Um algoritmo *branch-and-price* para o problema generalizado de atribuição

Edson Luiz França Senne (UNESP/FEG) elfsenne@feg.unesp.br
Luiz Antonio Nogueira Lorena (INPE/LAC) lorena@lac.inpe.br

Resumo

Este trabalho apresenta um algoritmo exato para o Problema Generalizado de Atribuição (PGA). O PGA é um dos mais representativos problemas de Otimização Combinatória e consiste em otimizar a atribuição de n tarefas a m agentes, de forma que cada tarefa seja atribuída a exatamente um agente e a capacidade de cada agente seja respeitada. O problema é conhecido ser NP-difícil. O trabalho propõe um método branch-and-price que se baseia em uma abordagem de geração de colunas combinada com a relaxação lagrangeana/surrogate. São apresentados testes computacionais que demonstram a efetividade do algoritmo proposto.
Palavras-chave: Otimização Combinatória, Problema Generalizado de Atribuição, Relaxação Lagrangeana/Surrogate, Geração de Colunas, Branch-and-Price.

1. Introdução

O Problema Generalizado de Atribuição (PGA) pode ser visto como o problema de atribuir n tarefas a m agentes ao menor custo possível, de modo que cada tarefa seja atribuída a apenas um agente e cada agente tenha sua capacidade respeitada. Sejam:

- c_{ij} o custo associado de atribuir a tarefa j ao agente i;
- r_{ij} a quantidade de recursos requeridos do agente i pela tarefa j;
- b_i a capacidade (recursos disponíveis) do agente i;
- x_{ij} uma variável binária tal que $x_{ij} = 1$, se a tarefa j está atribuída ao agente i, e $x_{ij} = 0$, caso contrário.

Com isto, o PGA pode ser formulado como:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \qquad (1)$$

sujeito a:

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j \in N = \{1, \ldots, n\} \qquad (2)$$

$$\sum_{j=1}^{n} r_{ij} x_{ij} \leq b_i \qquad i \in M = \{1, \ldots, m\} \qquad (3)$$

$$x_{ij} \in \{0,1\} \qquad i \in M; j \in N \qquad (4)$$

A função-objetivo (1) corresponde ao custo total de atribuição. A restrição (2) garante que a tarefa j será atribuída a apenas um agente, a restrição (3) garante que a capacidade do agente i será respeitada e a restrição (4) corresponde às condições de integralidade.

Muitos problemas têm sido apresentados na literatura como aplicações do PGA, tais como: distribuição de tarefas entre processadores de um sistema de computação paralela (BOKHARI, 1987), alocação de salas de aula (LUAN e YAO, 1996), planejamento de tarefas de telescópio espacial (NOWAKOVSKI *et al*., 1999), alocação de pacientes em vôos médicos (RULAND, 1999), carregamento de caminhões (PIGATTI, 2003), recuperação de blocos de dados a partir de discos paralelos (AERTS *et al*., 2003), dentre muitos outros.

Uma revisão geral das técnicas de solução propostas para o PGA pode ser vista em Cattrysse e Van Wassenhove (1992). Com relação aos métodos aproximados, as abordagens, em geral, propõem heurísticas em que, numa primeira fase, as atribuições são calculadas de modo a satisfazer um determinado critério, por exemplo, minimizar uma função de penalização, como em Martello e Toth (1981) e, numa segunda fase, as atribuições são melhoradas. Amini e Racer (1994) também propõem uma abordagem heurística em duas fases para o problema e fazem uma cuidadosa comparação de vários métodos. Metaheurísticas como algoritmos genéticos (WILSON, 1997; CHU e BEASLEY, 1997), busca tabu (LAGUNA *et al*., 1995) e *simulated annealing* (OSMAN, 1995) também já foram propostas para o PGA. Narciso e Lorena (1999) aplicaram a relaxação lagrangeana/*surrogate* combinada com otimização por subgradientes ao PGA. Outras linhas de investigação para o problema são propostas por Alfandari *et al*. (2001) e Yagiura *et al*. (2004a e 2004b). Senne *et al*. (2004) propõem um método de geração de colunas utilizando a relaxação lagrangeana/*surrogate.*

