Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 186 (2008) 91–103
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

Local and global lifted cover inequalities for the 0–1 multidimensional knapsack problem

Konstantinos Kaparis, Adam N. Letchford *

Department of Management Science, Lancaster University, Lancaster LA1 4YW, United Kingdom

Received 17 May 2006; accepted 28 January 2007
Available online 20 February 2007

Abstract

The 0–1 multidimensional knapsack problem (0–1 MKP) is a well-known (and strongly $\mathcal{NP}$-hard) combinatorial optimization problem with many applications. Up to now, the majority of upper bounding techniques for the 0–1 MKP have been based on Lagrangian or surrogate relaxation. We show that good upper bounds can be obtained by a cutting plane method based on lifted cover inequalities (LCIs). As well as using traditional LCIs, we use some new 'global' LCIs, which take the whole constraint matrix into account.

Keywords: Integer programming; Combinatorial optimization

1. Introduction

A *0–1 Multidimensional Knapsack Problem* (0–1 MKP) is a problem of the form:

$$\max\{c^{\mathrm{T}}x : Ax \leqslant b, x \in \{0,1\}^n\},$$

where $c \in \mathbb{Z}_+^n$ is the objective function vector, $A \in \mathbb{Z}_+^{m\times n}$ is the matrix of constraint coefficients, and $b \in \mathbb{Z}_+^m$ is the vector of right-hand sides. In other words, the 0–1 MKP is the special case of integer programming in which all variables are binary, all constraints are of 'less-than-or-equal-to' type, and all objective and constraint coefficients are positive integers.

There is a vast literature on the 0–1 MKP. For detailed surveys on applications, complexity results, approximation algorithms, heuristics, upper bounds and exact algorithms, we refer the reader to Fréville [10], Kellerer et al. [23] and Fréville and Hanafi [11]. We mention only that the 0–1 MKP is strongly $\mathcal{NP}$-hard [13], though, if the number of constraints is bounded by a constant, it can be solved in pseudo-polynomial time by dynamic programming. Current exact methods can run into difficulties even for instances with $n \leqslant 200$ and $m \leqslant 5$, although larger instances can be solved if they have special structure.

* Corresponding author. Tel.: +44 1524 594719; fax: +44 1524 844885.
E-mail address: A.N.Letchford@lancaster.ac.uk (A.N. Letchford).

0377-2217/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2007.01.032

This paper is concerned with *upper bounds*, which are at the heart of all exact solution methods. Up to now, research on upper bounds has tended to concentrate on Lagrangian, surrogate (and composite) relaxations. We will show that, in fact, very good upper bounds can be obtained with a cutting plane approach. As potential cutting planes we consider *lifted cover inequalities* (LCIs), which are already well known in the integer programming literature. However, we also propose a stronger class of inequalities, which we call 'global' LCIs, which take the whole constraint matrix into account.

Surprisingly, despite the fact that LCIs are known to be effective cutting planes, and are even included in many commercial integer programming packages, we are only aware of two works in which they are applied to the 0–1 MKP. Gabrel and Minoux [12] performed experiments with a very restricted form of LCIs, the so-called *extended* cover inequalities, and Bektas and Oguz [4] used only cover inequalities, with no lifting at all.

The rest of the paper is structured as follows. In Section 2, we review the literature on LCIs and show how they relate to the 0–1 MKP. In Section 3, we introduce the global LCIs and explain how lifting can be performed effectively. Then, in Section 4, we consider the *separation problem* associated with the various inequalities, i.e., the problem of detecting when an inequality in a given class is violated by the solution of a given LP relaxation. In Section 5, we present extensive computational results using the standard OR-LIB instances. Finally, some concluding remarks are given in Section 6.

Throughout the paper, we use the following notation and terminology. The feasible region of the linear programming relaxation of the problem will be denoted by

$$\mathscr{P} := \{x \in [0,1]^n : Ax \leqslant b\},$$

where n, A and b are assumed to be fixed throughout the paper. The convex hull of feasible integer solutions will be denoted by

$$\mathscr{P}_I := \mathrm{conv}\{x \in \{0,1\}^n : Ax \leqslant b\}.$$

We will assume that the ith knapsack constraint, i.e., the ith inequality in the system $Ax \leqslant b$, takes the form

$$\sum_{j=1}^{n} a_{ij}x_j \leqslant b_i.$$

With the ith constraint we associate the 0–1 *knapsack polytope*

$$\mathscr{Q}_i := \mathrm{conv}\left\{x \in \{0,1\}^n : \sum_{j=1}^{n} a_{ij}x_j \leqslant b_i\right\}.$$

Then, we have

$$\mathscr{P}_I \subseteq \bigcap_{i=1}^{m} \mathscr{Q}_i \subseteq \mathscr{P}$$

and for most problems of practical interest both containments are strict.

2. Lifted cover inequalities

2.1. Motivation

In theory, it is possible to optimise a linear function over $\bigcap_{i=1}^{m} \mathscr{Q}_i$ in pseudo-polynomial time. This can be done, for example, by *Lagrangian decomposition* [17], or by using the so-called *Fenchel cuts* [5]. However, for large n, these methods are far too time-consuming. A more practical method, originally suggested by Crowder et al. [8] in the context of general 0–1 integer programs, is to *approximately* optimise over $\bigcap_{i=1}^{m} \mathscr{Q}_i$ via the following cutting plane method:

1. Solve the initial LP relaxation by primal simplex.
2. Let x^* be the solution vector. If x^* is integer, output the optimal solution and stop.
3. For each i, attempt to find inequalities which are valid for $\mathscr{Q}_i$ and violated by x^*.

4. If no such inequalities are found, output the final upper bound and stop.
5. Add the violated inequalities to the LP as cutting planes.
6. Re-optimise the LP by dual simplex and go to step 2.

