Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 1771–1776
computers & operations research
www.elsevier.com/locate/cor

On separating cover inequalities for the multidimensional knapsack problem

Tolga Bektas[*], Osman Oğuz

Department of Industrial Engineering, Bilkent University, 06800, Bilkent, Ankara, Turkey

Available online 19 August 2005

Abstract

We propose a simple and a quite efficient separation procedure to identify cover inequalities for the multidimensional knapsack problem. It is based on the solution of a conventional integer programming model. Solving this kind of integer programs is usually considered expensive and the proposed method may have been overlooked because of this assumption. The results of our experiments with a small set of randomly generated problems and problems taken from the literature indicate that the method may be a reasonable alternative to the one currently in use.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Integer programming; Cover inequalities; Separation problem

1. Introduction

In this paper, we propose an exact separation procedure to identify cover inequalities for the *multidimensional knapsack problem* (*mKP*)

$$\max \quad \sum_{i \in N} c_i x_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i \in N} a_{ij} x_i \leqslant b_j, \quad j \in M, \tag{2}$$

$$x_i \in \{0, 1\}, \quad i \in N, \tag{3}$$

[*] Corresponding author. Tel.: +90 312 2341010; fax: +90 312 2341051.
E-mail address: tolgab@bilkent.edu.tr (T. Bektas).

where a_{ij}, b_i and c_i are nonnegative integers. The reader is referred to Chapter 9 of Kellerer et al. [1] for a recent survey on the problem. Before presenting the procedure which is a straightforward extension of the well-known separation procedure for the unidimensional knapsack problem, we provide a brief background on the cover inequalities for the knapsack problem and the related separation problem in the next section. In Section 3, we present the proposed separation procedure for the *mKP*. Computational results of the proposed procedure on a set of test problems are given in Section 4. Remarks and conclusions are in Section 5.

2. Cover inequalities and the separation problem

Cover inequalities are one of the most important components of the branch-and-cut algorithms designed to solve 0–1 integer programming problems. The separation problem for cover inequalities in the context of 0–1 integer programming was introduced by Crowder et al. [2]. Recent studies by Gu et al. [3,4] provide extensive discussions of available strategic choices for using cover inequalities in the branch-and-cut process for 0–1 programming. We provide below a brief review of the separation problem for the unidimensional knapsack problem.

Consider the polyhedron $K=\{x \in \mathscr{B}^{|N|} : \sum_{i\in N} a_i x_i \leqslant b\}$ associated with the unidimensional knapsack problem. The index set $C \subseteq N$ of variables for which $\sum_{i\in C} a_i > b$ holds is called a *cover*. A cover that loses this property when any one of the indices in it is excluded is called a *minimal cover*. A minimal cover induces a so-called *cover inequality* that is valid for the polyhedron K and is given as follows:

$$\sum_{i\in C} x_i \leqslant |C| - 1. \tag{4}$$

Let $X^* = \{x_1^*, x_2^*, \ldots\} \in [0, 1]^n$ be an arbitrary feasible solution to $\sum_{i\in N} a_i x_i \leqslant b$, where a_i's are non-negative integers. Deciding whether X^* satisfies all of the possible cover inequalities of K is known as the *cover separation problem* (*CSP*). The *CSP* can be formulated as the following 0–1 integer linear program:

$$z = \min \quad \sum_{i\in N} (1 - x_i^*) z_i \tag{5}$$

$$\text{s.t.} \quad \sum_{i\in N} a_i z_i \geqslant b + 1 \tag{6}$$

$$z_i \in \{0, 1\}, \quad i \in N. \tag{7}$$

If $z < 1$, then the set $C = \{i : \bar{z}_i = 1\}$ induces a cover inequality of the form (4), where $\bar{z}_i$ is an optimal solution to the *CSP*. Otherwise, X^* satisfies all the possible cover inequalities.