Os primeiros métodos exatos desenvolvidos para o PGA baseavam-se no algoritmo *branch-and-bound* (B&B), combinado com a relaxação lagrangeana e com heurísticas para a obtenção de limites (FISHER *et al*., 1986). Métodos baseados em heurísticas duais e em relaxações de Programação Linear (PL) para a obtenção de limitantes superiores e inferiores para o problema primal também já foram propostos (GUIGNARD e ROSENWEIN, 1989). Abordagens mais recentes propõem a combinação do algoritmo B&B com métodos de geração de colunas, o que passou a ser conhecido como algoritmo *branch-and-price* (B&P). Savelsbergh (1997) propõe um algoritmo B&P para resolver uma reformulação do PGA como um problema de particionamento de conjuntos. Barnhart *et al.* (1998) propõem o uso de algoritmos B&P para solucionar problemas inteiros de grande porte e fazem uma revisão de alguns trabalhos similares já propostos na literatura. Nauss (2003) propõe a utilização de cortes, relaxação lagrangeana e otimização por subgradientes. Pigatti *et al*. (2004) propõem a utilização de cortes combinados com um algoritmo B&P para obter soluções eficientes para o PGA.

Neste trabalho apresenta-se um algoritmo B&P, que se baseia no método de geração de colunas proposto por Senne *et al.* (2004), para a solução exata do PGA.

2. O algoritmo proposto

O método de geração de colunas proposto por Senne *et al*. (2004) baseia-se na aplicação da decomposição de Dantzig e Wolfe (1960) de modo a reformular o PGA como um problema de cobertura de conjuntos. Para isto, considere $K_i = \{1, \cdots, k_i\}$ e $x_k^i = (x_{1k}^i, ..., x_{nk}^i)$, para $i \in M$ e $k \in K_i$, como um padrão de atribuição de tarefas ao agente i. Cada padrão x_k^i é uma solução factível para (3)-(4) e corresponde a um vetor (coluna) de n elementos binários representando se a tarefa j ($j \in N$) está ou não atribuída ao agente i e de um vetor unitário e_i, de m elementos, que identifica o agente.

Seja y_k^i para uma variável binária tal que $y_k^i = 1$ se o padrão x_k^i está sendo usado e $y_k^i = 0$, caso contrário. Com isto, o PGA pode ser reformulado como:

$$\min \sum_{i=1}^{m} \sum_{k=1}^{k_i} \left(\sum_{j=1}^{n} c_{ij} x_{jk}^{i} \right) y_k^i \quad (5)$$

sujeito a:

$$\sum_{i=1}^{m} \sum_{k=1}^{k_i} x_{jk}^{i} y_k^i = 1 \quad \forall\, j \in N \quad (6)$$

$$\sum_{k=1}^{k_i} y_k^i \leq 1 \quad \forall\, i \in M \quad (7)$$

$$y_k^i \in \{0, 1\}, i \in M; k \in K_i \quad (8)$$

Esta formulação foi usada por Cattrysse *et al*. (1994) para desenvolver um algoritmo heurístico para o PGA. Para o método de geração de colunas, a relaxação de PL deste problema, obtida substituindo-se a restrição (8) por $y_k^i \in [0,1]$, é conhecida como problema mestre restrito (PMR) (BARNHART *et al.*, 1998). Observe que nesta formulação, para determinar cada um dos padrões de atribuição x_k^i é necessário resolver um problema da mochila.