This method relies on having knowledge about valid inequalities for 0–1 knapsack polytopes. Crowder et al. found that one particular class of valid inequalities, the lifted cover inequalities, were very effective. These inequalities were first discovered independently in 1975 by Balas [2] and Wolsey [24], and have since been studied and generalised by many authors (see, e.g., [1]). Gu et al. [14] obtained particularly good computational results applying LCIs to general 0–1 integer programs, and we will be using their algorithmic framework as a 'benchmark'.

2.2. Definition

Consider a 0–1 knapsack polytope of the form

$$\mathcal{Q} := \text{conv}\{x \in \{0,1\}^n : a^T x \leqslant b\},$$

where in this subsection and the next we drop the i subscript. We say that the set $C \subseteq N = \{1,\ldots,n\}$ is a *cover* if it satisfies $\sum_{j\in C} a_j > b$. Given any cover C, the *cover inequality* $\sum_{j\in C} x_j \leqslant |C| - 1$ is clearly valid for $\mathcal{Q}$. Moreover, the strongest cover inequalities are obtained when the cover C is *minimal*, in the sense that no proper subset of C is also a cover.

In general, even minimal cover inequalities do not induce facets of $\mathcal{Q}$. To make them facet-inducing, one must compute appropriate left-hand side coefficients for the variables in $N \setminus C$, a process called *lifting*. The resulting *lifted* cover inequalities (LCIs) take the general form:

$$\sum_{j\in C} x_j + \sum_{j\in N\setminus C} \alpha_j x_j \leqslant |C| - 1,$$

where the lifting coefficients α_j satisfy $0 \leqslant \alpha_j \leqslant |C| - 1$. Normally, lifting is done *sequentially*, i.e., one variable at a time.

Example. Suppose the knapsack constraint is

$$x_1 + x_2 + 2x_3 + 2x_4 + 3x_5 \leqslant 3.$$

Since $\{1,2,3\}$ forms a minimal cover, the cover inequality $x_1 + x_2 + x_3 \leqslant 2$ is valid. Now, to lift x_4, we can solve the 0–1 knapsack problem

$$\max\left\{\sum_{j=1}^{3} x_j : x_1 + x_2 + 2x_3 + 2x_4 \leqslant 3, x_4 = 1, x \in \{0,1\}^4\right\}.$$

Since the optimum to this is 1, x_4 receives a lifting coefficient of $2 - 1 = 1$, giving the lifted inequality $x_1 + x_2 + x_3 + x_4 \leqslant 2$. Next, to lift x_5, we can solve the 0–1 knapsack problem

$$\max\left\{\sum_{j=1}^{4} x_j : x_1 + x_2 + 2x_3 + 2x_4 + 3x_5 \leqslant 3, x_5 = 1, x \in \{0,1\}^5\right\}.$$

Since the optimum to this is 0, x_5 receives a lifting coefficient of $2 - 0 = 2$, giving the LCI $x_1 + x_2 + x_3 + x_4 + 2x_5 \leqslant 2$.

In general, different *lifting sequences* may give rise to different LCIs. Suppose, for example, that the knapsack constraint is

$$7x_1 + 6x_2 + 4x_3 + 5x_4 + 5x_5 \leqslant 14,$$

and the initial cover C is $\{1,2,3\}$. One can check that lifting x_4 before x_5 yields the LCI $x_1 + x_2 + x_3 + x_4 \leqslant 2$, whereas lifting in the reverse order yields the LCI $x_1 + x_2 + x_3 + x_5 \leqslant 2$.

In fact, it is not necessary to solve a sequence of 0–1 knapsack problems to perform lifting. Several authors have presented results which make lifting much easier. First, Balas and Zemel [3] showed how to compute, in linear time, upper and lower bounds for the lifting coefficients which differ by at most one. For some variables, the upper and lower bounds coincide and the lifting coefficient is therefore immediately determined. Second, Zemel [26] presented a dynamic programming algorithm to compute exact lifting coefficients for the remaining variables. The algorithm runs in $\mathcal{O}(|C||N \setminus C|)$ time, and is very fast in practice. Finally, Gu et al. [16] showed how to strengthen the Balas–Zemel bounds without significantly increasing the time taken to compute them. As a result, even more lifting coefficients can be quickly fixed, leaving less work for Zemel's algorithm to do. We make use of all three of these developments in our cutting plane algorithm.

2.3. Down-lifting

Lifting can be viewed in the following way: we first take the face of $\mathcal{Q}$ defined by the equations $x_j = 0$ for all $j \in N \setminus C$. The cover inequality $\sum_{j \in C} x_j \leqslant |C| - 1$ induces a facet of this face. We then rotate this cover inequality to make it a facet of the original knapsack polytope. As noted by Wolsey and others [14,21], an analogous procedure can be performed with faces defined by equations of the form $x_j = 1$. More precisely, for any cover C and any subset $D \subset C$, the inequality

$$\sum_{j \in C \setminus D} x_j \leqslant |C \setminus D| - 1$$

induces a facet of the restricted polytope

$$\operatorname{conv}\left\{ x \in \{0,1\}^{|C \setminus D|} : \sum_{j \in C \setminus D} a_j x_j \leqslant b - \sum_{j \in D} a_j \right\}.$$

Standard sequential lifting then yields an inequality of the form

$$\sum_{j \in C \setminus D} x_j + \sum_{j \in N \setminus C} \alpha_j x_j \leqslant |C \setminus D| - 1,$$

which induces a facet of the polytope

$$\operatorname{conv}\left\{ x \in \{0,1\}^{|N \setminus D|} : \sum_{j \in N \setminus D} a_j x_j \leqslant b - \sum_{j \in D} a_j \right\}.$$

Finally, this can be lifted to obtain a facet of $\mathcal{Q}$ of the form

$$\sum_{j \in C \setminus D} x_j + \sum_{j \in N \setminus C} \alpha_j x_j + \sum_{j \in D} \beta_j x_j \leqslant |C \setminus D| + \sum_{j \in D} \beta_j - 1.$$

The process of computing coefficients for the variables fixed at 1 (i.e., the variables in D) is sometimes called *down-lifting*. The computation of coefficients for the variables fixed at 0 (i.e., the variables in $N \setminus C$) is sometimes referred to as 'up-lifting'.

