A computational study of exact knapsack separation for the generalized assignment problem — Source link

Pasquale Avella, Maurizio Boccia, Igor Vasilyev

Institutions: University of Sannio, Russian Academy of Sciences

Published on: 01 Apr 2010 - Computational Optimization and Applications (Springer US)

Topics: Generalized assignment problem, Continuous knapsack problem, Knapsack problem, Weapon target assignment problem and Quadratic assignment problem

A Computational Study of Exact Knapsack Separation for the Generalized Assignment Problem

Pasquale Avella [1] Maurizio Boccia [1] Igor Vasilyev [2]

[1]RCOST - Research Center on Software Technology, Università del Sannio, Viale Traiano, 82100 Benevento, Italy, {avella, maboccia}@unisannio.it

[2]Institute of System Dynamics and Control Theory, Siberian Branch of Russian Academy of Sciences, Lermontov Srt., 134, 664033 Irkutsk, Russia, vil@icc.ru

Abstract

The Generalized Assignment Problem is a well-known NP-hard combinatorial optimization problem which consists of minimizing the assignment costs of a set of jobs to a set of machines satisfying capacity constraints. Most of the existing algorithms are of a Branch-and-Price type, with lower bounds computed through Dantzig-Wolfe reformulation and column generation.

In this paper we propose a cutting plane algorithm working in the space of the variables of the basic formulation, whose core is an exact separation procedure for the knapsack polytopes induced by the capacity constraints. We show that an efficient implementation of the exact separation procedure allows to deal with large-scale instances and to solve to optimality several previously unsolved instances.

1 Introduction

Let $M = \{1, \dots, m\}$ be a set of machines and let $N = \{1, \dots, n\}$ be a set of tasks to be assigned to the machines M. Let c_{ij} be the cost of assigning the task j to the machine i. Let d_{ij} be the amount of resource required by the machine i to perform the task j. Each machine has a limited amount of resources available. Let u_i be the capacity of the machine $i \in M$.

The *Generalized Assignment Problem* (GAP) is to find a minimum assignment cost of the tasks N to the machines M satisfying the constraint that the total amount of resources required by each machine $i \in M$ does not exceed its capacity u_i.

Because of its computational difficulty, GAP is a challenging integer programming problem, which stimulated a wide interest among researchers [1, 9, 11, 14, 15, 18, 19, 16, 20, 21, 23, 24, 25, 26].

The basic GAP formulation includes m knapsack constraints. Tightening MIP formulations by deriving Lifted Cover Inequalities from knapsack constraints is now a consolidate technique, embedded into all the main MIP solvers. In this paper we go a step further in this direction and report on a computational experience with an exact separation procedure for the polytopes induced by each knapsack constraint, i.e. a separation procedure which either returns a separating hyperplane between a knapsack polytope and a given fractional solution or concludes that the fractional solution is an internal point of the knapsack polytope.

The separation procedure was embedded into a Branch-and-Cut scheme. The cutting plane algorithm yielded the optimal solution of all the OR-Library [2] instances with $n \leq 200$, with the only exception of $d20200$. Furthermore the algorithm yielded provably good solutions for the larger - previously unsolved - OR-Library instances and solved several of them to exact optimality.

The remainder of the paper is organized as follows. In section 2 GAP formulations are discussed. Section 3 shows the details of the exact knapsack separation procedure. Section 4 provides a detailed report on the computational experience. Finally, section 5 highlights the points of strength and weakness of the separation procedure.

2 Problem formulation

Let x_{ij} be a binary variable expressing the assignment of the task $j \in N$ to the machine $i \in M$. The Generalized Assignment Problem can be formulated as:

$$\min_x \quad \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij}$$

$$\sum_{i \in M} x_{ij} = 1, \quad j \in N \tag{1}$$

$$\sum_{j \in N} d_{ij} x_{ij} \le u_i, \quad i \in M \tag{2}$$

$$x_{ij} \in \{0,1\}, \quad i \in M, j \in N \tag{3}$$

Constraints (1) require that each task must be assigned to a machine. Capacity constraints (2) enforce the condition that the amount of resources required by the tasks assigned to the machine i does not exceed its capacity u_i.

