances for each combination of $n \in \{50, 100, 250\}$ items and $m \in \{5, 10, 30\}$ constraints. The Cho test set employed a systematic and explicit correlation induction problem generation scheme. It covers the full range of correlation values of the correlation structure of MKP, while the Chu and Beasley test set has a quite narrow correlation range which further reduces as the problem size increases [28].

All the algorithms were implemented in C++ and run on a cluster of 2.3GHz AMD Opteron(tm) Processor 6376. The reduced core problems are solved by

the IBM ILOG CPLEX Callable Library version 12.5 with a time limit of 500 seconds. The maximum number of threads for CPLEX is limited to 8.

5.1. GLCI and its variants

We compare the performance of GSA-LP, GSA-IP and GSA-UP1 in the cutting plane stage of CCH. The total number of cuts is limited to 100. The results for the Chu and Beasley test set are reported in Table 1 for test cases with 100 items, in Table 2 for test cases with 250 items, and in Table 3 for test cases with 500 items. The class of test cases is named in the format of $m.\alpha$. Each number in the table is an average over the 10 instances of the belonging class. The columns are explained as follows: z^* is the best known objective value for the MKP; $\bar{z}$ is the objective value of the LP relaxation; Δ_D is the relative integrality gap calculated as $100(\bar{z}-z^*)/\bar{z}\%$; I_K, I_G, I_{IP} and I_1 are the percentage of the integrality gap closed by the GSA-LP implemented in [17], the GSA-LP implemented in this paper, the GSA-IP and GSA-UP1; N_c is the number of cuts generated in total; T_v is the total time to generate the cuts in seconds, but is replaced with "-" if the time is smaller than 0.1 second.

In Table 1 the closed integrality gap by our implementation of GSA-LP, I_G, is comparable to I_K. The results of GSA-LP for $n = 250$, 500 are new. GSA-LP is comparable to GSA-IP and GSA-UP1 when $m = 5$, but becomes inferior in terms of closed integrality gap when the number of constraints increases.

GSA-UP1 performs similarly to GSA-IP for $m = 5$, 10, but has noticeable improvements for $m = 30$. For example, for the class 30.25 of 100 items, the closed integrality gap is improved from 6.49% of GSA-IP to 10.49% of GSA-UP1, which is much higher than the 2.74% of GSA-LP.

GSA-LP is the fastest and scales well to the largest instances. Surprisingly the GSA-IP and GSA-UP1 are also computationally efficient when $m = 5$, 10. However the computation times increase tremendously when $m = 30$.

GSA-IP and GSA-UP1 generate more cuts than GSA-LP when m increases, and the differences become significant when $m = 30$. GSA-UP1 generates more cuts than GSA-IP when $m = 30$, and this partially explains why GSA-UP1

performs the strongest for $m = 30$.

Table 1: Comparison of GLCIs on 100-item instances from Chu and Beasley test set [11]

					GSA-LP			GSA-IP			GSA-UP1		
$m.\alpha$	z^*	$\bar{z}$	Δ_D	I_K	I_G	N_c	T_v	I_{IP}	N_c	T_v	I_1	N_c	T_v
5.25	24197.2	24438.4	0.99	7.92	6.99	15.8	-	6.80	16.3	0.4	6.39	15	0.2
5.5	43252.9	43449.5	0.45	6.96	6.11	13	-	6.38	14.4	0.2	6.64	15.4	0.3
5.75	60470.9	60663.8	0.32	5.83	6.50	12.7	-	7.46	12.7	0.2	7.31	13.5	0.3
10.25	22601.9	22960.5	1.56	5.59	6.49	19	-	8.78	37.3	9.1	9.62	36.2	7.7
10.50	42660.6	43000.8	0.79	6.46	6.96	21.8	-	10.3	30	4.5	9.01	25	2.8
10.75	59555.4	59844.2	0.48	3.12	4.18	13.2	-	5.07	14.8	2	6.13	19	1.8
30.25	21660.4	22305.3	2.89	2.74	2.74	15.5	1.1	6.49	73.1	269	10.5	96.1	399
30.50	41440.4	41994.8	1.32	4.89	5.30	33.5	2.3	7.79	90.5	233	10.3	98.6	211
30.75	59201.8	59693.6	0.82	3.9	4.84	28.4	2.5	11.3	86.7	231	13.1	92.2	164

Table 2: Comparison of GLCIs on 250-item instances from Chu and Beasley test set [11]