Após resolver o PMR, a busca por novas colunas pode ser realizada resolvendo-se o problema de custo reduzido dado por:

$$\max_{i \in M} (z_{M_i} - \mu_i)$$

onde μ_i é o custo dual ótimo associado à restrição (7) de viabilidade dos agentes no problema mestre restrito e z_{M_i} é o valor da solução ótima do seguinte problema da mochila:

$$\min \sum_{j=1}^{n} (c_{ij} - t\, \lambda_j)\, x_j^i \quad (9)$$

sujeito a:

$$\sum_{j=1}^{n} r_{ij} x_j^i \leq b_i \quad (10)$$

$$x_j^i \in \{0,1\}, j \in N \quad (11)$$

onde λ_j é o custo dual ótimo associado às restrições (6) das tarefas j no problema mestre restrito, e t é o multiplicador lagrangeano/*surrogate* (LORENA *et al.*, 2003) Assim, determinado um conjunto inicial de colunas para o problema (5)-(8), o método de geração de colunas para o PGA pode ser estabelecido pelo seguinte algoritmo:

1) Resolva o PMR considerando o conjunto atual de colunas, obtendo as variáveis duais λ_j ($j \in N$), referentes às restrições (6) e μ_i ($i \in M$), referentes às restrições (7);
2) Utilize as variáveis duais λ_j ($j \in N$) para determinar o multiplicador lagrangeano/*surrogate* t;
3) Utilize as variáveis duais λ_j ($j \in N$) e o valor do multiplicador t encontrado no passo anterior e resolva os subproblemas (9)-(11) para cada $i \in M$. Sejam x_k as colunas

referentes às soluções destes subproblemas;

4) Adicione as colunas x_k com custos reduzidos negativos ao PMR;

5) Se, no passo anterior, nenhuma nova coluna foi acrescentada ao PMR, então pare. Caso contrário, volte para o passo 1.

A menos que todas as variáveis y_k^i sejam inteiras, a solução obtida pelo método de geração de colunas não é factível para o PGA. Assim, faz-se necessário empregar um algoritmo B&B para determinar soluções factíveis. Isto implica em separar o problema em subproblemas, o que se traduz em uma estratégia de ramificação para a árvore de busca. A regra de ramificação deve ser compatível com o subproblema, de modo a evitar a geração de colunas que já foram retiradas pelas ramificações. No método B&P proposto neste trabalho adotou-se a estratégia de ramificação sugerida por Pigatti (2003):

– no ramo da esquerda, para proibir que uma tarefa j seja alocada ao agente i, todas as colunas associadas ao agente i tais que $x_{jk}^i = 1$ devem ser fixadas em zero, ou seja, $y_k^i = 0$, para todo $k \in K_i$;

– no ramo da direita, para assegurar que uma tarefa j seja alocada ao agente i, todas as colunas associadas ao agente i tais que $x_{jk}^i = 0$ são fixadas em zero, ou seja, $y_k^i = 0$ para todo $k \in K_i$, e todas as colunas associadas a um agente $i' \neq i$ tais que $x_{jk}^{i'} = 1$, são fixadas em zero, ou seja, $y_k^{i'} = 0$ para todo $k \in K_{i'}$.

Com isto, o método B&P proposto para o PGA pode ser estabelecido pelo seguinte algoritmo:

Seja lista = { problema mestre inicial };

Enquanto (lista $\neq \varnothing$), repetir:

Recuperar um problema p da lista; Fazer lista = lista – { p };

Estabelecer as colunas úteis para o problema p;

z = Resolver o problema p; Seja L_{inf} o limite inferior para o problema p, obtido resolvendo-se o problema dual lagrangeano/*surrogate*;

Se (z < msv), então:

Resolver o PMR final como um problema inteiro;

Se (L_{inf} < msv – 1)

Sejam p_E o problema resultante de p quando se considera a ramificação da árvore pela esquerda e p_D o problema resultante de p quando se considera a ramificação da árvore pela direita.

Fazer lista = lista $\cup$ { p_E, p_D };

Senão:

Podar a árvore no nó correspondente ao problema p.