Let X_{GAP} denote the set of solutions satisfying (1)-(3) and let $P_{GAP} = conv(X_{GAP})$ denote the GAP polytope. The polyhedral properties of the "submissive" of P_{GAP} have been studied in [11, 14, 15].

The lower bound returned by the LP relaxation of the formulation (1)-(3) is usually weak. Formulation (1)-(3) can be tightened by considering the knapsack polytopes defined by the capacity constraints (2):

$$\max_x \quad \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij}$$

$$\sum_{i \in M} x_{ij} = 1, \quad j \in N \tag{4}$$

$$x \in P_{KN}(i), \quad i \in M \tag{5}$$

$$x_{ij} \in \{0,1\}, \quad i \in M, j \in N \tag{6}$$

where

$$P_{KN}(i) = conv(\{x \in B^{m \times n} : \sum_{j \in N} d_{ij} x_{ij} \le u_i\})$$

is the knapsack polytope associated with the i-th capacity constraint.

All the known exact algorithms devised for GAP are based on formulation (4)-(5). Nauss [19] proposed a Branch-and-Bound based on the lagrangian relaxation of the equality constraints (4). Savelsbergh proposed a Branch-and-Price algorithm [23] based on Dantzig-Wolfe decomposition and column generation.

A main drawback of Dantzig-Wolfe decomposition is the convergence difficulty of column generation. Recently Pigatti et al. [21] presented a Robust Branch-and-Price algorithm where a stabilization technique for the dual variables is introduced to improve the convergence of column generation. They yielded remarkable results, reporting on the optimal solution (for some of them for the first time) of all the OR-Library instances with $n \leq 200$, with the only exceptions of $d10200$ and $d20200$.

The algorithm presented in this paper runs in the space of original variables, using the facets of $P_{KN}(i)$ as cutting planes, as detailed in the following section.

3 Exact knapsack separation

The exact separation of knapsack polytope was first suggested by Boyd [4, 5, 6, 7] as a tool for Integer Programming. The recent papers by Fukasawa and Goycoolea [13], Kaparis and Letchford [17] and the dissertations of P. Bonami [3] and D. Espinoza [12], show a renewed interest in this topic.

Let $X_{KN} = \{y \in \{0,\ 1\}^n :\ a^T y \leq b\}$ be the set of feasible solution of a knapsack problem. Given a knapsack polytope $P_{KN} = conv(X_{KN})$ and a point $\bar{y} \in I\!R^n$, the *separation problem* consists of finding a separating hyperplane between P_{KN} and $\bar{y}$, or saying that $\bar{y} \in P_{KN}$. The separation problem amounts to solve the following LP:

$$\begin{aligned} \theta \quad = \quad & \max_{(\alpha,\beta)} \left[\bar{y}^T \alpha - \beta\right] \\ & v^T \alpha \leq \beta \qquad\qquad v \in X_{KN} \qquad (7) \\ & (\alpha, \beta) \in S \qquad (8) \end{aligned}$$

where S is a convex compact set under some conditions. Some possible choices for the set S are discussed in subsection 3.1. Let (α^*, β^*) be an optimal solution of the problem. If $\theta \leq 0$, then $\bar{y} \in P_{KN}$, otherwise ${\alpha^*}^T y \leq \beta^*$ is the desired separating hyperplane.

The separation LP contains a huge number of constraints (7) and is solved by the following procedure:

i) The separation problem is solved over the polytope

$$P_{KN}(\bar{y}) = \{y \in P_{KN} :\ y_i = 0 \text{ if } \bar{y}_i = 0,\ y_i = 1 \text{ if } \bar{y}_i = 1\},$$

i.e. the polytope defined by the fractional support of $\bar{y}$, since it is known that a separating hyperplane for P_{KN} exists iff it exists for $P_{KN}(\bar{y})$. This leads to a significant reduction of the problem size.

ii) The reduced problem still contains an intractable number of constraints and requires row generation for its solution, as described in subsection 3.2.

iii) After a cutting plane has been generated, it is post-processed as described in subsection 3.3 to avoid numerical errors.

iv) Finally, standard sequential lifting is used to convert the facets of $P_{KN}(\bar{y})$ into facets of P_{KN}.

