Drexl, Andreas; Jørnsten, Kurt

Pricing the generalized assignment problem

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 627

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 627

Pricing the generalized assignment problem

Andreas Drexl, Kurt Jørnsten

June 2007

Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität, Kiel, Germany,
andreas.drexl@bwl.uni-kiel.de

Norwegian School of Economics and Business Administration, Department of Finance
and Management Science, Bergen, Norway, kurt.jornsten@nhh.no

Abstract

The generalized assignment problem (GAP) examines the maximum profit assignment of jobs to processors such that each job is assigned to precisely one processor subject to capacity restrictions on the processors. Due to the fact that the GAP is an NP-hard integer program dual prices are not readily available. In this paper we propose a family of linear programming models the optimal solution of which is integral "almost always". We provide a computational proof of this conjecture by an in-depth experimental study of 1500 instances generated according to the standard procedure adopted in literature. Summarizing this analysis we have linear prices for all but 17 of the whole bunch of instances and, hence, there exists a linear price function that supports the optimal assignment of jobs to processors.

Keywords: Generalized assignment problem, integer programming, duality, linear programming, pricing

1 Introduction

Given a set of m jobs and a set of n processors, the generalized assignment problem (GAP) consists of finding the most profitable assignment of each job to a single processor that respects the capacity constraints on the processors. Although interesting in its own right, its main importance stems from the fact that it appears as a substructure in many models developed to solve real-world problems in different applications such as facility location (see, e.g., [10, 15]), flexible manufacturing (see, e.g., [11, 12]), and vehicle routing (see, e.g., [1]).

In literature the GAP is defined in several ways. Here we adopt the following convention: Let $M = \{1, \ldots, m\}$ denote the set of processors and $N = \{1, \ldots, n\}$ the set of jobs. Given $i \in M$ and $j \in N$, $\kappa_i \in \mathbb{Z}_+$ denotes the capacity of processor i, $w_{ij} \in \mathbb{Z}_+$ the claim on the capacity of processor i by job j, and $p_{ij} \in \mathbb{Z}_+$ the profit of assigning job j to processor i. Using the 0-1 variable $x_{ij} = 1$ indicating whether job j is assigned to processor i ($x_{ij} = 1$) or not ($x_{ij} = 0$) we get the linear integer program (1).

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} \tag{1a}$$

$$\text{s.t.} \sum_{i=1}^{m} x_{ij} = 1 \qquad \text{for } j = 1, \ldots, n \tag{1b}$$

$$\sum_{j=1}^{n} w_{ij} x_{ij} \leq \kappa_i \qquad \text{for } i = 1, \ldots, m \tag{1c}$$

$$x_{ij} \in \{0, 1\} \qquad \text{for } i = 1, \ldots, m, j = 1, \ldots, n \tag{1d}$$

The objective function (1a) is to maximize profit. Constraints (1b) assure that each job is assigned to exactly one processor. Constraints (1c) ensure that the capacity of each processor is respected.

If we relax the integrality constraint (1d) on x_{ij} we obtain the linear programming relaxation (2), used in section 4 for benchmarking purposes.

$$\max \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij} \tag{2a}$$

$$\text{s.t.} \sum_{i=1}^{m} x_{ij} = 1 \qquad \text{for } j = 1,\ldots,n \tag{2b}$$

$$\sum_{j=1}^{n} w_{ij}x_{ij} \leq \kappa_i \qquad \text{for } i = 1,\ldots,m \tag{2c}$$

$$x_{ij} \geq 0 \qquad \text{for } i = 1,\ldots,m, j = 1,\ldots,n \tag{2d}$$

The GAP is easily shown to be NP-hard and a considerable body of literature exists on the search for effective exact (see, e.g., [5, 8, 9, 13, 16]) and heuristic (see, e.g., [3, 14, 17]) algorithms. Polyhedral results have been presented in [2, 4, 6, 7].