Example. Suppose the knapsack constraint is $x_1 + 2x_2 + 2x_3 + 3x_4 \leqslant 5$, and we choose $C = \{1,2,4\}$ and $D = \{4\}$. The cover inequality $x_1 + x_2 \leqslant 1$ is (trivially) valid for the restricted polytope

$$\operatorname{conv}\{x \in \{0,1\}^2 : x_1 + 2x_2 \leqslant 2\}.$$

Up-lifting x_3 then yields the inequality $x_1 + x_2 + x_3 \leqslant 1$. Now, to down-lift x_4, we solve the 0–1 knapsack problem

$$\max\left\{ \sum_{j=1}^{3} x_j : x_1 + 2x_2 + 2x_3 + 3x_4 \leqslant 5, x_4 = 0, x \in \{0,1\}^4 \right\}.$$

Since the optimum to this is 3, x_4 receives a lifting coefficient of $3 - 1 = 2$. The resulting LCI, $x_1 + x_2 + x_3 + 2x_4 \leqslant 3$, induces a facet of the original knapsack polytope.

As this example illustrates, the use of down-lifting enables one to construct LCIs which cannot be obtained using up-lifting alone. Computational results in Gu et al. [14] show that using down-lifting as well as up-lifting leads to a much more effective cutting plane algorithm. We will follow other authors in calling an LCI 'simple' if it has been derived without down-lifting.

A word of caution, however, must be made about down-lifting. If we choose a set $D \subset C$ such that $\max_{j \in N \setminus D} a_j > b - \sum_{j \in D} a_j$, then it will not be possible to perform up-lifting before down-lifting. One solution to this difficulty is to remove one or more items from the set D.

Example. Suppose the knapsack constraint is $x_1 + x_2 + 2x_3 + 3x_4 \leqslant 4$, and we choose $C = \{1,2,4\}$ and $D = \{4\}$. The cover inequality $x_1 + x_2 \leqslant 1$ is (trivially) valid for the restricted polytope

$$\text{conv}\{x \in \{0,1\}^2 : x_1 + x_2 \leqslant 1\}.$$

If we now want to up-lift x_3, we need to solve

$$\max\{x_1 + x_2 : x_1 + x_2 + 2x_3 \leqslant 1, x_3 = 1, x \in \{0,1\}^3\}.$$

But this problem is infeasible. To avoid this difficulty, then, we could set $D = \emptyset$ instead. The resulting (simple) LCI would then be $x_1 + x_2 + x_4 \leqslant 2$.

As in the case of up-lifting, it is not actually necessary to solve a sequence of 0–1 knapsack problems to perform down-lifting. Gu et al. [16] claim that Zemel's algorithm can be adapted to compute all down-lifting coefficients in $\mathcal{O}(|C|n^3)$ time. Although this is polynomial, it is time-consuming in practice. A faster and simpler alternative for down-lifting is to solve the LP relaxation of the auxiliary 0–1 knapsack instances, which takes only $\mathcal{O}(n^2)$ time in total. Of course, one should round down the optimal value of each lifting LP to the nearest integer, both to make the LCI as strong as possible, and to avoid having fractional lifting coefficients.

3. Global lifted cover inequalities

3.1. Motivation

As mentioned above, the idea of using facets of the knapsack polytope to tackle more complex 0–1 integer programs was already present in Crowder et al. [8]. They argued that, *provided* the constraint matrix A is sparse, the intersection of the individual $\mathscr{Q}_i$ should give a reasonable approximation to $\mathscr{P}_I$ itself. They also gave convincing empirical evidence that this was so.

In most instances of the 0–1 MKP, however, the constraint matrix A is *dense*. In this situation, we cannot expect that valid inequalities derived from individual rows will always be useful, and it seems more sensible to attempt to derive valid inequalities which somehow take into account the global structure of the problem. We can of course resort to general-purpose cutting planes such as Gomory cuts, but these perform poorly for the 0–1 MKP [19]. Inequalities of a more 'combinatorial' nature seem to be needed. Some explorations in this direction have been performed, for example, by Martin and Weismantel [20]. Here, we consider the most straightforward generalization of the LCIs.

3.2. Definition

The 0–1 MKP has the following nice property: to check if the inequality $\sum_{j \in C} x_j \leqslant |C| - 1$ is valid for $\mathscr{P}_I$, it suffices to check if it is valid for $\mathscr{Q}_i$ for some i (that is, whether the set C is a cover for at least one of the individual knapsack constraints). This property is not shared by general 0–1 integer programs. Indeed, in general it is $\mathscr{NP}$-hard in the strong sense to check if such an inequality is valid, even for $|C| = 1$, as is easily shown.

Given a cover inequality $\sum_{j \in C} x_j \leqslant |C| - 1$ that is valid for $\mathscr{P}_I$, and a subset $D \subset C$, we call an inequality of the form

$$\sum_{j \in C \setminus D} x_j + \sum_{j \in N \setminus C} \alpha_j x_j + \sum_{j \in D} \beta_j x_j \leqslant |C \setminus D| + \sum_{j \in D} \beta_j - 1$$

a *global lifted cover inequality* (GLCI) if it is valid for $\mathscr{P}_I$. A global LCI need not be valid for any of the individual $\mathscr{Q}_i$. By analogy with the standard LCIs, we say that a GLCI is *simple* if $D = \emptyset$.

3.3. Lifting

In order to compute exact (i.e., best possible) up-lifting and down-lifting coefficients for GLCIs, one can solve a sequence of 0–1 MKP instances, as illustrated in the following example.