Most studies on this subject indicate that the exact solution of the separation problem is costly in practice and usually resort to a greedy type algorithm to obtain approximate solutions. For more details, the reader may refer to Wolsey [5] and Nemhauser and Wolsey [6]. Recently, Kellerer et al. [1] presented computational results on solving the separation problem.

3. An exact separation procedure for mKP

We now extend the previously discussed separation procedure given for the unidimensional knapsack problem to the *mKP*. In this case, each constraint $j \in M$ of the *mKP* is associated a set of cover inequalities denoted by C_j. Given a fractional solution $X^* = \{x_1^*, x_2^*, \ldots\}$ to the *mKP* (usually the solution to the corresponding LP-relaxation), we refer to the problem of identifying whether X^* violates a cover inequality in $\bigcup_{j \in M} C_j$ or concluding that it satisfies all the possible ones as the *generalized cover separation problem* (*GCSP*). We formulate the *GCSP* by the following 0–1 integer linear programming formulation:

$$z = \min \sum_{i \in N^f} (1 - x_i^*) z_i \tag{8}$$

$$\text{s.t.} \quad \sum_{i \in N^f} a_{ij} z_i \geqslant b'_j + 1 - R(1 - y_j), \quad j \in M \tag{9}$$

$$\sum_{j \in M} y_j \geqslant 1 \tag{10}$$

$$z_i \in \{0, 1\}, \quad i \in N^f, \tag{11}$$

$$y_j \in \{0, 1\}, \quad j \in M, \tag{12}$$

where $b'_j = b_j - \sum_{i \in N^1} a_{ij}$ with $N^1 = \{i : x_i^* = 1\}$ and $N^f = \{i : 0 < x_i^* < 1\}$. The parameter R used in constraint (9) is a sufficiently large constant (which may be chosen as $R = \max_{j \in M} b_j + 1$). In this formulation, the additional binary variable y_j is used to check the violation of cover inequalities in the set C_j.

Given an optimal solution $\overline{Z} = \{\overline{z}_1, \overline{z}_2, \ldots\}$ with value $z < 1$ to the integer linear programming formulation, the set $C = N^1 \cup \{i : \overline{z}_i = 1\}$ induces a violated cover inequality given by (4). Note that we only include in the formulation variables $i \in N^f$, since one can easily fix $\overline{z}_i = 1$ for $i \in N^1$ and $\overline{z}_i = 0$ for $i \in N \backslash \{N^1 \cup N^f\}$. This also reduces the size of the binary variables in the formulation, thereby facilitating its solvability.

The separation procedure consists of identifying the cover inequality via the *GCSP* that is violated by the fractional solution X^* and appending it to the formulation. The augmented formulation is resolved and cuts are appended in a similar and an iterative manner until no violated cover inequalities are found. We refer to this procedure as *generalized cover separation* (*GCS*).

As it has been already pointed out, the 0–1 programming model above is very straightforward, and it is a simple exercise to extend the unidimensional knapsack version to this generalized case. However, the computational experimentation results presented in the next section clearly demonstrates that, with currently available mathematical programming software, there are benefits to reap in using it for solving the separation problem.

4. Computational results

In this section, we describe our computational experience with the proposed procedure on test problems. The generalized cover separation procedure proposed for the *mKP* has been implemented in C and all

Table 1
Statistics for a sample of 50 randomly generated instances

$m \times n$	Avg. cover		Max. cover		T_{avg}		*CPLEX*	
	GR	*GCS*	*GR*	*GCS*	*GR*	*GCS*	T_{C}	n_{C}
5×20	2.20	7.20	5	13	0.02	0.03	0.01	5/5
10×20	1.20	9.60	2	22	0.04	0.03	0.01	5/5
25×100	2.80	5.60	5	8	0.10	0.10	0.28	5/5
50×100	1.40	6.20	4	12	0.21	0.14	0.89	5/5
125×500	1.80	9.20	4	26	2.61	1.44	93.67	5/5
250×500	1.80	2.80	3	5	5.16	3.23	505.24	5/5
250×1000	2.00	4.20	4	8	11.75	6.64	3353.80	5/5
500×1000	2.00	4.00	4	6	23.58	13.09	3326.60	4/5
500×2000	1.20	4.80	5	8	82.26	28.17	8512.40	2/5
1000×2000	1.60	7.20	4	15	145.01	57.88	6785.40	2/5

the tests are performed on a Sun UltraSPARC 12×400 MHz with 3 GB RAM, using CPLEX 9.0 as the optimization package.

