# Stabilized Branch-and-cut-and-price for the Generalized Assignment Problem *

Alexandre Pigatti, Marcus Poggi de Aragão
Departamento de Informática, PUC do Rio de Janeiro
{apigatti, poggi}@inf.puc-rio.br

Eduardo Uchoa
Departamento de Engenharia de Produção, Universidade Federal Fluminense
uchoa@producao.uff.br

October, 2004

**Abstract**

The Generalized Assignment Problem (GAP) is a classic scheduling problem with many applications. We propose a branch-and-cut-and-price for that problem featuring a stabilization mechanism to accelerate column generation convergence. We also propose ellipsoidal cuts, a new way of transforming the exact algorithm into a powerful heuristic, in the same spirit of the cuts recently proposed by Fischetti and Lodi. The improved solutions found by this heuristic can, in turn, help the task of the exact algorithm. The resulting algorithms showed a very good performance and were able to solve three among the last five open instances from the OR-Library.

## 1 Introduction

The Generalized Assigment Problem (GAP) is defined as follows. Let $I = \{1, ..., m\}$ be a set of machines and $J = \{1, ..., n\}$ a set of tasks. Each machine $i \in I$ has a capacity $b_i$. Each task $j \in J$ when assigned to machine $j \in J$ consumes $a_{ij}$ units of capacity and implies in a cost of $c_{ij}$ units. One searches for an assignment of every task to one of the machines, respecting the machine capacities and minimizing the total cost. This classical NP-hard problem has many applications in industrial scheduling. Some recent applications of the GAP in other contexts include the allocation of patients to medical flies in the US army [10] and in the operation of the international spatial telescope ROSAT [7].

*The results in this article were already presented in [8]

The current best exact algorithms for the GAP appear to be the branch-and-price by Savelsbergh [11] and the branch-and-cut by Farias and Nemhauser [2]. Good heuristics include [5][1][13][12].

A natural formulation for the GAP is defined over binary variables $x_{ij}$, meaning that task $i$ is assigned to machine $j$:

$$\text{F1} \begin{cases} \min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} & & (1) \\ \text{s.t.} \quad \sum_{j=1}^{n} a_{ij} x_{ij} \leq b_i & i \in I & (2) \\ \sum_{i=1}^{m} x_{ij} = 1 & j \in J & (3) \\ x_{ij} \in \{0,1\} & i \in I,\ j \in J & (4) \end{cases}$$

The bounds obtained by solving the linear relaxation of this formulation are not good on many instances. A stronger formulation, with an exponential number of variables, can be obtained as follows. For a given machine $i$, let $v_i^k = \left\{v_{i1}^k, v_{i2}^k, \ldots, v_{in}^k\right\}$ be a 0-1 vector representing one of the $K_i$ possible solutions of $\sum_{j=1}^{n} a_{ij} v_{ij} \leq b_i; \quad v_{ij} \in \{0,1\}, \ j \in \{1,\ldots,n\}$. In other words, $v_{ij}^k = 1$ if task $j$ is assigned to machine $i$ in the allocation with number $k$, $1 \leq k \leq K_i$. Let $y_i^k$ be a binary variable indicating if allocation $v_i^k$ is selected to machine $i$. The GAP can be reformulated as follows:

$$\text{F2} \begin{cases} \min \sum_{i=1}^{m} \sum_{k=1}^{K_i} \left( \sum_{j=1}^{n} c_{ij} v_{ij}^k \right) y_i^k & & (1) \\ \text{s.t.} \quad \sum_{i=1}^{m} \sum_{k=1}^{K_i} v_{ij}^k y_i^k = 1 & j \in J & (2) \\ \sum_{k=1}^{K_i} y_i^k \leq 1 & i \in I & (3) \\ y_i^k \in \{0,1\} & i \in I,\ k \in \{1,\ldots,K_i\} & (4) \end{cases}$$

The exponential number of variables in F2 requires the use of column generation. The pricing subproblem is a binary knapsack problem, which is NP-hard but can be solved by pseudo-polinomial algorithms with an excellent practical performance [6].

A fractional solution of the linear relaxation of F2 can be converted into a fractional solution of F1 by the relation $x_{ij} = \sum_{1 \leq k \leq K_i} v_{ij}^k y_i^k$. Therefore one can branch over $x$ variables, even while working with F2. This leads to a branch-and-price algorithm, as the one proposed in [11]. Moreover, as indicated in [9], a generic cut over $x$ variables $\sum_{i \in I} \sum_{j \in J} d_{ij} x_{ij} \leq e$ can be included in F2 as $\sum_{i=1}^{m} \sum_{k=1}^{K_i} \left( \sum_{j=1}^{n} d_{ij} v_{ij}^k \right) y_i^k \leq e$. The dual variable corresponding to this new cut will not change the structure of the pricing subproblem. When both column generation and cut separation are performed we have a branch-and-cut-and-price (BCP) algorithm.