				GSA-LP			GSA-IP			GSA-UP1		
$m.\alpha$	z^*	$\bar{z}$	Δ_D	I_G	N_c	T_v	I_{IP}	N_c	T_v	I_1	N_c	T_v
5.25	60413.5	60547.41	0.22	2.69	12	-	2.84	13	0.4	2.73	11.5	0.3
5.5	109292.8	109411.8	0.11	3.71	13.5	0.2	4.42	13.2	0.4	4.24	12.8	0.6
5.75	151560.3	151676.5	0.08	3.86	12.4	-	4.32	13.7	0.8	4.25	13.9	0.7
10.25	59021.6	59290.15	0.45	1.72	11.2	0.2	2.72	16.4	3.5	2.93	17.1	3.1
10.50	108729.3	108980.9	0.23	1.22	8.1	0.3	1.78	10.1	1.5	1.81	10	1.1
10.75	151346.2	151560.1	0.14	1.67	11.9	0.4	2.17	14.1	1.9	2.22	14.1	1.7
30.25	56940.6	57554.07	1.07	0.73	3.2	0.6	2.43	59.2	425	4.42	85.4	477
30.50	106712.9	107229.9	0.48	0.94	10.1	2.4	3.03	60.3	259	3.53	57.3	205
30.75	150482.5	150903.7	0.28	1.11	10.6	2.5	3.76	62.4	275	4.50	67.2	258

Table 3: Comparison of GLCIs on 500-item instances from Chu and Beasley test set [11]

				GSA-LP			GSA-IP			GSA-UP1		
$m.\alpha$	z^*	$\bar{z}$	Δ_D	I_G	N_c	T_v	I_{IP}	N_c	T_v	I_1	N_c	T_v
5.25	120630.3	120717	0.07	3.61	12.9	0.5	3.68	13.1	0.9	3.51	12.7	0.7
5.5	219512.7	219595.8	0.04	2.91	11.7	0.4	3.34	13.4	1.2	3.34	13.5	1.2
5.75	302360.9	302435	0.02	2.66	11.7	0.5	2.80	12.3	1	2.80	12.6	1
10.25	118608.8	118835.8	0.19	0.76	5.5	0.4	1.10	7.9	2	1.07	7.6	1.7
10.50	217321.7	217503.8	0.08	0.86	4.9	0.5	1.07	7.2	1.9	1.07	6.9	1.5
10.75	302596.7	302776	0.06	0.67	5.7	0.5	0.98	7.9	2	0.98	8.1	1.4
30.25	115584.8	116184.4	0.52	0.33	1.5	0.9	1.37	42.7	399	1.80	34.2	233
30.50	216237.5	216729.7	0.23	0.45	6.8	5.2	1.05	23.2	158	1.15	22.8	122
30.75	302423.3	302855.6	0.14	0.44	3.7	2.6	1.16	16	105	1.66	23.1	132

The results for the Cho test set are reported in Table 4. The closed integrality gaps are much higher than those of the Chu and Beasley test set. The three GLCIs all achieved over 20% closed gap even on the largest class with $n = 250$ and $m = 25$. This partly explains why the Cho test set is relatively easier to solve for CPLEX. GSA-IP outperforms GSA-LP on all the classes except when $n = 50$ and $m = 5$ in terms of closed integrality gaps, which is at the cost of much higher CPU time. GSA-UP1 even performs worse than GSA-LP on all the classes except when $n = 100$ and $m = 5$, which is very different from the observations from the Chu and Beasley test set. GSA-LP is still the fastest and scales well to the largest instances. Since the Cho test set has the most diverse correlation structures, it would be practical to use GSA-IP and GSA-UP1 only when GSA-LP fails.

Table 4: Comparison of GLCIs on the Cho test set [27]