Example. Suppose $n = 5$ and $m = 3$, and that our system $Ax \leqslant b$ takes the form:

$$x_1 + x_2 + x_3 + x_5 \leqslant 2,$$
$$x_2 + x_3 + x_4 + 2x_5 \leqslant 3,$$
$$x_1 + x_4 + x_5 \leqslant 2.$$

The set $C = \{1,4,5\}$ forms a cover (third row). Suppose we set $D = \{5\}$. The restricted system, which we shall denote by $A'x \leqslant b'$, becomes

$$x_1 + x_2 + x_3 \leqslant 1,$$
$$x_2 + x_3 + x_4 \leqslant 1,$$
$$x_1 + x_4 \leqslant 1.$$

Immediately, we see that $\{1,4\}$ forms a cover for the third row, giving the initial cover inequality $x_1 + x_4 \leqslant 1$. Now, to up-lift x_2, we solve the 0–1 MKP:

$$\max\{x_1 + x_4 : A'x \leqslant b', x_2 = 1, x_3 = 0, x \in \{0,1\}^n\}.$$

Since the optimum is 0, we have shown that the inequality $x_1 + x_2 + x_4 \leqslant 1$ is valid for the restricted polytope. In a similar way, we can uplift x_3 to yield $x_1 + x_2 + x_3 + x_4 \leqslant 1$. Finally, to down-lift x_5 we solve the 0–1 MKP:

$$\max\{x_1 + x_2 + x_3 + x_4 : Ax \leqslant b, x_5 = 0, x \in \{0,1\}^n\}.$$

Since the optimum is 3, x_5 receives a lifting coefficient of $3 - 1 = 2$. The resulting GLCI, which induces a facet of $\mathscr{P}_I$, is $x_1 + x_2 + x_3 + x_4 + 2x_5 \leqslant 3$.

The above lifting method is too time-consuming to be of use in practice, since computing even a single lifting coefficient requires the solution of a 0–1 MKP. Our solution to this difficulty, as for the LCIs, is simply to solve the LP relaxation of these 0–1 MKP instances, and round down to the nearest integer. In fact, this idea appeared already in Wolsey [25], in the context of general 0–1 integer programs, but has apparently never been tested computationally.

Although the resulting GLCIs are no longer guaranteed to induce facets of $\mathscr{P}_I$, they can still be much stronger than standard LCIs, as the following example shows.

Example. Suppose our 0–1 MKP has $\binom{n}{2}$ constraints of the form $x_j + x_k \leqslant 1$ for all $1 \leqslant j < k \leqslant n$. Consideration of the first constraint yields the trivial cover inequality $x_1 + x_2 \leqslant 1$. Now suppose we lift x_3. The solution of the LP

$$\max\{x_1 + x_2 : Ax \leqslant b, x_3 = 1, x \in [0,1]^n\}$$

has a value equal to 0. Therefore, x_3 receives a lifting coefficient of 1, giving the inequality $x_1 + x_2 + x_3 \leqslant 1$. Similarly, when we lift x_4 we obtain a lifting coefficient of 1, giving the inequality $x_1 + x_2 + x_3 + x_4 \leqslant 1$. Repeating this argument yields the GLCI $x_1 + \cdots + x_n \leqslant 1$.

This example also shows that the resulting GLCIs can have unbounded *rank* in the sense of Chvátal [6], since the rank of the inequality $x_1 + \cdots + x_n \leqslant 1$ is $\log_2 n$.

Although this method for computing lifting coefficients involves a sequence of $n - |C \setminus D|$ 'auxiliary' LPs to produce a single GLCI, each LP is obtained from the previous one by changing only one objective coefficient and two bounds. Moreover, the early LPs in the sequence have a very small number of variables. Using a

simplex solver with a fast re-optimizing facility, one can usually solve the entire series of LPs in less time than it takes to solve the LP relaxation of the original 0–1 MKP. Thus, the computational burden is not excessive.

3.4. Enhancements

The above lifting scheme, based on LP relaxations, can be enhanced in at least two ways:

(i) After one or more rounds of a cutting plane algorithm, many GLCIs may have already been generated. These previously generated cuts can be added to the auxiliary LPs which are used to calculate the lifting coefficients. Although the size of the LPs to be solved is increased, this might be outweighed by the possibility of generating better lifting coefficients.
(ii) Suppose that a lower bound L is available, obtained by running any heuristic for the 0–1 MKP. Then one could add an additional inequality $c^{\mathrm{T}}x \geqslant L+1$ to the auxiliary LPs. We will call this inequality the *optimality constraint*. The resulting GLCIs will not in general be valid for P_I, but they will cut off only feasible integer solutions which are no better than the one we already have. (A similar idea was used by Nobili and Sassano [22] for producing lifting coefficients in the context of set covering.) A potential problem with this idea is that the auxiliary LPs can be infeasible. We discuss how to deal with this issue in Section 4.2.

We report results obtained with these ideas in Section 5.

4. Separation

To use a class of valid inequalities as cutting planes, one must solve the associated *separation problem*, i.e., the problem of detecting when an inequality in that class is violated by the solution of a given LP relaxation. In this section, we review some of the separation techniques that have been suggested for cover inequalities and their variants, and propose some new ones.

4.1. Local lifted cover inequalities (LCIs)

Crowder et al. [8] showed that the separation problem for cover inequalities (without lifting) can be formulated as a 0–1 knapsack problem. Since the knapsack problem is itself $\mathcal{NP}$-hard, they recommended simply inserting items into the cover in non-decreasing order of the ratio $(1-x_j^*)/a_{ij}$ (where $x^* \in [0,1]^n$ is the current fractional solution and i is the index of the knapsack constraint in question). Later, it was proven that the separation problem for cover inequalities is $\mathcal{NP}$-hard [9], and the same is true for standard LCIs [15,18].