3.1 Normalizations

Different choices for the "normalization set" S can affect the quality of the generated cuts and the performances of exact knapsack separation, as already pointed out for other classes of general cutting planes [10, 3]. Here we consider four different normalizations:

- $\boldsymbol{\beta = 1}$

  By letting $\beta = 1$, the ratio between the violation and the right hand side is maximized. Moreover, every extreme solution of the LP (7)-(8) produces a facet of P_{KN}.

- $\boldsymbol{\mathcal{L}^1}$ **norm**

  With this normalization we pose $\sum_{j \in N} \alpha_j = 1$ and $\alpha_j \geq 0$. The generated cut maximizes the ratio between the violation and the size of the support, i.e. the distance between $\bar{y}$ and P_{KN} computed according to the $\mathcal{L}_1$ norm. Cutting planes generated with this normalization are not facet-inducing for P_{KN}.

- **Inverted** $\boldsymbol{\mathcal{L}^1}$ **norm**

  With this separation problem, the objective function is to minimize the sum of the coefficients with the constraint that the cutting plane must be violated by a given amount, i.e. the separation problem becomes:

$$
\begin{aligned}
\theta \quad = \quad & \min_{(\alpha,\beta)} \sum_{j \in N} \alpha_j \\
& v^T \alpha \leq \beta \qquad\qquad v \in X_{KN} \\
& \bar{y}^T \alpha - \beta = 1 \\
& \alpha \geq 0
\end{aligned}
$$

  Cutting planes generated with this normalization are not facet-inducing for P_{KN}.

- $\boldsymbol{\mathcal{L}^\infty}$ **norm**

  This normalization imposes $0 \leq \alpha_j \leq 1$ for each $j \in N$. Cutting planes generated with this normalization are not facet-inducing for P_{KN}.

3.2 Row generation procedure

A *row generation approach* is an iterative approach where, at each iteration, a *partial separation problem* (the problem which includes only a subset of the constraints (7)) is considered. Let $(\bar{\alpha}, \bar{\beta})$ be an optimal solution of the partial separation problem. If all the feasible solutions of X_{KN} satisfy the inequality $h^T\bar{\alpha} \leq \bar{\beta}$, then $(\bar{\alpha}, \bar{\beta})$ is the optimal solution of the complete separation problem too. Otherwise a new inequality is added to the partial separation problem and the procedure iterates. The main steps of the row generation procedure are summarized below.

Row generation procedure

Step 1 Let $U \subset X_{KN}$ be a subset of the feasible solutions of the knapsack problem. To prevent unboundedness of the separation problem, it can be initialized as $U = \bigcup_{j \in N} \{e^j\}$, where e^j is the j-th unit vector.

Step 2 Solve the partial separation LP over U:

$$\begin{aligned} \theta \quad = \quad & \max_{(\alpha,\beta)} \left[\bar{y}^T\alpha - \beta\right] \\ & v^T\alpha \leq \beta, \qquad\qquad v \in U \\ & \alpha \in S \end{aligned}$$

Let $(\bar{\alpha}, \bar{\beta})$ be its optimal solution.

Step 3 Solve the knapsack problem:

$$\bar{v} = \underset{v \in X_{KN}}{\operatorname{argmax}} \;\; \bar{\alpha}^T v$$

Step 4 If $\bar{v}^T\bar{\alpha} > \bar{\beta}$ then set $U := U \cup \{\bar{v}\}$ and goto *Step 1*.

Step 5 If $\bar{v}^T\bar{\alpha} \leq \bar{\beta}$ then $(\bar{\alpha}, \bar{\beta})$ is the optimal solution of the separation LP and the inequality $\bar{\alpha}^T y \leq \bar{\beta}$ is valid for P_{KN}.

The row generation procedure requires to solve a large number of knapsack problems, so using efficient knapsack algorithms is a key issue. In our computational experiments we used the modification of Pisinger's MINKAP algorithm [22] that combines dynamic programming with bounding and reduction technique.

MINKNAP algorithm requires that all the coefficients of the knapsack problem are integer. However, in our case the objective function coefficients of the problems in the row generation routine can be fractional. Ceselli and Righini [8] modified the MINKNAP algorithm to deal with real coefficients allowing us to solve the problem with the given accuracy.

3.3 Numerical errors

It is well-known that the rounding errors can affect the solution of linear systems and hence of linear programming solvers. Rounding errors can occur in the solution of LP problems on *Step 2* in the row generation and in solution of knapsack problem with fractional objective coefficients on *Step 3*. Such errors can lead to weak or even invalid cuts.