				GSA-LP			GSA-IP			GSA-UP1		
$n.m$	z^*	$\bar{z}$	Δ_D	I_G	N_c	T_v	I_{IP}	N_c	T_v	I_1	N_c	T_v
50.5	1481.6	1494.6	0.91	51.9	13.0	-	49.5	12.9	0.2	47.8	10.7	0.3
50.10	1199.2	1214.9	1.32	50.2	19.9	0.1	53.4	22.6	1.3	42.5	15.1	0.4
50.25	968.7	984.6	1.63	46.9	29.8	0.2	49.6	37.9	7.4	43.8	25.8	3.1
100.5	2824.7	2830.8	0.22	41.8	9.2	-	44.2	10.0	0.5	42.0	9.7	0.3
100.10	2548.2	2558.8	0.42	33.3	15.8	-	33.5	18.6	1.9	31.5	13.5	0.6
100.25	1959.5	1972.3	0.64	34.1	23.6	0.2	37.4	30.0	8.2	33.4	19.0	2.4
250.5	7036.3	7040.2	0.05	41.2	9.1	0.1	42.3	9.3	0.7	38.4	9.3	0.4
250.10	6097.8	6103.2	0.09	26.2	12.5	0.2	28.2	13.2	2.2	25.1	13.0	1.1
250.25	4874.6	4882.7	0.16	22.0	17.7	0.6	23.9	22.2	17.5	20.8	17.5	5.0

5.2. Dynamically reduced Core Heuristics (DCH)

DCH has been tested on a comprehensive test suite [15], but the quality of the identified cores are not explicitly analysed for the Chu and Beasley test set since the core problems are solved heuristically. Here we compare the performance of DCH with different ϵ , and also compare with the widely cited Genetic Algorithm (GA) approach in [11]. The results are presented in Table 5. The class of test cases is named in the format of $n.m.\alpha$. Each number in the table is an average over the 10 instances of the belonging class. The columns are explained as follows: $\Delta_r = 100(\bar{z} - z_f)/\bar{z}\%$ is the average relative error where z_f is the objective value of the feasible solution found by the heuristics; Δ_a is the average difference between the objective value of the core reduction heuristic solution and the best known solution value; $|C_e|$ is the size of the reduced core; # is the number of times the optimum (or best known) was reached; T_c is the average CPU-time for solving the MKPC with a time limit of 500s.

For the smaller cases with $n = 100$, GA performs the best on all classes except for the class 100.30.25. Nevertheless the relative error values are all close. For the larger classes with $n = 250$ and 500, even DCH with $\epsilon = 0.1$ outperforms GA on 16 out of the 18 classes, and on a par for the remaining two classes although the core problem sizes are greatly reduced. With $\epsilon = 0.15$ DCH solves eight classes to optimality. This clearly shows the effectiveness of DCH in identifying high quality core problems of MKP, especially for large instances.

According to (5), the set of core items for $\epsilon = 0.1$ is a subset of the core items for $\epsilon = 0.15$. Therefore the solution values for $\epsilon = 0.15$ should be always larger than those for $\epsilon = 0.1$. However DCH achieved better solution values on most cases when $n = 500$ and $m \geq 10$. The reason is that the core problems for $\epsilon = 0.15$ are so big that CPLEX can only find inferior solutions within the 500s time limit.

5.3. Effects of GLCIs on the core reduction heuristic

Here we report the results of CCHs with different separation routines, i.e., the GSA-LP, GSA-IP and GSA-UP1 in Table 6 for the Chu and Beasley test

set. The ϵ is set to 0.1 in CCH so that we can compare the effects of GLCIs with the results of DCH in Table 5. The results for $n = 500$ are not reported since the added cuts are not helpful in most cases.

For the cases with $n = 100$ and $m = 5$, adding GLCI cuts significantly reduced the average absolute difference Δ_a for DCH with $\epsilon = 0.1$. The solution quality is even better than DCH with $\epsilon = 0.15$ for $\alpha \geq 0.5$, although the average core size is smaller.

For the cases with $n = 100$ and $m = 10$, CCH with GSA-IP and GSA-UP1 significantly improve on the average absolute difference Δ_a for DCH with $\epsilon = 0.1$. CCH with GSA-UP1 found optimal solutions for all the cases in $\alpha = 0.25$ which is much better than DCH with $\epsilon = 0.15$.

For the cases with $n = 100$ and $m = 30$, CCH with GSA-UP1 solved 29 cases to optimality out of the 30 test cases. Even CCH with GSA-LP found 4 more optimal solutions than DCH with $\epsilon = 0.15$, although the core sizes are similar.

CCH with GSA-LP performs the best for most of the classes with $n = 250$. It improved on the solution quality on all the test cases compared with DCH with $\epsilon = 0.1$, and found all the optimal solutions for $m = 10$. CCH with GSA-IP and GSA-UP1 have similar performance as CCH with GSA-LP on cases with $m = 5$, 10. This may be because the cuts from the different separating routines have similar strength, which leads to similar core sizes. The core sizes from CCH with GSA-IP and GSA-UP1 are much larger on the cases with $m = 30$. The inferior solution quality to CCH with GSA-LP may be due to the difficulty faced by the CPLEX solver for larger MKPC.