Gu et al. [14] suggested a rather different, and simpler, greedy heuristic for building the initial cover: simply insert items in non-increasing order of x_j^*. They provide computational results which illustrate the superiority of their approach compared to that of Crowder et al. and we confirm this in Section 5. A possible explanation is that the method used by Crowder et al. tends to lead to covers of smaller cardinality, which in turn leads to more restricted lifting possibilities.

To our knowledge, the only paper which presents results for down-lifting is Gu et al. [14]. They present a particularly effective separation heuristic for LCIs, which, when specialised to the 0–1 MKP, becomes the following:

1. Take the current LP solution vector, x^*. Let $N^0 = \{j \in N : x_j^* = 0\}$ and $N^1 = \{j \in N : x_j^* = 1\}$.
2. Sort the variables in non-increasing order of x_i^* (breaking ties arbitrarily).
3. For $i = 1, \ldots, m$, do the following:
   3.1. Let S_i be the 'support' of the ith knapsack constraint, i.e., $S_i := \{j \in N : a_{ij} > 0\}$.
   3.2. Insert variables in S_i into C in non-increasing order of x_i^*, until a cover is obtained.
   3.3. Make the cover minimal: delete elements from C as long as the property of being a cover is preserved.
   3.4. Set $D = C \cap N^1$.
   3.5. Construct the cover inequality $\sum_{j \in C \setminus D} x_j \leqslant |C \setminus D| - 1$, which is valid for the restricted polyhedron.

3.6. Up-lift the variables in $S_i \setminus (C \cup N^0)$ in arbitrary order.
3.7. If the resulting inequality is not violated, stop.
3.8. Down-lift the variables in D.
3.9. Up-lift the variables in $(S_i \cap N^0) \setminus C$ in arbitrary order.
3.10. Output the violated LCI.

We found that the following refinements to this general scheme led to improvements in practice:

- In step 3, for a given i, if we have
$$\sum_{j \in S_i \setminus N^0} a_{ij} \leqslant b_i,$$
then x^* is easily shown to belong to the knapsack polytope $\mathcal{Q}_i$. Therefore, it is impossible for a violated LCI to be obtained from that row of the problem. Therefore, we immediately skip that row. Note that, as a result of this test, $C \cap N^0 = \emptyset$ always holds.
- In step 3.3, we omit elements from the cover in *non-decreasing* order of x^*-value. This is a greedy heuristic motivated by the fact that, if we delete element j, the violation of the cover inequality increases by $1 - x^*_j$.
- If the down-lifting coefficients in step 3.8 all turn out to be 1, down-lifting will have been a waste of time. To reduce the chance of this happening, we set $D = \emptyset$ in step 3.4 if either:
  - $S_i \setminus (C \cup N_0) = \emptyset$ (there would be no variables to uplift in step 3.6), or
  - $\sum_{j \in C} a_{ij} - \max_{j \in C}\{a_{ij}\} + \max_{j \in S_i \setminus (C \cup N_0)}\{a_{ij}\} \leqslant b_i$ (all up-lifting coefficients would be 0 in step 3.6).
- Suppose that $D \neq \emptyset$. If we encounter the difficulty mentioned in Section 2.3, i.e., if one of the coefficients on the left-hand side of the reduced knapsack inequality exceeds the right-hand side, we remove one or more items from D. More precisely, we compute the reduced knapsack capacity
$$b'_i = b_i - \sum_{j \in D} a_{ij}$$
and the maximum weight of the items to be up-lifted:
$$a_{\max} := \max_{k \in S_i \setminus (D \cup N^0)} a_{ik}.$$
If $a_{\max} > b'_i$, we iteratively remove items from D until $a_{\max} \leqslant b'_i$. As a greedy heuristic, we remove items from D in non-decreasing order of a_{ij}.
- If, after the above steps, $D = \emptyset$, we simply up-lift all variables using the bounds of [3,16] and the algorithm of [26]. We up-lift variables in $S_i \setminus (C \cup N^0)$ first, and check if the inequality is violated. If so, we proceed to up-lift the variables in $(S_i \cap N^0) \setminus C$.
- If on the other hand $D \neq \emptyset$, we use the bounds of [3,16] and the algorithm of [26] in step 3.6 only. In steps 3.8 and 3.9, on the other hand, we compute approximate lifting coefficients by solving the LP relaxation of the auxiliary 0–1 knapsack problems, and rounding down to integers, as explained in Section 2.3.

4.2. Global lifted cover inequalities (GLCIs)

After experimenting with various heuristics for GLCI separation, we decided in the end to use a scheme which is very similar to the one described in Section 4.1. Specifically, we use exactly the same method to produce our initial covers (one for each row) and our down-lifting sets, and an analogous lifting order: first up-lift variables in $N \setminus (C \cup N^0)$, then down-lift variables in D, then up-lift variables in $N^0 \setminus C$.

There is, however, one important difference. We sometimes have to remove additional items from D to prevent having undefined lifting coefficients for items in $N \setminus (C \cup N^0)$. This is because an item in this set, though it may have a small coefficient a_{ij} in the row under consideration, might have a very large weight in one of the other rows in the system. We implement this reduction in a simple greedy manner.

There is another issue. As mentioned in Section 3.4, if the optimality constraint $c^{\mathrm{T}}x \geqslant L + 1$ is added to the auxiliary lifting LPs, it is possible to encounter infeasibility when computing up-lifting coefficients. Our solu-

tion to this is as follows. If it occurs while up-lifting a fractional variable x_k in step 3.6, it means that there exists no integer solution with a profit larger than L that satisfies $x_k = 1$ and $x_j = 1$ for all $j \in D$. Thus, in this situation, one can simply generate the 'special' cover inequality $x_k + \sum_{j \in D} x_j \leqslant |D|$, which is easily seen to be violated. Note that, when $D = \emptyset$, the inequality reduces to $x_k \leqslant 0$, and therefore we can eliminate the variable x_k from the entire problem. Similarly, if infeasibility occurs while up-lifting a variable x_k in step 3.9, one can immediately eliminate the variable x_k.