The outline of the paper is as follows: In section 2 we present a family of linear programming models the solution of which is integral "almost always". An instance is used in Section 3 for illustrative purposes. In section 4 we provide a computational proof of the "almost always" conjecture. Section 5 concludes the paper.

2 Variable elimination/aggregation

The formulation of the family of linear programming models can be accomplished in two steps. First, we solve the integer program (1). Then we eliminate all variables corresponding to the chosen job-processor assignments and introduce instead an artificional assignment of all the jobs to a fictitional processor by means of column aggregation.

More precisely, assume that we know an optimal solution $(x_{ij}^{(1)})$ of (1). Let

$$X_0 = \left\{(i,j) \in M \times N : x_{ij}^{(1)} = 0\right\}$$

denote the set of variables which have been fixed to zero. Likewise,

$$X_1 = \left\{(i,j) \in M \times N : x_{ij}^{(1)} = 1\right\}$$

denotes the set of variables which have been fixed to one.

The reformulation is based on the idea that we eliminate the variables (jobs) contained in X_1 and that we introduce instead an additional variable. This variable contributes the amount

$$\bar{p} = \sum_{(i,j)\in X_1} p_{ij}$$

to the optimal objective function value and it needs

$$\bar{\kappa}_i = \sum_{j:(i,j)\in X_1} w_{ij}$$

of the capacity κ_i of processor $i \in M$ and, hence, $\delta_i = \kappa_i - \bar{\kappa}_i$ is the portion (slack) of the capacity of processor $i \in M$ not used in the optimal solution.

A linear program based on the idea of variable elimination/aggregation, that is, which solely uses the variables X_0 and the variable z, is provided in (3).

$$\max \sum_{(i,j)\in X_0} p_{ij}x_{ij} + \bar{p} \cdot z \tag{3a}$$

$$\text{s.t.} \sum_{i:(i,j)\in X_0} x_{ij} + z = 1 \qquad \forall j \in N \tag{3b}$$

$$\sum_{j:(i,j)\in X_0} w_{ij}x_{ij} + \bar{\kappa}_i z \leq \kappa_i \qquad \forall i \in M \tag{3c}$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in X_0 \tag{3d}$$

$$z \geq 0 \tag{3e}$$

The objective function (3a) comprises the contribution of the variable z and of the variables contained in X_0. Constraint (3b) assures that each job is either assigned to z or to one of the processors which has not been chosen in the optimal solution. Constraint (3c) assures that the capacity κ_i of processor $i \in M$ is not exceeded by the weights associated with the jobs covered by X_0 and z. Finally, (3d) and (3e) define the decision variables to be nonnegative.

Apparently, in terms of the optimal solution $(x_{ij}^{(1)})$ in general we have positive slack in constraint (3c). If we reduce the right-hand-side κ_i to $\bar{\kappa}_i$ we get the linear program (4).

$$\max \sum_{(i,j)\in X_0} p_{ij}x_{ij} + \bar{p} \cdot z \tag{4a}$$

$$\text{s.t.} \sum_{i:(i,j)\in X_0} x_{ij} + z = 1 \qquad \forall j \in N \tag{4b}$$

$$\sum_{j:(i,j)\in X_0} w_{ij}x_{ij} + \bar{\kappa}_i z \leq \bar{\kappa}_i \qquad \forall i \in M \tag{4c}$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in X_0 \tag{4d}$$

$$z \geq 0 \tag{4e}$$

Constraint (4c) is more tight than constraint (3c) and, hence, we can expect that model (4) will produce an integral solution more often than model (3). However, the dual prices then will not reflect the given right-hand-side of each processor but only the amount used in the optimal solution.