To generate safe cutting planes, the obtained inequalities are post-processed to get the equivalent cuts with integer coefficients and verifying their validity. Let $\bar{\alpha}^T x \leq \bar{\beta}$ be a violated inequality generated by the row generation procedure. The solution $(\check{\alpha}, \check{\beta})$ of the integer linear problem:

$$\begin{aligned} \min_{(\alpha,\beta,t)} \quad & t \\ & \alpha = t\bar{\alpha} \\ & \beta = t\bar{\beta} \\ & \alpha \in Z^n \\ & \beta \in Z \\ & t \geq 1 \end{aligned}$$

returns the inequality $\check{\alpha}^T x \leq \check{\beta}$ which is equivalent to the original cut. The problem has $n+2$ variables and $n+1$ constraints and it can be easily handled by any MIP solver, if n is not too large.

The validity of each inequality is then checked by solving the knapsack problem

$$\nu = \max_{v \in X_{KN}} \check{\alpha}^T v.$$

If $\nu - \check{\beta} \leq 0$, the inequality is valid. Since all the cut coefficients are now integer, the checking problem can be solved by the MINKNAP algorithm, which is free of the possibility of rounding errors because it works on integers. In the subsequent sequential lifting procedure, only the knapsack problems with integer coefficients are used, so they are immune from rounding errors as well.

4 Computational experience

The algorithm was tested on the GAP instances included in the OR-Library [2]. The test-bed consists of five types of instances (A, B, C, D, E) with size from 5×100 to 80×1600. They are named according to their type and size, e.g. d05100 is an instance of type D with $m = 5$ machines and $n = 100$ tasks. As reported in [21], instances of types A and B can be easily solved in a few seconds by MIP solvers, so we did not consider them in our experiments. All the instances of size up to 20×200 were solved to optimality by the Robust Branch-and-Price algorithm of Pigatti et al. [21], with the only exceptions of $d10200$ and $d20200$. No optimal solutions are known for the larger instances. Best known upper bounds yielded by heuristics are reported in [1, 25].

Computational experiments were carried out on a Pentium IV 3.2GHZ PC with 1Gb of RAM and Windows XP operating system. ILOG CPLEX 10.1 is used both as a LP-solver and as a Branch-and-Cut framework.

4.1 Fine-tuning of the exact separation procedure

A preliminary computational experience was devoted to find the right tunings for the exact separation procedure. Table 1 reports on the results obtained by using different normalizations. Instances of type D were used as preliminary benchmarks, since they are the most difficult. The following notation is used in the table:

- $Name$ is the instances name,
- $b = 1$ – results with $b = 1$ normalization,
- $\mathcal{L}^1$ – results with $\mathcal{L}^1$ norm,
- $\mathcal{L}^1_{in}$ – results with inverted $\mathcal{L}^1$ norm,
- $\mathcal{L}^\infty$ – results with $\mathcal{L}^\infty$ norm,
- $\sharp cuts$ – number of generated cuts,
- $\sharp rows$ – total number of rows generated during the separation,
- $Time$ – computation time in seconds.

The results reported in Table 1 show that the different normalizations did not cause great variations in the number of generated cuts. Normalization $\beta = 1$ was adopted in our final experimens, since the cutting planes produced with this normalization are facet-inducing for $P_{KN}(i)$.

To validate the effectiveness of the algorithm, we compared the lower bound returned by exact knapsack separation with that returned by Lifted Cover Inequalities. Columns C reports on the the results yielded by Lifted Cover Ienqualities, columns $C+E$ reports on results from the combination of Lifted Cover and exact knapsack separation cuts, columns E report on results of the exact knapsack separation standalone. Columns *Closed gap* report on the percentage of