5. Computational experiments

To evaluate the performance of the various inequalities and separation routines, we designed and coded a cutting plane algorithm and conducted extensive computational experiments on a set of standard 0–1 MKP test instances. The cutting plane algorithm took the following simple form:

1. Solve the initial LP by primal simplex. If the solution is integer, stop.
2. Call one or more separation algorithms for LCIs and/or GLCIs.
3. If any violated inequalities are found, add them to the LP, re-optimise the LP by dual simplex and go to 2.
4. Output the final upper bound and stop.

The algorithm was implemented in Microsoft Visual Studio.NET 2003 and called on functions from version 10.0 of the ILOG CPLEX Callable Library. All the experiments were performed using a PC with an Intel Pentium IV, 2.8 GHz processor and 512 MB RAM.

The set of instances used in our experiments was presented in Chu and Beasley [7] and contains 90 instances in total, all of which have 100 variables. This set is divided into three different subsets of 30 instances with 5, 10 and 30 constraints. Each subset is further divided based on the *tightness ratio* α, which takes the values 0.25, 0.50 and 0.75 (10 instances in each case).

Table 1 shows some relevant information for these instances. There is one row for each set of 10 instances. The first two columns show the number of constraints, m, and the tightness ratio, α. The following three columns show the average, over the given 10 instances, of the profit of the integer optimum, the upper bound given by the initial LP relaxation, and the average percentage difference between the two (sometimes called the *integrality gap*). To our knowledge, the values for $m = 30$ (computed using CPLEX) have never been displayed before in the literature.

The last two columns in Table 1 are intended to represent some kind of 'benchmark' for the evaluation of the strength of the LCIs and GLCIs. The column headed 'G&M' reports the average percentage of the integrality gap closed by the Gabrel and Minoux [12] method, based on exact separation of extended cover inequalities. The column headed 'LR' reports the average percentage gap closed by using the *best possible* bound that can be obtained by Lagrangian relaxation, which (see [21]) is equal to

$$\min_{1 \leqslant i \leqslant m} \max\{c^{\mathrm{T}} x : x \in \mathscr{P} \cap \mathscr{Q}_i\}.$$

Table 1
Some relevant data concerning the 90 Chu and Beasley instances

m	α (%)	Optimum	Initial UB	%Gap	G&M	LR
5	25	24197.2	24438.4	0.99	3.45	6.43
5	50	43252.9	43449.5	0.45	4.69	10.33
5	75	60470.9	60663.8	0.32	5.15	10.58
10	25	22601.9	22960.5	1.59	0.62	1.45
10	50	42660.6	43000.8	0.80	1.28	2.68
10	75	59555.4	59844.2	0.48	0.57	2.62
30	25	21660.4	22305.3	2.97	0.00	0.00
30	50	41440.4	41994.8	1.34	0.00	0.03
30	75	59201.8	59693.6	0.83	0.00	0.10

We computed this bound for each instance using an extremely time-consuming procedure similar to that of Boyd [5].

We remark that computing the best possible bound obtainable by *Lagrangian decomposition*, i.e., the optimum over $\bigcap_{i=1}^{m} \mathcal{Q}_i$, seems to be out of the question for these instances (using current methods).

Table 1 reveals that the average integrality gap, and so the difficulty of the instances, is positively correlated with the number of constraints, and negatively correlated with the tightness ratio. The extended cover inequalities do not close a significant amount of gap; in fact they close no gap at all for $m = 30$. Lagrangian relaxation closes a significant amount of gap for $m = 5$, but a negligible amount for larger m. This is probably due to the fact that the interaction between different constraints becomes important when m is large.

In the following three subsections, we report results obtained with the various lifted cover inequalities.

5.1. Local lifted cover inequalities (LCIs)

Table 2 reports the computational results obtained using local (i.e., traditional) LCIs only. For each set of 10 instances, and for six different separation strategies, we report the percentage of the integrality gap that was closed by the addition of LCIs. The columns headed 'simple' (respectively, 'general') correspond to the case in which only up-lifting is performed (respectively, both up- and down-lifting). The columns headed 'CJP' (respectively, 'GNS') correspond to the case in which the Crowder, Johnson and Padberg method (respectively, the Gu, Nemhauser and Savelsbergh method) is used to select the initial cover. The columns headed 'BOTH' correspond to the following strategy: the Crowder et al. method is applied only when the Gu et al. method fails.

Results are not given for $m = 30$, since no violated LCIs were found in that case. Moreover, we do not report running times since they are negligible (less than 0.03 seconds).

Several conclusions can be drawn from these results. The Gu et al. method of cover selection is clearly superior to the Crowder et al. method. The use of down-lifting is also clearly worthwhile. This confirms statements made in [14]. Also, as expected, the gap closed is smaller when $m = 10$ than when $m = 5$, since the LCIs do not take into account the interaction of different constraints. Interestingly, the extended cover inequalities close a slightly larger gap than the LCIs, despite the fact that they are less general. However, this is to be expected, since the Gabrel and Minoux separation algorithm is exact (and time-consuming), whereas we are using a heuristic for LCI separation.

We also experimented with several other strategies for the separation of LCIs, but they did not give substantially better results. In any case, the gaps closed by the LCIs are not impressive.

5.2. Global lifted cover inequalities (GLCIs)

Results obtained from the application of global LCIs are given in Table 3. As expected, GLCIs perform significantly better than local LCIs. In fact, for $m = 10$ and $m = 30$, they even outperform Lagrangian relaxation.