Fortunately, we can improve model (3) by attaching the "optimum" slack δ_i to the capacity $\bar{\kappa}_i$ used in the optimum solution, that is, make use of the fact that $\kappa_i = \bar{\kappa}_i + \delta_i$ for all $i \in M$ trivially is valid. Doing so we get the linear program (5).

$$\max \sum_{(i,j)\in X_0} p_{ij}x_{ij} + \bar{p}\cdot z \tag{5a}$$

$$\text{s.t.} \sum_{i:(i,j)\in X_0} x_{ij} + z = 1 \qquad \forall j \in N \tag{5b}$$

$$\sum_{j:(i,j)\in X_0} w_{ij}x_{ij} + \kappa_i z \leq \kappa_i \qquad \forall i \in M \tag{5c}$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in X_0 \tag{5d}$$

$$z \geq 0 \tag{5e}$$

In section 4 we will show be means of a computational study how the models (3) to (5) behave in terms of integrality of optimal solutions and dual degeneracy. Before we illustrate the idea using an example.

3 Illustrative example

Consider the following instance (of the type D generated according to the specification given in section 4) with three processors and eight jobs:

$$(p_{ij}) = \begin{pmatrix} 52 & 15 & 15 & 24 & 47 & 77 & 79 & 108 \\ 34 & 39 & 31 & 12 & 106 & 81 & 1 & 68 \\ 95 & 67 & 78 & 29 & 101 & 28 & 94 & 63 \end{pmatrix}$$

$$(w_{ij}) = \begin{pmatrix} 49 & 16 & 20 & 34 & 56 & 72 & 80 & 99 \\ 28 & 38 & 22 & 16 & 100 & 75 & 11 & 73 \\ 90 & 70 & 82 & 23 & 95 & 23 & 85 & 62 \end{pmatrix}, \quad (\kappa_i) = \begin{pmatrix} 113 \\ 96 \\ 141 \end{pmatrix}$$

An optimal solution of the integer program (1) with objective function value 309 is:

$$(x_{ij}^{(1)}) = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

An optimal solution of the linear programming relaxation (2) with objective function value 387.99 is:

$$(x_{ij}^{(2)}) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.98 \\ 1 & 0 & 1 & 0 & 0.40 & 0 & 0.57 & 0 \\ 0 & 0 & 0 & 1 & 0.60 & 1 & 0.43 & 0.02 \end{pmatrix}$$

An optimal solution of the aggregate model (3) with objective function value 359.93 is z=0.45 and:

$$(x_{ij}^{(3)}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0.53 & 0.54 & 0 & 0.05 \\ 0.54 & 0.54 & 0.54 & 0.54 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.54 & 0.49 \end{pmatrix}$$

An optimal primal solution of the aggregate model (4) is z=1 and x_{ij}=0 for all $(i,j) \in X_0$ with objective function value 309.

An optimal dual solution of (4) is

$$(u_j) = (-5.45, -14.54, -0, -10.54, -21.80, -11.46, -5.02, -9.23)$$

and

$$(v_i) = (1.23, 1.41, 1.16)$$

where u_j is the dual variable corresponding to the jth job completion constraint and v_i is the dual variable corresponding to the ith knapsack constraint.

The optimal solution of (5) equals the one for (4). An optimal dual solution of (5) is

$$(u_j) = (-56.03, -83.18, -39.74, -39.44, -137.45, -160.15, -88.36, -70.02)$$

and

$$(v_i) = (3.29, 3.22, 2.15).$$

The first price vector (u_j) contains one dual variable with value 0 while the second price vector (u_j) contains none. Contrary to this particular observation the computational results presented below will show that dual degeneracy of model (5) in general is larher than that of model (4). Of course, the dual solution chosen depends on the particular solver used (in our case Cplex).

The dual variables can be used for economic reasoning in many ways. One important question in the presence of scarce processor resources is whether to increase it or not and to what extent. The dual prices $(v_i) = (3.29, 3.22, 2.15)$ suggest to proceed as follows: First of all, we can compare the price of each processor with the particular (overtime) cost per unit of expanding the capacity. If we tentatively expand κ_1 to 114, the optimal solution does not change, but if we increase it to 115 we get a new solution worth 366. Doing the same for processor 2 we have to increase κ_2 by 3 units to 99 in order to get a new optimal solution with value 371. For processor 3 we have to increase κ_3 to 155 units in order to get a new optimal solution with value 337.