$Name$	$\sharp cuts$				$\sharp rows$				$Time$			
	$b=1$	$\mathcal{L}^1$	$\mathcal{L}^1_{in}$	$\mathcal{L}^\infty$	$b=1$	$\mathcal{L}^1$	$\mathcal{L}^1_{in}$	$\mathcal{L}^\infty$	$b=1$	$\mathcal{L}^1$	$\mathcal{L}^1_{in}$	$\mathcal{L}^\infty$
d05100	333	360	373	355	17973	19656	20902	20867	6.50	7.47	7.36	7.70
d10100	715	686	723	690	44521	35351	37289	38408	15.98	14.07	12.45	14.23
d20100	962	1027	1055	970	45176	48294	41089	52845	16.80	18.62	14.38	19.61
d05200	436	402	413	452	26180	20966	22846	22887	14.55	12.90	13.61	13.12
d10200	801	813	857	816	46362	47234	43647	49060	25.58	26.83	24.53	27.11
d20200	1407	1471	1453	1500	90712	94967	90578	91737	49.34	53.94	49.38	58.78
d10400	683	698	696	712	39875	35258	40671	35651	41.31	41.67	45.48	41.79
d20400	1718	1766	1812	1816	134620	140022	124073	145914	141.04	140.47	139.19	148.87

Table 1: Comparison among different normalizations

	Closed gap			$\sharp Cuts$			$Time$		
$Name$	C	$C+E$	E	C	$C+E$	E	C	$C+E$	E
d05100	14.3	57.1	57.1	41	349	333	0.13	6.97	6.50
d10100	13.0	78.3	78.3	61	700	715	0.16	14.02	15.98
d20100	16.7	81.0	81.0	122	975	962	0.31	15.72	16.80
d05200	0.0	80.0	80.0	30	413	436	0.16	13.41	14.55
d10200	9.1	63.6	63.6	94	821	801	0.41	27.92	25.58
d20200	11.5	46.2	46.2	163	1431	1407	0.70	49.98	49.34
d10400	7.7	23.1	23.1	68	709	683	0.81	48.05	41.31
d20400	6.3	25.0	25.0	161	1763	1718	1.89	139.72	141.04

Table 2: Exact knapsack separation vs. separation of Lifted Cover Inequalities

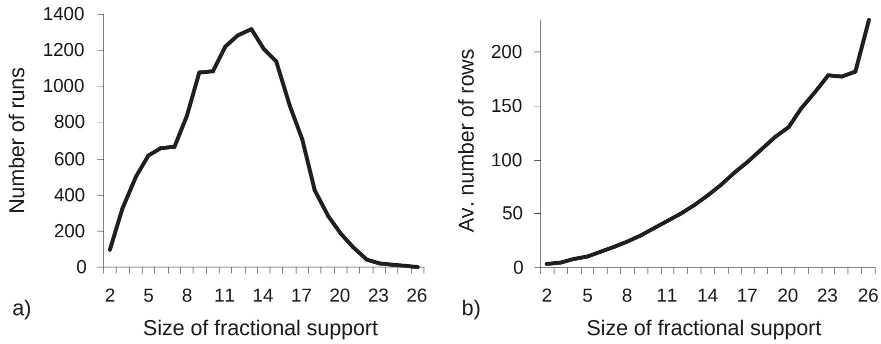

Figure 1: Size of fractional support vs a) number of runs of separation problem of a given size, b) average number of generated rows

$Name$	LP	RG	INT	$LIFT$	SEP	$TOTAL$
d05100	3.07	0.67	0.75	1.17	5.72	6.08
d10100	7.25	1.37	1.47	2.00	12.22	13.84
d20100	6.95	1.64	1.95	2.52	13.18	16.48
d05200	5.17	0.90	1.04	5.90	13.04	14.00
d10200	9.34	1.57	1.92	9.92	22.81	25.41
d20200	17.75	3.19	3.62	14.84	39.58	48.75
d10400	8.02	1.37	1.58	29.45	40.56	43.69
d20400	31.03	5.09	4.73	74.40	115.66	141.04

Table 3: Profiling of cutting plane procedure

closed gap, computed as $\frac{LB-LB_{LP}}{UB-LB_{LP}} \cdot 100\%$, where LB_{LP} is the value of LP relaxation, LB is the obtained lower bound with cuts, UB is the optimal value (best known value for the unsolved instances). It is evident that using Lifted Cover Inequalities does not give any advantage.

A main question is how the time spent by row generation grows with the size of the problems. Fig. 1 shows two diagrams where the size of the fractional support is on the abscissa axis. On the ordinate axis there are: a) the number of separation problems which occurred for a given size for a given size of the fractional support and b) the average number of generated rows. It is interesting to note that the number of generated rows does not grow exponentially and that the number of problems with size more than 20 is small.