The computational time required for GLCIs is quite short as well. More precisely, for $m = 5$ (respectively, 10, 30), the mean computational time for the combined separation scheme (BOTH) for GLCIs is 0.20 (respec-

Table 2
Percentage gap closed by LCIs using different separation strategies

α (%)	m	Simple LCIs			General LCIs		
		CJP	GNS	BOTH	CJP	GNS	BOTH
25	5	2.46	2.82	2.86	2.79	3.24	3.50
50	5	2.97	3.46	3.87	3.75	4.67	4.85
75	5	3.47	3.61	3.82	4.35	5.67	5.85
25	10	0.39	0.36	0.38	0.48	0.56	0.59
50	10	0.69	0.65	0.74	1.00	1.18	1.30
75	10	0.15	0.21	0.21	0.29	0.56	0.56

Table 3
Percentage gap closed by GLCIs using different separation strategies

α (%)	m	Simple GLCIs			General GLCIs		
		CJP	GNS	BOTH	CJP	GNS	BOTH
25	5	2.55	4.25	4.28	3.31	5.85	6.41
50	5	3.12	3.69	4.11	4.22	6.88	6.90
75	5	3.84	4.61	4.63	4.58	5.89	6.12
25	10	0.50	2.92	2.92	0.90	5.61	5.61
50	10	0.72	1.84	1.86	1.45	5.37	5.39
75	10	0.29	0.35	0.35	0.58	2.79	2.85
25	30	0.09	1.39	1.39	0.31	2.44	2.44
50	30	0.03	0.55	0.56	0.51	4.64	4.64
75	30	0.00	0.22	0.22	0.29	3.75	3.75

tively, 0.35, 4.10) seconds. In all cases, the time of the cutting plane procedure was negligible compared to the time taken to solve the 0–1 MKP instances to optimality by branch-and-bound or branch-and-cut.

We also experimented with various combined strategies, using local and global LCIs in different orders, but the results obtained were not promising.

5.3. Enhanced GLCIs

In Section 3.4, we mentioned two simple ways in which the lifting coefficients for GCLIs could be improved. The following three tables include results from the application of these ideas, both alone and in combination. Table 4 corresponds to the idea of appending previously generated covers to the auxiliary lifting LPs. The gap closed from the application of GLCIs is improved in this way, though not substantially.

Although one might expect the enlargement of the auxiliary LPs to increase the overall computational time required, in practice the observed times were on average similar to those mentioned in Section 5.2. Indeed, a general point is that the overall computing time is rather unpredictable, affected by the interaction between factors such as cut quality, cut density, number of cuts generated, and number of rounds of separation performed.

Table 5 includes the results obtained by adding the optimality constraint $c^T x \geqslant L + 1$ to the auxiliary LPs. The lower bound value L could have been derived using any primal heuristic for the 0–1 MKP. Here, however, we chose to set L to the profit of the optimal integer solution to the 0–1 MKP, so that the results displayed represent the *best possible* bounds that could be obtained by the application of this idea (with our separation heuristics of course). For $m = 5$ and $m = 10$, there is a significant improvement in the gap closed. However, this does not hold for $m = 30$. The mean computational time for $m = 5$ and $m = 10$ was slightly higher (0.3 and 0.95 seconds, respectively), while for $m = 30$ it was slightly lower (3.75 seconds).

Table 4
Percentage gap closed by GLCIs using different separation strategies when the lifting LPs include the previously generated covers

α (%)	m	Simple GLCIs			General GLCIs		
		CJP	GNS	BOTH	CJP	GNS	BOTH
25	5	2.55	4.25	4.28	3.31	6.11	7.92
50	5	3.12	3.69	4.11	4.22	6.94	6.96
75	5	3.84	4.61	4.63	4.58	5.60	5.83
25	10	0.50	2.92	2.92	0.90	5.61	5.59
50	10	0.72	1.84	1.84	1.45	5.37	6.46
75	10	0.29	0.35	0.35	0.61	2.79	3.12
25	30	0.09	1.37	1.37	0.31	2.74	2.74
50	30	0.03	0.40	0.41	0.60	4.89	4.89
75	30	0.00	0.22	0.22	0.29	3.90	3.90

Table 5
Percentage gap closed by GLCIs using different separation strategies when the lifting LPs include the optimality constraint

α (%)	m	Simple GLCIs			General GLCIs		
		CJP	GNS	BOTH	CJP	GNS	BOTH
25	5	4.27	5.45	5.78	6.31	8.38	9.10
50	5	5.95	5.85	6.70	9.47	10.83	11.29
75	5	7.70	8.21	8.66	11.95	12.37	13.54
25	10	1.10	2.46	2.87	2.32	6.83	7.07
50	10	2.40	2.62	3.16	3.37	7.31	7.35
75	10	1.35	1.76	2.10	3.49	6.20	6.33
25	30	0.00	1.32	1.34	0.00	2.55	2.34
50	30	0.07	0.53	0.53	0.61	4.60	4.31
75	30	0.07	0.51	0.52	0.32	4.73	3.77

Table 6
Percentage gap closed by GLCIs using different separation strategies when the lifting LPs include the previously generated covers and the optimality constraint

α (%)	m	Simple GLCIs			General GLCIs		
		CJP	GNS	BOTH	CJP	GNS	BOTH
25	5	4.24	5.88	6.24	7.11	14.08	14.69
50	5	6.21	6.43	7.56	12.53	14.19	14.59
75	5	8.06	8.47	9.55	16.57	19.92	21.81
25	10	1.11	2.99	4.03	2.71	8.58	8.72
50	10	2.60	3.14	3.45	5.33	9.27	10.94
75	10	1.63	2.09	2.81	5.09	10.03	10.62
25	30	0.00	1.31	1.31	0.00	2.55	2.55
50	30	0.07	0.47	0.47	0.71	4.60	4.99
75	30	0.07	0.52	0.53	0.31	4.73	4.73

Finally, the results obtained from the combined application of both ideas are given in Table 6. For $m = 5$ and $m = 10$, the improvement is quite substantial. For $m = 30$, even though there is still an improvement, it is not as significant. Using this configuration, the computational times were the highest observed: 1.40, 4.05 and 6.15 seconds, respectively, for $m = 5$, $m = 10$ and $m = 30$. In our view, however, this increased computational time is justified by the improvement achieved in terms of the gap closed. We also applied this specific lifting strategy to a series of easier (uncorrelated) instances contained in the `mknapl` dataset of the OR-library. For all seven instances in this dataset we were able to quickly close 100% of the integrality gap.