Recall that the prices $(v_i) = (1.23, 1.41, 1.16)$ produced by model (4) reflect the used capacity, that is, $(\bar{\kappa}_i) = (85, 84, 141)$, while the prices $(v_i) = (3.29, 3.22, 2.15)$ produced by model (5) reflect $(\kappa_i) = (113, 96, 141)$. In this sense $v_2 > v_1$ indicated by model (4) might falsely signal that it is more profitable to expand the capacity of processor 2 compared to processor 1. However, if model (5) perhaps does not yield an integral solution it might be the only choice to use the the prices produced by model (4). The computational study presented in the following section is going to address the question which model is best in terms of producing integral solutions.

4 Computational results

In this section we provide the results of an in-depth computational study in order to show that the models (3) to (5) in general have different optimal (primal and/or dual)

solutions. Furthermore, we show that the degree of dual degeneracy associated with the optimal primal solutions in general also differs.

The models described earlier have been imlemented in Java using the Cplex callable library (version 9.0; all parameters with default values) on an AMD Athlon with 2 GB RAM and 2.1 Ghz clockpulse running under the operating system Linux.

In literature, algorithms usually are tested on four classes of random problems, usually referred to as A, B, C, and D, generated according to the following scheme (see for instance [16]):

A. p_{ij} and w_{ij} are integer from a uniform distribution between 10 and 25 and between 5 and 25, respectively. $\kappa_i = \left\lfloor 9(n/m) + 0.4 \max_{h \in M} \sum_{j \in N_h^*} w_{ij} \right\rfloor$, where $N_h^* = \{j \in N : h = \text{argmin}_{r \in N} p_{rj}\}$.

B. Same as A for p_{ij} and w_{ij}. κ_i equals 0.7 of κ_i in A.

C. Same as A for p_{ij} and w_{ij}. $\kappa_i = 0.8/m \sum_{j \in N} w_{ij}$.

D. Same as C for κ. w_{ij} is integer from a uniform distribution between 1 and 100. $p_{ij} = 100 - w_{ij} + l$ where l is integer from a uniform distribution between 1 and 21.

This scheme produces instances for the minimization variant of the GAP. Since our algorithm handles the maximization variant of the GAP, all instances are converted to the maximization form by the following transformation: Let $t = \max_{i \in M, j \in N} p_{ij} + 1$ .We replace p_{ij} by $t - p_{ij}$ for all $i \in M$ and $j \in N$.

We have generated 100 instances at random for each of the four types. For each instance an optimal solution of the integer program (1) has been computed using Cplex. Note that while in gerenal instances of type A, B and C can be solved very quickly to optimality one single instance of type D can take hours or even days of computation.

The results for the problem type A are displayed in table 1 for different problem sizes m and n. For each of the 4 considered models we present the average number of times where the solution of the linear program turned out to be integral (indicated by LP=IP). For the subset of instances with integral solutions we report the percentage average number of dual variables equal to zero (#DV0(%) used as abbreviation) in order to give an indication of the degree of dual degeneracy observed. The last row displays average values for each of the columns. For the problem types B and C the results are displayed in tables 2 to 3 in the same way.