Our BCP algorithm included the family of cuts used in [2]. Unhappily, those cuts (which are effective in their branch-and-cut) had little impact

in bound quality. However, the BCP mechanism developed was used with success to introduce the ellipsoidal cuts, as discussed in Section 4.

## 2 Stabilizing the column generation

Column generation is known to be prone to convergence difficulties. Worse, on some situations the convergence can be erratic: even when solving instances of the same size, the number of iterations can vary by one or two orders of magnitude. This can be a serious obstacle when implementing branch-and-price and BCP algorithms. Du Merle et al. [3] proposed a *dual stabilization* technique to alleviate this, based on a simple observation: the columns that will be part of the final solution are only generated in the very last iterations, when the dual variables are already close to their optimal values. They also observed that dual variables may oscillate wildly in the first iterations, leading to "extreme columns" that have no chance of being in the final solution.

In this work we introduce a simplified variant of this technique that was crucial in our BCP for the GAP. We explain this variant in a general setting. Suppose we have a linear problem $P$ with a large number of variables and let $\pi$ be the corresponding vector of dual variables.

$$P = \left\{ \begin{array}{lll} \min & f = \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} & \sum_{j=1}^{n} \alpha_{ij} x_j = b_i \quad i = 1, \ldots, m & \\ & x_j \geq 0 \quad j = 1, \ldots, n & \end{array} \right.$$

Let $\bar{\pi}$ be an educated guess about the values in some optimal dual solution and $\epsilon$ a parameter. Instead of solving $P$, we solve (by column generation) the following stabilized problem:

$$P(\bar{x}, \epsilon) = \left\{ \begin{array}{lll|l} \min & f(\bar{x}, \epsilon) = \sum_{j=1}^{n} c_j x_j - \sum_{i=1}^{m} \bar{\pi}_i w_i + \sum_{i=1}^{m} \bar{\pi}_i z_i & & \\ \text{s.t.} & \sum_{j=1}^{n} \alpha_{ij} x_j - w_i + z_i = b_i \quad i = 1, \ldots, m & & \pi \text{ irrestricted} \\ & 0 \leq w_i \leq \epsilon \quad i = 1, \ldots, m & & \omega_i \geq 0 \\ & 0 \leq z_i \leq \epsilon \quad i = 1, \ldots, m & & \zeta_i \geq 0 \\ & x_j \geq 0 \quad j = 1, \ldots, n & & \end{array} \right.$$

We have introduced two sets of artificial variables. The $w$ variables have coefficients corresponding to a negative identity and the $z$ to a positive identity. Such variables have costs $\bar{\pi}$ and are restricted to be lesser or equal to $\epsilon$. The dual variables corresponding those new constraints (as shown in the right of $P(\bar{x}, \epsilon)$) are $\omega$ and $\zeta$. It can be seen that, for any values of $\bar{x}$ and $\epsilon$, $f(\bar{x}, \epsilon) \leq f$. Solving $P(\bar{x}, \epsilon)$ by column generation may require much less

iterations than $P$. This can be explained by the fact that the dual variables $\pi$ are now subject to the following constraints:

$$\bar{\pi}_i - \omega_i \leq \pi_i \leq \bar{\pi}_i + \zeta_i \quad i = 1, \ldots, m.$$

In other words, when $\pi_i$ deviates from $\bar{\pi}_i$ there is a penalization of $\epsilon \cdot |\bar{\pi}_i - \pi_i|$ units. Therefore, $\pi$ is not likely to oscillate much from one iteration to another. Moreover, if we are very lucky in our guess and $\bar{\pi}$ is an optimal dual solution or if $\epsilon = 0$, $f(\bar{x}, \epsilon) = f$.

We apply this stabilization in our BCP in the following way. In the root node, we start guessing $\bar{\pi}$ by solving the linear relaxation of F1 and taking the optimal dual variables of constraints (F1.3). Then we solve $P(\bar{x}, 0.1)$ as the first step. The second step takes the optimal values of $\pi$ from the first step as the new guess to $\bar{\pi}$ and solves $P(\bar{x}, 0.01)$. The third step updates $\bar{\pi}$ with the solution of the second step and solves $P(\bar{x}, 0.001)$. In the last step, we solve $P(\bar{x}, 0)$, which is equivalent to solving $P$. To make possible a fast convergence, it is necessary to keep in the LP all columns generated from previous steps. In fact, solving the fourth step only differs from solving $P$ from scratch due to the presence of such columns, which are supposed to be good because they were priced with $\pi$ values close to our sequence of guesses $\bar{x}$. Those four steps are done in all nodes along the BCP. On the non-root nodes, we use the optimal dual values of the father node to set $\bar{\pi}$ on the first step.