Table 3 reports on the profiling of the different modules of the exact knapsack separation procedure. Column LP reports on the time spent to solve the separation LP. RG – time spent to generate rows, i.e. to solve knapsack problems, INT – time spent to make coefficients integer, $LIFT$ – time spent in the sequential lifting, SEP – total time of separation procedure, $TOTAL$

– total time of cutting plane procedure, i.e. SEP plus time spent in solving the LP relaxations.

We can observe that row generation is not a bottleneck for our approach, since it does not take a significant amount of time in the overall cutting plane algorithm. The most time-consuming module is sequential lifting. Nevertheless, experiments also show that avoiding lifting and considering all the variables in the row generation procedure is not practical even on small instances.

4.2 Computational results for the *easy* instances

We term *easy* all the instances which can be solved within three hours of computation time by a Cut-and-Branch algorithm, i.e. where the cutting planes are added only at the root node of the search tree. The results are presented in Table 4. Column *Best known* reports on the best known upper bounds collected from [1, 21, 25], column *Opt* reports on the optimal values (in boldface the values which are better than the best known). Columns *Cut&Branch* reports on the results yielded by our algorithm, where $\sharp Nodes$ is the size of the search tree, $Time$ is the computation time in seconds. By comparing with the results of Pigatti et al. [21] on the instances with $n \leq 200$ (columns *Branch&Price*) we can observe that the algorithm performs comparably to Robust Branch-and-Price.

4.3 Computational results for the *hard* instances

The instances that cannot be solved in less than 3 hours by Cut-and-Branch, are termed *hard*. For their solution we used a Branch-and-Cut algorithm, e.g. cutting planes were generated at every node of the search tree. The node selection strategy was "best first" and the best known upper bounds was used as a cutoff value to prune the nodes in the search tree (MIP heuristics perform very poorly on GAP instances and this is worth of further investigation). We point out that the instances $d10100$ and $d20100$ were previously solved in [21], while the optimal solutions of the other instances were unknown. The best known solution for d10200 was obtained by running Cplex 10.0 for several hours with an "aggressive" setting of the RINS heuristic.

Table 5 reports on the results on these instances obtained with a time-limit of 24 hours of computation time or 1 Gb of memory limit. Column *Gap* reports on the remaining gap, i.e. $\frac{UB-BLB}{UB} \cdot 100\%$, where UB is the best known upper bound (in boldface the values which are better than the best known values), BLB is the best lower bound yielded by enumeration. The instances solved to optimality are marked with *opt*. In addition to $d10100$ and $d20100$ we were able to solve to optimality other 4 instances and significantly reduce the integrality gap for the

	Best	Cut&Branch			Branch&Price	
Name	*known*	*Opt*	♯*Nodes*	*Time*	♯*Nodes*	*Time*
c05100	1931	1931	250	2.39	5	1.17
c10100	1402	1402	325	2.67	47	422.98
c20100	1243	1243	10	1.83	17	1.84
c05200	3456	3456	847	9.03	35	266.17
c10200	2806	2806	1866	17.13	29	2.12
c20200	2391	2391	399	13.98	23	55.38
c10400	5597	5597	729	38.61		
c20400	4782	4782	1139	91.52		
c40400	4244	4244	200	52.38		
c30900	9984	**9982**	42577	3997.56		
c60900	9328	**9326**	35400	8369.69		
d05100	6353	6353	1247	17.08	171	96.30
d05200	12742	12742	624	17.83	57	583.68
e05100	12681	12681	1309	5.33	63	30.09
e10100	11577	11577	624	8.94	31	1305.62
e20100	8436	8436	3406	51.14	87	22.85
e05200	24930	24930	1293	5.69	155	670.62
e10200	23307	23307	11569	39.41	37	6.81
e20200	22379	22379	2040	42.89	21	40.74
e10400	45746	45746	15304	98.91		
e20400	44877	44877	977	79.89		
e40400	44574	**44561**	39068	2875.01		
e15900	102422	**102421**	1800	203.45		
e30900	100434	**100427**	2202	671.51		
e201600	180646	**180645**	4124	1003.47		
e401600	178302	**178293**	9194	4123.61		