The results can be summarized as follows:

- Pricing A instances seems to be most easy (see table 1). Here in a couple of cases the linear programming relaxation (2) is integral. Moreover, all the three model reformulations yield in all cases an integral solution. The solution of the models (3) and (5) show much larger dual degeneracy than model (4).
- The picture for B is very much the same as for A, except the fact that the linear programming relaxation (2) is integral only once (see table 2).

		model (2)		model (3)		model (4)		model (5)	
m	n	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)
5	30	3	13.3	100	13.7	100	7.4	100	12.9
	50	1	9.1	100	8.7	100	3.3	100	8.3
	70	2	6.7	100	6.5	100	1.6	100	5.8
	90	1	5.3	100	5.1	100	2.1	100	4.5
10	30	0	–	100	24.5	100	0.4	100	25.0
	50	0	–	100	16.3	100	0.1	100	16.5
	70	0	–	100	12.2	100	0.1	100	12.4
	90	0	–	100	9.8	100	0.1	100	9.8
20	30	0	–	100	39.4	100	0.6	100	40.0
	50	0	–	100	28.2	100	0.3	100	28.6
	70	0	–	100	21.9	100	0.4	100	22.2
	90	0	–	100	18.0	100	0.2	100	18.2
averages		0.6	8.9	100	16.0	100	1.0	100	16.0

Table 1: Type A Instances

		model (2)		model (3)		model (4)		model (5)	
m	n	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)
5	30	1	14.3	100	12.8	100	7.6	100	9.8
	50	0	–	100	7.8	100	5.4	100	3.9
	70	0	–	100	5.6	100	3.6	100	5.1
	90	0	–	100	4.1	100	2.5	100	3.3
10	30	0	–	100	23.8	100	0.5	100	21.3
	50	0	–	100	15.9	100	0.1	100	10.5
	70	0	–	100	11.7	100	0.1	100	6.8
	90	0	–	100	9.5	100	0.1	100	5.3
20	30	0	–	100	39.2	100	0.3	100	40.0
	50	0	–	100	28.0	100	0.4	100	28.6
	70	0	–	100	21.7	100	0.4	100	22.0
	90	0	–	100	17.8	100	0.3	100	17.8
averages		0.1	14.3	100	15.5	100	1.4	100	13.6

Table 2: Type B Instances

		model (2)		model (3)		model (4)		model (5)	
m	n	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)
5	30	0	–	97	10.4	100	33.7	100	26.5
	50	0	–	100	5.8	100	24.0	100	22.7
	70	0	–	100	4.0	100	17.3	100	15.7
	90	0	–	100	3.3	100	15.9	100	14.5
10	30	0	–	56	20.5	100	8.6	100	2.3
	50	0	–	98	12.4	100	2.3	100	1.0
	70	0	–	100	9.1	100	1.4	100	0.8
	90	0	–	100	6.4	100	0.8	100	0.6
20	30	0	–	0	–	100	4.5	75	0.0
	50	0	–	23	23.4	100	0.1	100	0.0
	70	0	–	83	18.2	100	0.0	100	0.0
	90	0	–	99	14.4	100	0.0	100	0.0
averages		0.0	–	79.7	9.3	100	7.1	97.9	6.0

Table 3: Type C Instances

Table 4: Type D Instances

m	n	model (2)		model (3)		model (4)		model (5)	
		LP=IP	#DV0 (%)	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)	LP=IP	#DV0 (%)
5	30	0	–	0	–	100	3.5	100	2.7
	50	0	–	1	0.0	100	2.9	100	3.2
	70	0	–	12	0.4	83	1.8	83	2.0
averages		0.0	–	4.3	0.3	94.3	2.8	94.3	3.2

- For the instance type C (see table 3) we do not have a unique picture (except the fact that the linear programming relaxation (2) is never integral). Once more models (4) and (5) in general are more powerful in terms of producing integral solutions than model (3); additionally model (4) is slightly superior to model (5). Comparing (4) and (5) in terms of dual degeneracy model (5) is slightly superior to model (4). Interestingly, dual degeneracy decreases with increasing problem size.

- For the problem type D we aborted computation prematurely after some hours of CPU-time per instance. Unfortunately, instances with 5 processors and 90 jobs or with 10 processors and 30 jobs or more cannot be solved to optimality in a reasonable amount of time. Fortunately, only for 17 of the 300 optimally solved integer programs the aggregated models (4) and (5) do not provide an integral solution.