Neste algoritmo, msv corresponde ao valor da melhor solução viável encontrada. Inicialmente, msv é igual ao valor da primeira solução inteira obtida pelo CPLEX para o problema (5)-(8). As colunas referentes a esta primeira solução inteira compõem o problema mestre restrito inicial, juntamente com um conjunto de n colunas artificiais. Cada coluna

artificial corresponde a um vetor nulo, exceto por um elemento correspondente à uma tarefa j. O custo de cada coluna artificial é estabelecido como o custo da atribuição (agente, tarefa) mais cara.

Antes de resolver um problema é preciso verificar se as colunas do PMR se aplicam ou não ao problema a ser resolvido, tendo em vista as fixações de variáveis feitas pelas ramificações da árvore. Para o nó raiz da árvore, todas as colunas do PMR são consideradas como colunas úteis. Para os demais nós da árvore, uma coluna que não se aplica ao problema é "excluída" do PMR atribuindo-se o valor zero ao limite superior da variável y_k^i correspondente a esta coluna.

Para resolver um problema emprega-se o algoritmo de geração de colunas estabelecido anteriormente. A cada iteração deste algoritmo utiliza-se um valor adequado para o multiplicador lagrangeano/*surrogate* t. O melhor valor de t, obtido pelo algoritmo de busca proposto por Senne e Lorena (2000), é utilizado apenas na primeira iteração. Nas demais iterações, o valor do parâmetro t cresce a um passo conveniente até atingir o valor 1. A partir daí, o multiplicador t é mantido inalterado nas demais iterações.

Ao término do algoritmo de geração de colunas, considera-se o PMR final como um problema inteiro (ou seja, $y_k^i \in \{0,1\}$), e utiliza-se o software CPLEX para determinar uma solução viável para o PGA. Por razões práticas, considera-se apenas a primeira solução inteira encontrada pelo CPLEX. Esta solução pode não ser válida para o problema original, pois pode incluir alguma coluna artificial. De qualquer forma, a solução obtida é passada para um procedimento heurístico que, por meio de trocas, tenta viabilizar e melhorar o valor da solução viável atual.

Para determinar a ramificação de um problema escolhe-se a variável y_k^i que tiver o menor valor de $\left(\left(1-\left|y_k^i-0.5\right|\right)*100+c_{ij}\right)$. Uma vez escolhida a coluna, ou seja, o índice do agente i, determina-se o índice da tarefa j para o qual o valor de x_{ij} deverá ser fixado. Optou-se por escolher o índice j da primeira tarefa desta coluna que não viole as fixações de variáveis já estabelecidas anteriormente na árvore. A ramificação da árvore pela esquerda corresponde à fixação $x_{ij} = 1$ e a ramificação pela direita, à fixação $x_{ij} = 0$.

3. Resultados computacionais

O algoritmo de B&P proposto foi implementado na linguagem C e executado em um microcomputador Pentium 4 de 3 GHz e 512 MB de RAM, utilizando a versão 7.5 do software CPLEX.

Implementou-se um procedimento de remoção de colunas que é aplicado quando o PMR excede 40000 colunas. Neste caso, removem-se as 10000 piores colunas (com maiores custos reduzidos). Para os experimentos computacionais foram consideradas as instâncias das classes C, D e E da OR-Library (Beasley, 2004). Os resultados estão mostrados na Tabela 1. Nesta tabela são mostrados os resultados obtidos pelo algoritmo proposto (que utiliza a relaxação lagrangeana/*surrogate*) e, entre colchetes, os resultados obtidos pelo mesmo algoritmo, mas fixando-se t = 1, o que corresponde a utilizar a relaxação lagrangeana tradicional. Na Tabela 1, **m** é o número de agentes, **n** é o número de tarefas, **SÓtima** é a solução ótima para o problema, **SEnc** é a solução encontrada pelo algoritmo proposto, **NEnc** é o nó da árvore em que a solução SEnc foi encontrada, **TEnc** é o tempo para encontrar a solução SEnc (em segundos), **TamArv** é o total de nós visitados na árvore e **TÓtimo** é o tempo total para provar

a otimalidade da solução (em segundos). O símbolo –, nesta tabela, significa que o algoritmo excedeu o limite máximo de tempo de execução, estabelecido em 18000 segundos.