Table 4: Computational results on easy instances

	Best	Branch&Cut				
Name	*known*	*BLB*	*UB*	*Gap*	*♯nodes*	*Time*
c15900	11341	11340	11340	opt	4278	2257.37
c201600	18803	18802	18803	0.01	3857	10265.86
c401600	17145	17145	17145	opt	1853	3231.14
c801600	16289	16284	16289	0.03	953	86400.00
d10100	6347	6347	6347	opt	992	385.29
d20100	6185	6185	6185	opt	42623	27783.04
d10200	12430	12430	12430	opt	16222	8921.75
d20200	12244	12233	12244	0.09	5365	18372.98
d10400	24969	24960	24969	0.04	7516	14093.31
d20400	24585	24562	24585	0.09	2493	22258.50
d40400	24417	24350	24417	0.27	1401	86400.00
d15900	55414	55403	55414	0.02	2117	18193.56
d30900	54868	54833	54868	0.06	712	19739.46
d60900	54606	54551	54606	0.10	453	8272.59
d201600	97837	97823	97837	0.01	640	16051.54
d401600	97113	97105	97113	0.01	830	8463.08
d801600	97052	97034	97052	0.02	181	10940.26
e60900	100169	100149	100149	opt	307	6341.15
e801600	176857	176820	176857	0.02	510	23708.65

It is not complete. Computations are in progress.

Table 5: Computational results on hard instances

others.

By analyzing the results of Table 6, where the closed gap of exact separation is presented for the all instances, we can observe that exact separation does not significantly improve the lower bound at the root node for the unsolved instances and this calls for further polyhedral investigation of the GAP polytope.

5 Conclusions

This paper reports on the computational experience with an exact knapsack separation procedure for the Generalized Assignment Problem. The cutting plane algorithm based on this procedure turned out to be quite effective since it could solve to optimality many previously unsolved test instances, dealing with large-scale problems, up to 80×1600.

A main advantage of the proposed approach is that it works with the "natural" formulation of the problem, containing only the original variables. This facilitates the implementation of a

Easy (I)		Easy (II)		Hard	
	Closed		*Closed*		*Closed*
Name	*gap*	*Name*	*gap*	*Name*	*gap*
c05100	85.7	e05100	82.1	c15900	66.7
c10100	85.7	e10100	75.8	c201600	100.0
c20100	95.8	e20100	94.7	c401600	80.0
c05200	80.0	e05200	62.5	c801600	0.0
c10200	80.0	e10200	69.2	d10100	78.3
c20200	100.0	e20200	91.3	d20100	81.0
c10400	80.0	e10400	83.3	d10200	63.6
c20400	85.7	e20400	93.3	d20200	46.2
c40400	100.0	e40400	89.2	d10400	23.1
c30900	100.0	e15900	75.0	d20400	25.0
c60900	90.0	e30900	100.0	d40400	1.4
d05100	57.1	e201600	50.0	d15900	15.4
d05200	80.0	e401600	60.0	d30900	7.7
				d60900	0.0
				d201600	6.7
				d401600	0.0
				d801600	0.0
				e60900	66.2
				e801600	31.6

Table 6: Closed gap (%)

cutting plane algorithm and allows us to use standard Branch-and-Cut frameworks.

Nevertheless there are some points which still have to be addressed to make the approach truly effective.

There are instances where knapsack inequalities derived from a single capacity constraint close only a small percentage of the integrality gap. Such instances point out the need for further investigation of the GAP polytope, aimed at identifying new families of "joint inequalities" involving two or more capacity constraints.

Finally, even if upper bound heuristics are beyond the scope of this paper, we observe that more effective heuristics, and particularly MIP heuristics, could significantly reduce computation time for the hard instances or even lead to solve to optimality some of the remaining unsolved instances.

Acknowledgements

The authors wish to thank Alexandre Pigatti, Marcus Poggi de Aragão and Eduardo Uchoa for providing us with their Robust Branch-and-Price code, and Adam Letchford for his comments and suggestions. We are indebted with Laurence A. Wolsey for his constant support.

References

[1] Y. Asahiro, M. Ishibashi, and M. Yamashita. Independent and cooperative parallel search methods for the generalized assignment problem. *Optimization Methods and Software*, 18(2):129–141, 2003.