Summarizing, our linear programs have provided integral solutions for 1483 of the 1500 ($= 3 \cdot 1200 + 300$) instances studied. Hence, we have given a computational proof of the "almost always" conjecture.

5 Summary and future work

In this paper we have provided a family of linear programming models for the GAP the solution of which is "almost always" integral. In particular, for three out of four instance types usually studied in literature at least one of the models produces an integral solution. Hence, for these instances dual prices are readily available.

Subsequently we will enhance the linear programs by valid inequalities so as to get linear prices also for the difficult instances of the type D.

Acknowledgement

The authors are indebted to Christof Kluß for professionally coding the algorithms.

References

[1] BRAMEL, J., SIMCHI-LEVI, D. (1995), A location based heuristic for general routing problems, Operations Research, Vol. 43, pp. 649–660

[2] CATTRYSSE, D.G., DEGRAEVE, Z., TISTAERT, J. (1998), Solving the generalized assignment problem using polyhedral techniques, European Journal of Operational Research, Vol. 108, pp. 618–628

[3] DÍAZ, J.A., FERNÁNDEZ, E. (2001), A tabu search algorithm for the generalized assignment problem, European Journal of Operational Research, Vol. 132, pp. 22–38

[4] DE FARIAS, I.R.JR., NEMHAUSER, G.L. (2001), A family of inequalities for the generalized assignment problem, Operations Research Letters, Vol. 29, pp. 49-55

[5] FISHER, M.L., JAIKUMAR, R., VAN WASSENHOWE, L.N. (1986), A multiplier adjustment method for the generalized assignment problem, Management Science, Vol. 32, pp. 1095–1103

[6] GOTTLIEB, E.S., RAO, M.R. (1990), The generalized assignment problem: valid inequalities and facets, Mathematical Programming, Vol. 46, pp. 31–52

[7] GOTTLIEB, E.S., RAO, M.R. (1990), $(1, k)$-configuration facets for the generalized assignment problem, Mathematical Programming, Vol. 46, pp. 53–60

[8] GUIGNARD, M., ROSENWEIN, M.B. (1989), An improved dual based algorithm for the generalized assignment problem, Operations Research, Vol. 37, pp. 658–663

[9] JÖRNSTEN, K., NÄSBERG, M. (1986), A new Lagrangian relaxation approach to the generalized assignment problem, European Journal of Operational Research, Vol. 27, pp. 313–323

[10] KLASTORIN, T.D. (1979), On the maximal covering location problem and the generalized assignment problem, Management Science, Vol. 25, pp. 107–112

[11] KUHN, H. (1995), A heuristic algorithm for the loading problem in flexible manufacturing systems, The International Journal of Flexible Manufacturing Systems, Vol. 7, pp. 229–254

[12] LEE, D.-H., KIM, Y.-D. (1998), A multi-period order selection problem in flexible manufacturing systems, Journal of the Operational Research Society, Vol. 49, pp. 278–286

[13] NAUSS, R.M. (2003), Solving the generalized assignment problem: an optimizing and heuristic approach, INFORMS Journal on Computing, Vol. 15, pp. 249–266

[14] ROMEIJN, H.E., ROMERO MORALES, D. (2000), A class of gready algorithms for the generalized assignment problem, Discrete Applied Mathematics, Vol. 103, pp. 209–235

[15] ROSS, G.T., SOLAND, R.M. (1977), Modelling facility location problems as generalized assignment problems, Management Science, Vol. 24, pp. 345–357

[16] SAVELSBERGH, M.W.P. (1997), A branch-and-price algorithm for the generalized assignment problem, Operations Research, Vol. 45, pp. 831–841

[17] YAGIURA, M., IBARAKI, T., GLOVER, F. (2004), An ejection chain approach for the generalized assignment problem, INFORMS Journal on Computing, Vol. 16, pp. 133–151