Table 5: Comparison of DCH for different ϵ on the Chu and Beasley test set[11]

	GA	$\epsilon = 0.1$					$\epsilon = 0.15$				
$n.m.\alpha$	Δ_r	Δ_r	Δ_a	$\lvert C_e \rvert$	#	T_c	Δ_r	Δ_a	$\lvert C_e \rvert$	#	T_c
100.5.25	0.99	1.07	19.1	20.7	6	0.2	1.01	5.3	28.5	8	0.6
100.5.50	0.45	0.50	21.8	18.9	4	0.1	0.46	5.8	26.5	7	0.2
100.5.75	0.32	0.36	23.8	19.6	6	0.1	0.33	9.4	27.3	8	0.2
100.10.25	1.56	1.73	38.4	27.1	4	0.9	1.56	0.0	36	10	2.9
100.10.50	0.79	0.91	50.0	24.7	4	0.2	0.84	20.3	30.3	7	0.7
100.10.75	0.48	0.50	13.1	26	7	0.2	0.49	6.9	33.9	9	1.1
100.30.25	2.91	2.98	20.0	37.3	7	7.9	2.90	3.0	45.2	9	34.7
100.30.50	1.34	1.42	42.4	33.9	2	4.1	1.36	18.0	41.1	5	15.3
100.30.75	0.83	0.86	19.1	34	6	2.8	0.85	15.9	40.9	6	7.3
250.5.25	0.23	0.22	2.2	49.5	8	28.3	0.22	0.0	72.4	10	25.6
250.5.50	0.12	0.11	2.7	46	7	19.5	0.11	0.0	64.1	10	25.3
250.5.75	0.08	0.08	0.8	46.4	9	8.2	0.08	0.0	68.5	10	11.5
250.10.25	0.51	0.47	7.9	55.9	8	335	0.45	0.0	74.4	10	449
250.10.50	0.25	0.23	2.1	49.8	8	126	0.23	0.0	70.7	10	444
250.10.75	0.15	0.14	4.6	55.5	9	178	0.14	0.0	76.4	9	330
250.30.25	1.19	1.13	36.0	68.6	3	461	1.08	8.9	86.3	5	502
250.30.50	0.53	0.49	12.9	62.8	2	467	0.48	2.6	82.3	4	502
250.30.75	0.31	0.29	12.3	65.1	6	485	0.29	9.8	85.8	5	502
500.5.25	0.09	0.07	0.3	110	9	408	0.07	1.1	156	9	417
500.5.50	0.04	0.04	0.0	82.6	10	350	0.04	0.0	123	10	368
500.5.75	0.03	0.02	0.6	90.3	9	237	0.02	0.0	133	10	208
500.10.25	0.24	0.20	38.1	108	0	502	0.20	41.7	152	0	502
500.10.50	0.11	0.08	7.6	85.3	5	502	0.09	28.1	123	2	502
500.10.75	0.07	0.06	14.3	94.2	4	502	0.07	29.4	140	0	502
500.30.25	0.61	0.54	76.6	121	2	502	0.58	119.	166	0	502
500.30.50	0.26	0.23	55.1	97.2	2	502	0.24	68.1	134	0	502
500.30.75	0.17	0.15	46.1	113	1	502	0.15	50.0	157	0	502

Table 6: Comparison of CCHs with $\epsilon = 0.1$ on the Chu and Beasley test set [11]