Classe	m	n	SÓtima	SEnc	NEnc	TEnc	TamArv	TÓtimo
C	5	100	1931	1931 [1931]	1 [1]	1,39 [1,62]	25 [23]	32,13 [24,75]
	10	100	1402	1402 [1402]	27 [47]	10,58 [17,35]	49 [67]	16,19 [22,44]
	20	100	1243	1243 [1243]	17 [31]	3,14 [4,73]	17 [61]	3,14 [8,78]
	5	200	3456	3456 [3456]	34 [7]	2836,03 [569,72]	65 [97]	4850,24 [5237,39]
	10	200	2806	2806 [2806]	227 [738]	791,72 [2641,03]	343 [793]	1241,07 [3116,83]
	20	200	2391	2391 [2391]	26 [16]	40,76 [28,42]	26 [19]	40,76 [30,53]
D	5	100	6353	6353 [6353]	568 [651]	2783,60 [3341,29]	583 [667]	3142,27 [3407,99]
	10	100	6347	6347 [6347]	2395 [2707]	1946,71 [2261,44]	2715 [2899]	2282,87 [2441,65]
	20	100	6185	6188 [6190]	7782 [11793]	3186,21 [4867,94]	31600 [33471]	– [–]
E	5	100	12681	12681 [12681]	32 [90]	58,05 [178,06]	101 [179]	283,95 [501,47]
	10	100	11577	11577 [11577]	81 [170]	45,97 [87,44]	189 [251]	103,58 [142,26]
	20	100	8436	8436 [8436]	17 [127]	9,55 [50,58]	83 [131]	29,53 [52,14]
	5	200	24930	24930 [24930]	72 [139]	6386,13 [9724,08]	133 [145]	12013,82 [13424,88]
	10	200	23307	23307 [23307]	337 [362]	2324,08 [2652,20]	577 [617]	5051,86 [5219,91]
	20	200	22379	22379 [22379]	6 [9]	55,55 [111,01]	23 [31]	94,85 [161,11]
Médias					**774,80 [1125,87]**	**1365,30 [1769,13]**	**2435,27 [2630,07]**	**2084,73 [2413,72]**

Tabela 1 – Resultados obtidos

Como é possível notar pelos valores médios mostrados na Tabela 1, o algoritmo que utiliza a relaxação lagrangeana/*surrogate* é superior ao algoritmo que utiliza a relaxação lagrangeana tradicional. Pode-se notar também pelos resultados mostrados nesta tabela que quanto maior o número de tarefas, mais difícil é o problema. Mas, fixado o número de tarefas, a complexidade do problema é inversamente proporcional ao número de agentes. Isto pode ser observado para os problemas das classes C e E. Os problemas da classe D, no entanto, são muito difíceis e isto se deve ao fato do custo de atribuição das tarefas aos agentes ser inversamente proporcional aos recursos requeridos por estas atribuições. Os problemas da classe D para 200 tarefas não foram resolvidos, pois exigiriam um tempo de execução maior do que o limite estabelecido.

4. Conclusão

O algoritmo proposto pode ser comparado com o de Nauss (2003), que utiliza cortes e otimização por subgradientes. A Tabela 2 compara os resultados obtidos pelo algoritmo proposto neste trabalho (**B&P**) com os resultados obtidos por Nauss (**B&B**). O algoritmo B&B foi executado em um computador Pentium 2 de 300MHz.