We compare the proposed method with a simple and a straightforward separation procedure in which violated covers are identified using a greedy algorithm (henceforth denoted by *GR*). To the best of the authors' knowledge, there are no other heuristic procedures previously proposed to separate cover inequalities for the *mKP*. At every iteration of the algorithm, we try to identify a violated cover inequality for each constraint $j \in M$. More specifically, for each constraint $j \in M$, the variables of the *mKP* are put in increasing order of the ratios $(1 - x_i^*)/a_{ij}$ and the variable with the smallest ratio is set equal to 1 while keeping the rest at zero. Then, constraint *j* is checked as to whether this solution causes a violation. If there is a violation, then a cover inequality for this constraint is identified consisting of this single variable. Otherwise, the variable with the second lowest ratio is raised to 1, and the process is repeated until a violating solution is found for constraint *j*. Among all the cover inequalities identified as a result of scanning all $|M|$ constraints, the one with the maximum violation is appended to the LP-relaxation of the problem and the resulting problem is solved to optimality. This concludes a single iteration. The procedure continues in an iterative manner until no violated cover inequalities are found. For each constraint, the *GR* has a time complexity of $\mathcal{O}(|N| \log |N|)$ to sort the variables and $\mathcal{O}(\log |N|)$ for the binary search. Therefore, as a result of scanning all the constraints, the *GR* has an overall time complexity of $\mathcal{O}(|M||N| \log |N|) + \mathcal{O}(|M| \log |N|)$ to identify a single violated cover inequality at each iteration.

The performance of the algorithm was tested on both randomly generated instances and instances taken from the literature. For the former group, a batch of 50 multidimensional 0–1 integer programming problems were generated pseudo-randomly with the following specifications: the number of constraints (m) range between 5 and 1000. That of variables (n) range between 20 and 2000. There are 10 different combinations of these dimension parameters as may be seen in the first column of Table 1. For each combination, linear relaxations of five pseudo-randomly generated problems are solved. All objective function coefficients and constraint coefficients have values uniformly distributed between 0 and 100. The right-hand side constants are computed using the formula $b_i = 10np_1 + 0.5p_2 \sum_{j=1}^{n} a_{ij}, \ \forall i = 1, \ldots, m$.

Table 2
Statistics for the OR-Library instances

Instance	$m \times n$	Avg. cover		Max. cover		T_{avg}		*CPLEX*	
		GR	*GCS*	*GR*	*GCS*	*GR*	*GCS*	T_{C}	n_{C}
mknapcb1	5×100	2.20	5.77	5	14	0.05	0.06	20.58	30/30
mknapcb2	5×250	2.33	4.33	6	8	0.11	0.10	532.69	30/30
mknapcb3	5×500	2.30	5.10	6	12	0.22	0.17	6702.10	17/30
mknapcb4	10×100	0.40	1.70	2	7	0.20	0.13	164.47	30/30
mknapcb5	10×250	0.57	1.13	2	3	0.34	0.23	10 506.07	2/30
mknapcb6	10×500	0.33	1.20	1	3	1.10	0.36	10 848.13	0/30

Here, p_1 and p_2 are pseudo-random variates uniformly distributed between 0 and 1. This formula was chosen to provide variability in tightness among constraints, and avoid the possibility of having a constraint with right-hand side equal to zero.