	GSA-LP				GSA-IP				GSA-UP1			
$n.m.\alpha$	Δ_a	$\lvert C_e\rvert$	#	T_c	Δ_a	$\lvert C_e\rvert$	#	T_c	Δ_a	$\lvert C_e\rvert$	#	T_c
100.5.25	6.8	27	8	0.6	6.8	27.8	8	0.6	6.8	27.5	8	0.4
100.5.50	3.9	24.2	7	0.3	3.9	25.4	7	0.2	3.9	25.6	7	0.2
100.5.75	6.2	25.7	9	0.1	6.2	26.4	9	0.2	6.2	26.7	9	0.1
100.10.25	8.4	33.9	8	2	5	38.2	9	3.1	0	37.8	10	3.3
100.10.50	6.6	30.9	8	1.3	6.6	33.9	8	1.6	6.6	32.8	8	1.2
100.10.75	15.7	30.5	8	0.5	7.2	31.7	8	1	7.2	32.7	8	1
100.30.25	0	43.2	10	21.9	0	49.3	10	48.3	0	55.3	10	85
100.30.50	7.4	45.4	8	25.8	1.7	52	9	50.5	1.7	57.6	9	67
100.30.75	19.1	40.3	6	6.9	3	50.8	9	20.4	0	53.5	10	20.3
250.5.25	1.8	52.9	9	31.6	1.8	53.5	9	31.9	1.8	53.5	9	29.1
250.5.50	1.9	51.9	8	23.7	1.9	51.1	8	24	1.9	51.1	8	20.8
250.5.75	0.6	51.1	9	10.6	0.6	51.8	9	12.2	0.6	51.8	9	9.9
250.10.25	0	60.1	10	374	0	62.2	10	376	0.3	62.9	9	362
250.10.50	0	53.1	10	204	2.1	55	8	252	2.1	54.9	8	241
250.10.75	0	58.8	10	211	0	60.3	10	253	0	59.7	10	228
250.30.25	17.4	71.9	6	470	27.2	81.4	4	502	22.6	91.5	5	502
250.30.50	9.7	69.7	2	502	13.4	79.8	3	502	5.3	83	4	502
250.30.75	8.3	71.1	7	502	2.2	80	8	502	16.5	85.2	3	502

The results for the Cho test set are shown in Table 7. The average CPU time to solve the core problems of all the classes is no more than one second. The relative error in Table 7 is $\Delta_r = 100(z^* - z_f)/z^*\%$ as in [15]. As expected, DCH with $\epsilon = 0.15$ outperforms DCH with $\epsilon = 0.1$ and solves 19 more instances to optimality. However, by adding GLCI cuts, even CCH with GSA-LP obtained average relative errors comparable with DCH with $\epsilon = 0.15$. CCH with GSA-IP can solve 3 more instances to optimality. The CCHs achieved the performance improvement with significantly smaller core sizes on all the large instances with $n = 250$ than DCH with $\epsilon = 0.15$.

Table 7: Comparison of CCHs with $\epsilon = 0.1$ on the Cho test set [27]

	DCH $\epsilon = 0.1$			DCH $\epsilon = 0.15$			GSA-LP			GSA-IP		
$n.m$	Δ_r	#	$\lvert C_e \rvert$	Δ_r	#	$\lvert C_e \rvert$	Δ_r	#	$\lvert C_e \rvert$	Δ_r	#	$\lvert C_e \rvert$
50.5	0.03	24	9.8	0.01	28	13.8	0.02	28	16.7	0.0	29	16.7
50.10	0.27	21	11.3	0.08	26	15.4	0.02	27	18.4	0.01	28	19.8
50.25	0.10	23	12.2	0.07	26	16.6	0.01	28	21.0	0.01	28	22.5
100.5	0.0	28	22.7	0.0	30	32.7	0.0	27	29.1	0.0	28	29.4
100.10	0.05	27	21.8	0.0	30	31.6	0.0	30	29.1	0.0	30	29.9
100.25	0.01	26	23.7	0.0	28	34.3	0.0	29	34.1	0.0	29	35.6
250.5	0.0	30	54.1	0.0	30	77.0	0.0	29	59.7	0.0	29	60.1
250.10	0.0	30	55.7	0.0	30	80.5	0.0	30	62.2	0.0	30	62.6
250.25	0.0	30	58.2	0.0	30	84.3	0.0	30	67.0	0.0	30	68.1
Total		239			258			258			261	

5.4. Two-level Core Problem Heuristic

The effects of GLCIs diminish in CCH as the problem size increases, and the core problem itself becomes big and time-consuming to solve. In the following tests on TCH, we use $\epsilon = 0.2$ for the first level, and use $\epsilon = 0.15$ for the second level on the Chu and Beasley test set. In the second level, we compare the performance of DCH with CCHs and report the results in Table 8. All the numbers corresponding to the smallest average absolute difference Δ_a are highlighted in bold.

In the second level of TCH, CCHs outperform DCH on all cases in terms of solution quality. GSA-UP1 performs best on 8 classes, GSA-IP performs best on another 8 classes, and the three CCHs have a draw on the remaining 2 classes. GSA-UP1 performs no worse than GSP-LP on all cases, and the biggest improvements from GSA-UP1 are about 47 on the classes 250.30.25 and 250.30.50 compared with GSP-LP.