				B&P			B&B		
Classe	**m**	**n**	**SÓtimo**	**SEnc**	**TEnc**	**TÓtimo**	**SEnc**	**TEnc**	**TÓtimo**
C	5	100	1931	1931	1,4	32,1	1931	0,3	7,1
	10	100	1402	1402	10,6	16,2	1402	15,5	33,2
	20	100	1243	1243	3,1	3,1	1243	39,7	69,5
	5	200	3456	3456	2836,0	4850,2	3456	36,4	45,8
	10	200	2806	2806	791,7	1241,1	2806	631,1	1081,0
	20	200	2391	2391	40,8	40,8	2392	926,0	–
D	5	100	6353	6353	2783,6	3142,3	6353	170,7	174,7
	10	100	6347	6347	1946,7	2282,9	6349	1023,7	–
	20	100	6185	6188	3186,2	–	6196	2122,9	–
E	5	100	12681	12681	58,1	284,0	12681	44,8	46,1
	10	100	11577	11577	46,0	103,6	11577	58,3	125,4
	20	100	8436	8436	9,6	29,5	8436	1660,5	1770,1
	5	200	24930	24930	6386,1	12013,8	24930	61,5	69,5
	10	200	23307	23307	2324,1	5051,9	23307	357,6	990,2
	20	200	22379	22379	55,6	94,9	22379	988,2	1126,0

Tabela 2 – Comparação com a abordagem de Nauss (2003)

A Tabela 2 mostra que, em média, o algoritmo proposto por Nauss (2003) consegue obter e confirmar a otimalidade das soluções mais rapidamente do que o algoritmo proposto. No entanto, o algoritmo proposto foi capaz de encontrar a solução ótima para 14 dos 15 problemas considerados, ao passo que o algoritmo B&B obteve a solução ótima apenas para 12 destes problemas, dentro do limite de tempo estabelecido (3600 segundos). Resultados ainda melhores são obtidos pelo algoritmo *branch-and-cut-and-price* de Pigatti *et al*. (2004), o que sugere que a utilização de cortes é essencial para a solução eficiente do PGA.

Referências

AERTS, J.; KORST, J. & SPIEKSMA, F. Approximation of a retrieval problem for parallel disks. *Lecture Notes in Computer Science*, v. 2653, p. 178-188, 2003.

ALFANDARI, L.; PLATEAU, A. & TOLLA , P. *A Two-Phase Path-Relinking Algorithm for the Generalized Assignment Problem.* In: MIC2001 - 4th Metaheuristics International Conference, Porto, Portugal, July 2001.

AMINI, M.M. & RACER, M. A rigorous comparison of alternative solution methods for the generalized assignment problem. *Management Science*, v. 40, p. 868-890, 1994.

BOKHARI, S.H. *Assignment Problems in Parallel and Distributed Computing*. Kluwer Academic Publisher, Boston, USA, 1987.

BARNHART , C.; JOHNSON, E.L.; NEMHAUSER, G.L.; SAVELSBERGH , M.W.P. & VANCE, P.H. Branch-and-price: column generation for solving huge integer programs. *Operations Research*, v. 46, n. 3, p. 316-329, 1998.

BEASLEY, J.E. *OR-Library*. Disponível em: http://www.brunel.ac.uk/depts/ma/research/ jeb/orlib/gapinfo.html. Acesso em 23 de dezembro de 2004.

CATTRYSSE, D.; SALOMON, M. & VAN WASSENHOVE, L. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research*, v. 72 p. 167-174, 1994.

CATTRYSSE, D.G. & VAN WASSENHOVE, L.N. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, v. 60, p. 260-272, 1992.

CHU, P.C. & BEASLEY, J.E. A genetic algorithm for the generalised assignment problem. Computers and *Operations Research*, v. 24, p. 17-23, 1997.

DANTZIG. G.B. & WOLFE, P. Decomposition principle for linear programs. *Operations Research*, v. 8, p. 101-111, 1960.

FISHER, M.L.; JAIKUMAR, R. & VAN WASSENHOVE, L.N. A multiplier adjustment method for the generalized assignment problem. *Management Science*, v. 32, n. 9, p. 1095-1103, 1986.