6. Concluding remarks

We have seen that LCIs can be successfully applied to the 0–1 MKP for the derivation of upper bounds. Local LCIs can be useful for instances with a small number of constraints, but they perform poorly for instances with many constraints. Global LCIs perform reasonably well in all cases, since they can capture more information contained in the constraint matrix. Moreover, the computational burden for their derivation is not excessive.

Adding the previously generated covers to the auxiliary LPs can improve the upper bounds derived. Further improvement can be achieved by adding an optimality constraint to the auxiliary LPs as well, assuming that a lower bound is available.

Our preferred separation scheme not only combines heuristic ideas that were present in Crowder et al. [8] and Gu et al. [14], but also includes a series of refinements. Of great importance in this scheme is to produce general LCIs instead of simple LCIs, i.e., to use down-lifting, whenever possible.

The upper bounds that we obtain using the enhanced version of GLCIs dominate, to our knowledge, all of the available upper bounds in the literature for the 0–1 MKP. Nevertheless, we believe that there is room for improvement. One idea that merits investigation is the use of generic knapsack facets (such as Fenchel cuts [5]) in combination with GLCIs. Since generation of generic knapsack facets is time-consuming, one might generate them only when LCI or GLCI separation fails.

Acknowledgements

Thanks to Augusto de Conto, Paola Cappanera and the anonymous referees for useful comments.

References

[1] A. Atamturk, Cover and pack inequalities for (mixed) integer programming, Annals of Operations Research 139 (2005) 21–38.
[2] E. Balas, Facets of the knapsack polytope, Mathematical Programming 8 (1975) 146–164.
[3] E. Balas, E. Zemel, Facets of the knapsack polytope from minimal covers, SIAM Journal on Applied Mathematics 34 (1978) 119–148.
[4] T. Bektas, O. Oguz, On separating cover inequalities for the multidimensional knapsack problem, Computers and Operations Research 34 (2007) 1771–1776.
[5] E.A. Boyd, Generating Fenchel cutting planes for knapsack polyhedra, SIAM Journal on Optimization 3 (1993) 734–750.
[6] V. Chvátal, Edmonds polytopes and a hierarchy of combinatorial problems, Discrete Mathematics 4 (1973) 305–337.
[7] P.C. Chu, J.E. Beasley, A genetic algorithm for the multidimensional knapsack problem, Journal of Heuristics 4 (1998) 63–86.
[8] H. Crowder, E. Johnson, M. Padberg, Solving large-scale 0–1 linear programming programs, Operations Research 31 (1983) 803–834.
[9] C.E. Ferreira, A. Martin, R. Weismantel, Solving multiple knapsack problems by cutting planes, SIAM Journal on Optimizaton 6 (1996) 858–877.
[10] A. Fréville, The multidimensional 0–1 knapsack problem: An overview, European Journal of Operational Research 155 (2004) 1–21.
[11] A. Fréville, S. Hanafi, multidimensional 0–1 knapsack problem: Bounds and computational aspects, Annals of Operations Research 139 (2005) 195–227.
[12] V. Gabrel, M. Minoux, A scheme for exact separation of extended cover inequalities and application to multidimensional knapsack problems, Operations Research Letters 30 (2002) 252–264.
[13] M.R. Garey, D.S. Johnson, Computers and Intractability: An Introduction to the Theory of $\mathcal{NP}$-completeness, W.H. Freeman, San Francisco, 1979.
[14] Z. Gu, G.L. Nemhauser, M.W.P. Savelsbergh, Cover inequalities for 0–1 linear programs: computation, Informs Journal on Computing 10 (1998) 427–437.
[15] Z. Gu, G.L. Nemhauser, M.W.P. Savelsbergh, Cover inequalities for 0–1 linear programs: Complexity, Informs Journal on Computing 11 (1999) 117–123.
[16] Z. Gu, G.L. Nemhauser, M.W.P. Savelsbergh, Sequence independent lifting in mixed integer programming, Journal of Combinatorial Optimization 4 (2000) 109–129.
[17] M. Guignard, S. Kim, Lagrangean decomposition: A model yielding strong Lagrangean bounds, Mathematical Programming 39 (1987) 215–228.
[18] D. Klabjan, G. Nemhauser, C. Tovey, The complexity of cover inequality separation, Operations Research Letters 23 (1998) 35–40.
[19] A.N. Letchford, A. Lodi, Strengthening Chvátal–Gomory cuts and Gomory fractional cuts, Operations Research Letters 32 (2002) 74–82.
[20] A. Martin, R. Weismantel, The intersection of knapsack polyhedra and extensions, in: R.E. Bixby, E.A. Boyd, R.Z. Rios-Mercado (Eds.), Proceedings of the Sixth IPCO Conference, Lecture Notes in Computer Science, vol. 1412, Springer-Verlag, 1998.
[21] G.L. Nemhauser, L.A. Wolsey, Integer and Combinatorial Optimization, Wiley, New York, 1988.
[22] P. Nobili, A. Sassano, A separation routine for the set covering polytope, in: E. Balas, G. Cornuéjols, R. Kannan (Eds.), Proceedings of the Second IPCO Conference, CMU Press, Pittsburgh, 1992.
[23] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer-Verlag, 2004.
[24] L.A. Wolsey, Faces for linear inequalities in 0–1 variables, Mathematical Programming 8 (1975) 165–178.
[25] L.A. Wolsey, Facets and strong valid inequalities for integer programs, Operations Research 24 (1976) 367–372.
[26] E. Zemel, Easily computable facets of the knapsack polytope, Mathematics of Operations Research 14 (1989) 760–765.