The improvements from CCHs come at the cost of longer solution time for solving MKPC, however, all TCHs are still efficient when $m = 5, 10$. TCHs with GSA-IP and GSA-UP1 normally take more time than TCH with GSA-LP due to larger core sizes.

The solution quality from TCH deteriorates on most cases compared with the results of DCH in Table 5. However TCH is much faster, in many cases more than 10 times faster, than DCH. TCH even achieved better results on the two difficult classes 500.10.25 and 500.30.75.

Table 8: Comparison of Core Problem Heuristics at the second level of TCH on the Chu and Beasley test set [11]

		TCH-DCH			TCH-GSA-LP			TCH-GSA-IP			TCH-GSA-UP1		
$n.m.\alpha$	$\bar{z}^*$	Δ_a	$\lvert C_e\rvert$	T_c	Δ_a	$\lvert C_e\rvert$	T_c	Δ_a	$\lvert C_e\rvert$	T_c	Δ_a	$\lvert C_e\rvert$	T_c
250.5.25	60414	87.2	18.9	0.4	68.6	22.2	0.7	**63.4**	22.3	0.6	64.9	22	0.7
250.5.50	109293	119	17.1	0	45.4	22.7	0.4	49.3	22.8	0.3	**42.7**	24.1	0.4
250.5.75	151560	78.9	16.9	0	27.2	22.9	0.7	16.1	24.5	0.9	**15.9**	24.6	1.2
250.10.25	59022	123	24.3	0.4	69.3	30.3	2	**37.7**	34.3	16.4	40.5	34.4	9.7
250.10.50	108729	141	20.5	0.1	88.5	25.5	0.5	58.2	30.2	3.2	**48.7**	30.8	5.9
250.10.75	151346	70.5	24.5	0.8	32.6	30.8	2.4	**18.3**	35.2	8.5	23.2	34	6.5
250.30.25	56948	114	39.4	30.3	84.8	43.5	75.9	42.8	53	240	**37.5**	64.5	502
250.30.50	106721	99.7	34.9	6.6	71.9	43	45.8	27.6	54.2	378	**25.2**	63.1	466
250.30.75	150485	61.6	37.2	7.1	30.2	43.5	55	18.3	53.2	260	**15.8**	60.4	390
500.5.25	120630	11.1	38.9	86.7	**3.4**	45.4	194	**3.4**	47.3	229	**3.4**	45.5	187
500.5.50	219513	24.3	28.3	5.8	14.4	32.6	30.2	16.6	33.8	37.1	**8.5**	34.7	40.6
500.5.75	302363	14	30.8	19.9	**5.6**	35.9	36	**5.6**	35.9	39.1	**5.6**	36.6	39.7
500.10.25	118639	68.2	38.6	84.4	38.6	42.7	174	**34.5**	45.1	204	34.6	44.8	202
500.10.50	217334	80.3	35.2	54.7	40.3	39.3	70.6	**26.7**	41.9	177	28.9	41.6	176
500.10.75	302606	58.8	34.4	12.2	37.7	37.4	23.7	**34.4**	39.7	73.6	34.8	38.6	38.3
500.30.25	115635	100	56.3	365	97.3	61.3	432	**80.7**	73.8	502	87.5	81.2	502
500.30.50	216278	122	48.2	167	91.4	55.1	347	70.2	62.3	451	**58.4**	66.8	479
500.30.75	302447	70.5	52.4	375	44.3	57.6	398	**31.8**	67.9	502	36.9	75.6	502

6. Conclusion

In this paper we investigated how the GLCI cuts can improve on the core problem heuristic for the multidimensional knapsack problem. The GLCIs are presented in the general lifting framework and several variants are proposed. A two-level core problem heuristic is also proposed to tackle very large instances of MKP. Experiments on the classic benchmark problems show that adding GLCI cuts can significantly improve the performance and robustness of the core problem heuristic.

References

[1] A. Fréville, The multidimensional 0 - 1 knapsack problem: An overview, European Journal of Operational Research 155 (1) (2004) 1–21.

[2] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack problems, Springer, 2004.

[3] A. Fréville, S. Hanafi, The multidimensional 0-1 knapsack problem - bounds and computational aspects, Annals of Operations Research 139 (1) (2005) 195–227.