It may seem counter-intuitive to solve a rather complex integer programming formulation to separate valid inequalities for a seemingly simpler formulation of the *mKP*. Therefore, in order to investigate whether the separation problem is easier to solve than the original multidimensional knapsack problem, we have used state-of-the-art optimization software CPLEX 9.0 to solve the instances considered here using the formulation defined by (1)–(3). A time limit of 3 h (10 800 s) is imposed on CPLEX.

The results of the computational experiments on random instances are presented in Table 1. In this table, the two columns under the heading *Avg. cover* present the average number of cover inequalities found by *GR* and *GCS*, respectively. Each entry in these columns is calculated over five instances. The next two columns present the maximum number of cover inequalities found by the two procedures. The two columns under the heading T_{avg} indicate the unit time required by the corresponding algorithm (in seconds) to identify a violated cover inequality per cover and to solve the LP-relaxation, which is calculated by dividing the total solution time to the total number of cover inequalities found. The last two columns, T_C and n_C, show the average time required to solve the instances and the number of problems that could be solved to optimality out of the total number of problems within the 3 h time limit, respectively.

As Table 1 shows, the number of cover inequalities identified by *GCS* is clearly superior to that of *GR*. We additionally note that the *GCS* identified a total of 304 violated cover inequalities over all instances, whereas *GR* produced 90 of these. That is, *GR* missed about 70% of the violated cover inequalities produced by *GCS*. In addition, the unit time required by the *GCS* becomes superior to that of *GR* as the instances grow bigger in size. Thus, we can conclude that the *GCS* is quite efficient considering the size of the problems handled and the gain acquired in terms of the number of cover inequalities produced. The last two columns indicate that the time spent for cover generation is only a small fraction of the time required by CPLEX. This implies that the *GCS* is not computationally expensive as compared to CPLEX.

The performance of both algorithms on instances taken from the OR-Library [7] is given in Table 2. The first column of the table presents the name of the group of instances, where each group contains 30 problems. The average number of covers and the reported solution times are calculated as the average of 30 problems for each instance group. The remaining columns of this table are same as those of Table 1.

For the problems presented in Table 2, we also note that *GCS* identified a total of 577 violated cover inequalities over all instances whereas *GR* produced 244 of these, indicating that *GR* missed about 58% of the violated cover inequalities produced by *GCS*. Similar to the previous result, the last two columns of Table 2 indicate that the separation problem is indeed much more easier to solve than the *mKP*. In fact, as instances grow larger in size, the results demonstrate that CPLEX was not able to find the integer optimal solution for most of the problems.

5. Conclusions

In this paper, we have proposed a separation procedure for the multidimensional knapsack problem that is based on solving a single 0–1 integer programming formulation at each iteration. The procedure proves to be simple and quite efficient whilst the implementation can be done quite easily, requiring no specialized algorithm. In addition, the computational results indicate that the proposed algorithm may be a viable alternative in separating cover inequalities for the multidimensional knapsack problem.

Acknowledgements

The authors would like to thank the reviewers for useful comments and for pointing out a recent new reference.

References

[1] Kellerer H, Pferschy U, Pisinger D. Knapsack problems. Berlin: Springer; 2004.
[2] Crowder H, Johnson EL, Padberg MW. Solving large scale zero-one linear programming problems. Operations Research 1983;31:803–34.
[3] Gu Z, Nemhauser GL, Savelsbergh MWP. Lifted cover inequalities for 0–1 integer programs: computation. INFORMS Journal on Computing 1998;10(4):427–37.
[4] Johnson EL, Nemhauser GL, Savelsbergh MWP. Progress in linear programming-based algorithms for integer programming: an exposition. INFORMS Journal on Computing 2000;12(1):2–23.
[5] Wolsey LA. Integer programming. New York: Wiley; 1998.
[6] Nemhauser GL, Wolsey LA. Integer and combinatorial optimization. New York: Wiley; 1999.
[7] OR-Library. Available at ⟨http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/mknapinfo.html⟩.