[2] J.E. Beasley. Or-library: distributing test problems by electronic mail. *Journal of Operational Research Society*, 41(11):1069–1072, 1990.

[3] M.P. Bonami. *Étude et mise en oeuvre dapproches polyédriques pour la résolution de programmes en nombres entiers ou mixtes généraux.* PhD thesis, L'Université Paris 6, 2003.

[4] E.A. Boyd. Generating fenchel cutting planes for knapsack polyhedra. *SIAM Journal of Optimization*, 3:734–750, 1993.

[5] E.A. Boyd. Solving integer programs with cutting planes and preprocessing. *IPCO 1993*, pages 209–220, 1993.

[6] E.A. Boyd. Fenchel cutting planes for integer programming. *Operations Research*, 42:53–64, 1994.

[7] E.A. Boyd. On the convergence of fenchel cutting planes in mixed-integer programming. *SIAM Journal of Optimization*, 5:421–435, 1995.

[8] A. Ceselli and G. Righini. A branch and price algorithm for the capacitated p-median problem. *Networks*, 45(3):125–142, 2004.

[9] P.C. Chu and J.E. Beasley. A genetic algorithm for the generalised assignment problem. *Computers and Operations Research*, 24:17–23, 1997.

[10] G. Cornuejols and C. Lemarechal. A convex-analysis perspective on disjunctive cuts. *Mathematical Programming*, 106(3):567–586, 2006.

[11] I.R. de Farias and G.L. Nemhauser. A family of inequalities for the generalized assignment polytope. *Operations Research Letters*, 29:49–51, 2001.

[12] D. Espinoza. *Étude et mise en oeuvre dapproches polyédriques pour la résolution de programmes en nombres entiers ou mixtes généraux*. PhD thesis, L'Université Paris 6, 2003.

[13] R. Fukasawa and M. Goycoolea. On the exact separation of mixed integer knapsack cuts. In Springer, editor, *Proceedings of the 2007 Integer Programming and Combinatorial Optimization conference*, To appear.

[14] E.S. Gottlieb and M. R. Rao. $(1, k)$-configuration facets for the generalized assignment problem. *Mathematical Programming*, 46:53–60, 1990.

[15] E.S. Gottlieb and M. R. Rao. The generalized assignment problem: Valid inequalities and facets. *Mathematical Programming*, 46:31–52, 1990.

[16] J. J. Nowakovski, W. Schwrzler, and E. Triesch. Using the generalized assignment problem in scheduling the rosat space telescope. *European Journal of Operations Research*, 112:531–541, 1999.

[17] K. Kaparis and A.N Letchford. Separation algorithms for 0-1 knapsack polytopes. *In preparation. Available at* $http://www.lancs.ac.uk/staff/letchfoa/publications.htm$, 2007.

[18] M. Laguna, J.P. Kelly, J.L. Conzlez-Velarde, and F. F. Glover. Tabu search for the generalized assignment problem. *European Journal of Operations Research*, 82:176–189, 1995.

[19] R.M. Nauss. Solving the generalized assignment problem: An optimizing and heuristic approach. *INFORMS Journal on Computing*, 15(3):249–266, 2003.

[20] I.H. Osman. Heuristics for the generalized assignment problem: Simulated annealing and tabu search approaches. *OR Spektrum*, 17:211–225, 1995.

[21] A. Pigatti, M. Poggi de Aragao, and E. Uchoa. Stabilized branch-and-cut-and-price for the generalized assignment problem. In *2nd Brazilian Symposium on Graphs, Algorithms and Combinatorics, Electronic Notes in Discrete Mathematics, Vol. 19*, pages 385–395, 2005.

[22] D. Pisinger. A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, 46(0):758–767, 1995.

[23] M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations research*, 45(6):831–841, 1997.

[24] M. Yagiura, T. Ibaraki, and F. Glover. An ejection chain approach for the generalized assignment problem. *INFORMS Journal on Computing*, 16:133–151, 2004.

[25] M. Yagiura, T. Ibaraki, and F. Glover. A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research*, 169:548–569, 2006.

[26] M. Yagiura, T. Yamaguchi, and T. Ibaraki. A variable depth search algorithm with branching search for the generalized assignment problem. *Optimization Methods and Software*, 10:419–441, 1998.