[4] C. Wilbaut, S. Hanafi, S. Salhi, A survey of effective heuristics and their application to a variety of knapsack problems, IMA Journal of Management Mathematics 19 (3) (2008) 227–244.

[5] J. Drake, Or library mkp - best known solutions.
URL `http://www.cs.nott.ac.uk/~jqd/mkp/bestresults.html`

[6] Y. Vimont, S. Boussier, M. Vasquez, Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem, Journal of Combinatorial Optimization 15 (2) (2008) 165–178.

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

[8] C. Wilbaut, S. Hanafi, New convergent heuristics for 0–1 mixed integer programming, European Journal of Operational Research 195 (1) (2009) 62–74.

[9] F. Della Croce, A. Grosso, Improved core problem based heuristics for the 0/1 multi-dimensional knapsack problem, Computers & Operations Research 39 (1) (2012) 27–31.

[10] M. Vasquez, Y. Vimont, Improved results on the 0-1 multidimensional knapsack problem., European Journal of Operational Research 165 (1) (2005) 70–81.

[11] P. C. Chu, J. E. Beasley, A genetic algorithm for the multidimensional knapsack problem, J. Heuristics 4 (1) (1998) 63–86.

[12] E. Balas, E. Zemel, An algorithm for large zero-one knapsack problems, Oper. Res. 28 (1980) 1130–1154.

[13] J. Puchinger, G. R. Raidl, U. Pferschy, The core concept for the multidimensional knapsack problem, in: J. Gottlieb, G. R. Raidl (Eds.), Evolutionary Computation in Combinatorial Optimization, Vol. 3906 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 195–208.

[14] J. Puchinger, G. R. Raidl, U. Pferschy, The multidimensional knapsack problem: Structure and algorithms, INFORMS Journal on Computing 22 (2) (2010) 250–265.

[15] R. R. Hill, Y. Kun Cho, J. T. Moore, Problem reduction heuristic for the 0-1 multidimensional knapsack problem, Comput. Oper. Res. 39 (1) (2012) 19–26. doi:10.1016/j.cor.2010.06.009.
URL http://dx.doi.org/10.1016/j.cor.2010.06.009

[16] Z. Gu, G. L. Nemhauser, M. W. P. Savelsbergh, Lifted cover inequalities for 0-1 integer programs: Computation, INFORMS Journal on Computing 10 (4) (1998) 427437.

[17] K. Kaparis, A. N. Letchford, Separation algorithms for 0-1 knapsack polytopes, Mathematical Programming 124 (1-2) (2010) 69–91. doi:10.1007/s10107-010-0359-5.
URL http://dx.doi.org/10.1007/s10107-010-0359-5

[18] K. Kaparis, A. N. Letchford, Local and global lifted cover inequalities for the 01 multidimensional knapsack problem, European Journal of Operational Research 186 (1) (2008) 91–103.

[19] L. A. Wolsey, G. L. Nemhauser, Integer and Combinatorial Optimization, Wiley Series in Discrete Mathematics and Optimization, Wiley, 1999.

[20] S. Martello, P. Toth, A new algorithm for the 0-1 knapsack problem, Management Science 34 (1988) 633644.

[21] M. Fisher, The lagrangian relaxation method for solving integer programming problems, Management Science 27 (1981) 1–18.

[22] A. Geoffrion, Lagrangian relaxation for integer programming, Mathematical Programming Study 2 (1974) 82–114.

[23] C. Lemaréchal, A. Renaud, A geometric study of duality gaps, with applications, Mathematical Programming 90 (2001) 399–427.

[24] A. Frangioni, About lagrangian methods in integer optimization, Annals of Operations Research 139 (2005) 163–193,.

[25] M. Grötschel, L. Lovász, A. Schrijver, Geometric Algorithms and Combinatorial Optimization, Vol. 2 of Algorithms and Combinatorics, Springer, 1988.

[26] E. Balas, Facets of the knapsack polytope, Mathematical Programming 8 (1975) 146–164.

[27] Y. K. Cho, J. T. Moore, R. R. Hill, C. H. Reilly, Exploiting empirical knowledge for bi-dimensional knapsack problem heuristics, International Journal of Industrial and Systems Engineering 3 (5) (2008) 530–548.

[28] R. R. Hill, J. T. Moore, C. Hiremath, Y. K. Cho, Test problem generation of binary knapsack problem variants and the implications of their use, International Journal of Operations and Quantitative Management 18 (2) (2011) 105–128.