GUIGNARD, M. & ROSENWEIN, M. An improved dual-based algorithm for the generalized assignment problem. *Operations Research*, v. 37, n. 4, p. 658-663, 1989.

LAGUNA, M.; KELLY, J.P.; GONZALEZ-VELARDE, J.L. & GLOVER, F. Tabu search for the multilevel generalized assignment problem. *European Journal of Operations Research*, v. 42, p. 677–687, 1995.

LORENA, L.A.N., PEREIRA, M.A. & SALOMÃO, S.N.A. A relaxação lagrangeana/surrogate e o método de geração de colunas: novos limitantes e novas colunas. *Revista Pesquisa Operacional*, v. 23, n. 1, p. 29-47, 2003. Edição especial: 60 anos – Professor Nelson Maculan.

LUAN, F. & YAO, X. *Solving real-world lecture room assignment problems by genetic algorithms*, Complex Systems - From Local Interactions to Global Phenomena, IOS Press, Amsterdam, p.148-160, 1996.

MARTELLO, S. & TOTH, P. An algorithm for the generalized assignment problem. *Operational Research*, v. 81, p. 589-603, 1981.

NARCISO, M.G. & LORENA, L.A.N. Lagrangean/surrogate relaxation for generalized assignment problems. *European Journal of Operational Research*, v. 114, p. 165-177, 1999.

NAUSS, R.M. Solving the generalized assignment problem: An optimizing and heuristic approach. *INFORMS Journal on Computing*, v. 15, n. 3, p. 249-266, 2003.

NOWAKOVSKI, J.; SCHWÄRZLER, W. & TRIESCH, E. Using the generalized assignment problem in scheduling the ROSAT space telescope. *European Journal of Operational Research*, v. 112, n. 3, p. 531-541, 1999.

OSMAN, I.H. Heuristics for the Generalized Assignment Problem: Simulated Annealing and Tabu Search Approaches, *OR Spektrum*, v. 17, p. 211-225, 1995.

PIGATTI, A.A. *Modelos e algoritmos para o problema de alocação generalizada e aplicações*. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, Julho 2003.

PIGATTI, A.; ARAGÃO, M.; UCHOA, E. *Stabilized branch-and-cut-and-price for the generalized assignment problem.* Disponível em: http://www.optimization-online.org/db_html/2004/11/990.html. Acesso em 23 de dezembro de 2004.

RULAND, K.S. A model for aeromedical routing and scheduling. *International Transactions in Operational Research*, v. 6, n. 1, p. 57-73, 1999.

SAVELSBERGH, M. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, v. 45, n. 6, p. 831-841, 1997.

SENNE, E.L.F. & LORENA, L.A.N. *Lagrangean/surrogate heuristics for p-median problems*. In: Computing tools for modeling, optimization and simulation: interfaces in computer science and operations research. M. Laguna and J.L. Gonzalez-Velarde (eds.). Kluwer Academic Publishers, 2000, p. 115-130.

SENNE, E.L.F.; LORENA, L.A.N. & SALOMÃO, S.N.A. Uma abordagem de geração de colunas para o problema generalizado de atribuição. *Revista Produção Online*, v. 4, n. 4, p. 2950-2957, 2004.

WILSON, J.M. A genetic algorithm for the generalised assignment problem. *Journal of the Operational Research Society*, v. 48, n. 8, p. 804-809, 1997.

YAGIURA, M.; IBARAKI, T. & GLOVER, F. *A path relinking approach with ejection chains for the generalized assignment problem.* Technical Report 2004-002, Department of Applied Mathematics and Physics, Kyoto University, 2004a.

YAGIURA, M.; IBARAKI, T. & GLOVER, F. An ejection chain approach for the generalized assignment problem. *INFORMS Journal on Computing*, v. 16, n. 2, p. 133-151, Spring 2004b.