## 3 Computational Results

The OR-Library contains a set of benchmark instances for the GAP that are widely used in the literature. This set is divided into 5 series: A, B, C, D and E. Series A and B are too easy for todays standards and can be solved by commercial MIP solvers using formulation F1. So we present results on the remaining series. The hardest instances are those from series D, 5 out of 6 such instances were open before this work. The experiments here described were performed in a Pentium 2GHz with 512Mb of RAM.

The first table, below, presents the gains obtained by the stabilization described in the previous section on the root node of instances from series D. It can be seen that even on instances where convergence is not a problem, like D_20_100, that stabilization is not harmful. However, on instances like D_5_200, where traditional column generation performs poorly, stabilization leads to remarkable gains. Similar gains were observed not only in the root node, but all along the BCP.

The next table presents the results of the complete BCP on all instances from series C, D and E. Column *best known* shows the previously best known solutions, values in bold indicate proven optima. Columns *time to best* and *node best* give the time and number of solved nodes at the moment that the

| series | m | n | root LB | best known | time before stabilization | time after stabilization |
|---|---|---|---|---|---|---|
| D | 5 | 100 | 6350 | 6353 | 18.71 | 3.59 |
| D | 5 | 200 | 12741 | 12743 | 6610.31 | 138.03 |
| D | 10 | 100 | 6342 | 6349 | 1.98 | 0.78 |
| D | 10 | 200 | 12426 | 12433 | 181.25 | 19.23 |
| D | 20 | 100 | 6177 | 6196 | 0.79 | 0.53 |
| D | 20 | 200 | 12230 | 12244 | 26.12 | 5.01 |

Table 1: Results of stabilization in the root node.

BCP found the optimal solution. Two instances, D_10_200 and D_20_200, could not be solved in two days of cpu time. Three instances from series D were solved for the first time, the optimal solutions found improved upon the previous best known heuristic solutions. The solution of value 6185 for instance D_20_100 was not found by the BCP itself, but by the heuristic technique described in the next section. In that case, *total time* and *total nodes* are statistics on proving the optimality of that solution.

| series | m | n | root LB | best known | this work | time to best | total time | node best | total nodes |
|---|---|---|---|---|---|---|---|---|---|
| C | 5 | 100 | 1930 | **1931** | **1931** | 0.93 | 1.17 | 3 | 5 |
| C | 5 | 200 | 3455 | **3456** | **3456** | 420.38 | 422.98 | 46 | 47 |
| C | 10 | 100 | 1400 | **1402** | **1402** | 1.12 | 1.84 | 9 | 17 |
| C | 10 | 200 | 2804 | **2806** | **2806** | 224.78 | 266.17 | 22 | 35 |
| C | 20 | 100 | 1242 | **1243** | **1243** | 1.92 | 2.12 | 25 | 29 |
| C | 20 | 200 | 2391 | **2391** | **2391** | 53.81 | 55.38 | 22 | 23 |
| D | 5 | 100 | 6350 | **6353** | **6353** | 61.74 | 96.30 | 106 | 171 |
| D | 5 | 200 | 12741 | 12743 | **12742** | 312.68 | 583.68 | 11 | 57 |
| D | 10 | 100 | 6342 | 6349 | **6347** | 705.76 | 818.62 | 2416 | 2687 |
| D | 10 | 200 | 12426 | 12433 | | | | | |
| D | 20 | 100 | 6177 | 6196 | **6185** | * | 1043.92 | * | 2807 |
| D | 20 | 200 | 12230 | 12244 | | | | | |
| E | 5 | 100 | 12673 | **12681** | **12681** | 27.68 | 30.09 | 57 | 63 |
| E | 5 | 200 | 24927 | **24930** | **24930** | 1080.56 | 1305.62 | 27 | 31 |
| E | 10 | 100 | 11568 | **11577** | **11577** | 12.53 | 22.85 | 47 | 87 |
| E | 10 | 200 | 23302 | **23307** | **23307** | 135.62 | 670.62 | 19 | 155 |
| E | 20 | 100 | 8431 | **8436** | **8436** | 6.60 | 6.81 | 36 | 37 |
| E | 20 | 200 | 22377 | **22379** | **22379** | 36.79 | 40.74 | 18 | 21 |

Table 2: Complete Stabilized BCP Results.

# 4 Ellipsoidal Cuts

Suppose we have a 0-1 IP on variables $x_j$, $j \in N$. Assume for simplicity that all feasible solutions have exactly $n$ variables with value 1, as in the case of formulation F1 for the GAP. Let $\bar{x}$ be an heuristic solution and let $N_1(\bar{x})$ be the set of variables in that solution with value 1. A classical local search to improve this solution is $k$-opt, try all solutions obtained from $\bar{x}$ by changing at most $k$ variables from $N_1(\bar{x})$. The complexity of performing $k$-opt by complete enumeration grows exponentially with $k$, therefore it is only practical for very small values of $k$ (say, up to 5).

Fischetti and Lodi [4] proposed a way to perform $k$-opt for larger values of $k$ by using a generic MIP solver to perform the search. This is done by simply adding to the IP the following constraint:

$$\sum_{j \in N_1(\bar{x})} x_j \geq n - k$$

The idea is that even if the MIP solver is not able to solve the original problem, it may be capable of solving the much more restricted problem obtained after this cut is added. Of course, if one finds a better solution within this $k$-opt, this new incumbent solution can be used as base for another cut of the same kind. And after one is sure that no better solution exists in that neighborhood, one can add the reverse cut: $\sum_{j \in N_1(\bar{x})} x_j \leq n - k - 1$. In fact, Fischetti and Lodi have proposed this scheme as a branching strategy designed to find better solutions quickly.

In an attempt to improve the best known solutions for the open instances of series D, we have used our BCP code to perform $k$-opt for large values of $k$, starting from several different heuristically constructed solutions (from rounding LP fractional solutions). On instance D_20_100, even performing 16-opt on dozens of base solutions, we could never improve the best known solution from the literature with value 6196. The root LB for those instance is 6177, our BCP had no hope of closing a gap of 19 units.

Then we changed the strategy. Instead of searching a ball neighborhood centered on a single base solution, we selected pairs of good solutions as different to each other as possible. Let $\bar{x}_1$ and $\bar{x}_2$ be such a pair of solutions with $N_1(\bar{x}_1)$ and $N_1(\bar{x}_2)$ as the sets of variables with value 1. We introduced the following *ellipsoidal cuts*:

$$\sum_{j \in N_1(\bar{x}_1) \cap N_1(\bar{x}_2)} 2x_j + \sum_{j \in (N_1(\bar{x}_1) \cup N_1(\bar{x}_2)) \setminus (N_1(\bar{x}_1) \cap N_1(\bar{x}_2))} x_j \geq \alpha(\bar{x}_1, \bar{x}_2) - k$$

where $\alpha(\bar{x}_1, \bar{x}_2)$ is the value obtained by making $x$ equal to $\bar{x}_1$ or $\bar{x}_2$ on the left side of the cut and $k$ is the amount of slack to define the ellipsoidal neighborhood. Those cuts produced much better results and we found several improving solutions including the optimal one of value 6185. Using

this value as upper bound the exact BCP had little trouble to close this instance. There is a clear parallel between ellipsoidal cuts and the path relinking strategy, used for instance in [12], both try to obtain better solutions by exploring the region between two given base solutions. We believe that MIP based local search schemes may have a lot to borrow from the metaheuristics experience.

# References


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

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

[3] du Merle, O. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999.

[4] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–47, 2003.

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

[6] S. Martello, D. Pisinger, and P. Toth. New trends in exact algorithms for the 0-1 knapsack problem. *European Journal of Operational Research*, 123:325–332, 2000.

[7] J. Nowakovski, W. Schwrzler, and E. Triesch. Using the generalized assignment problem in scheduling the ROSAT space telescope. *Eur. Journal of Oper. Research*, 112:531–541, 1999.

[8] A. Pigatti. Modelos e algoritmos para o problema de alocação generalizada e aplicações. Master's thesis, Pontifícia Universidade Católica do Rio de Janeiro, Brazil, July 2003.

[9] M. Poggi de Aragão and E. Uchoa. Integer program reformulation for robust branch-and-cut-and-price. In *Annals of Mathematical Programming in Rio*, pages 56–61, Brazil, 2003.

[10] K.S. Ruland. A model for aeromedical routing and scheduling. *International Transactions in Operational Research*, 6:57–73, 1999.

[11] M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 45:831–841, 1997.

[12] M. Yagiura and T. Ibaraki. A path relinking approach for the generalized assignment problem. Proc. International Symposium on Scheduling, Hamamatsu, Japan, 2002.

[13] M. Yagiura, T. Ibaraki, and F. Glover. An ejection chain approach for the generalized assignment problem. Technical Report 99013, Departament of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 1999.