Heurísticas GRASP para o problema de alocação dinâmica de espaços

Geiza Cristina da Silva,Paulo Oswaldo Boaventura Netto

COPPE-PRODUÇÃO - Universidade Federal do Rio de Janeiro (UFRJ)
e-mail: aziegc@yahoo.com.br; boaventu@pep.ufrj.br

Luiz Satoru Ochi

Instituto de Computação – Universidade Federal Fluminense (IC-UFF)
e mail: satoru@ic.uff.br

Resumo

O Problema de Alocação Dinâmica de Espaços (PADE) é relativo novo na literatura e foi inspirado na necessidade de otimização da distância percorrida por recursos requeridos para a realização de atividades em projetos. Um projeto é dividido por um número de períodos consecutivos e, em cada um deles, uma quantidade de atividades é realizada. Os recursos necessários para as atividades devem ser associados a espaços de trabalho e, os recursos ociosos no período devem ser guardados em depósitos. O objetivo do problema é minimizar a distância total percorrida pelos recursos entre locações. Neste trabalho são propostos métodos heurísticos de construção e busca local que, combinados, são usados como base em diferentes versões do algoritmo GRASP (*Greedy Randomized Adaptive Search Procedure*). Resultados computacionais mostram que os algoritmos propostos sempre alcançam uma solução ótima quando esta é conhecida e, para instâncias maiores, apresentam um desempenho médio superior quando comparados com outras heurísticas da literatura.

1. Introdução

O problema de alocação dinâmica de espaços (*Dynamic space allocation problem*) foi recentemente proposto em McKendall et al. (2006). Este problema foi inspirado na necessidade de otimização da distância percorrida por recursos requeridos para a realização de atividades em projetos. Um projeto é dividido por um número de períodos consecutivos e, em cada um deles, uma quantidade de atividades é realizada. Os recursos necessários para as atividades devem estar disponíveis em espaços de trabalho e, os recursos ociosos no período devem ser guardados em depósitos.

Este problema é considerado difícil para se resolver computacionalmente (McKendall e Jaramillo, 2006) e, portanto, a dificuldade em encontrar uma solução ótima para instâncias de elevadas dimensões justifica o uso de métodos aproximados. Os métodos *simulated annealing* e busca tabu foram aplicados por McKendall et al. (2006) e McKendall e Jaramillo (2006). Neste trabalho, propomos heurísticas GRASP (Feo e Resende, 1995; Rezende e Ribeiro, 2003) e comparamos os resultados obtidos com os métodos da literatura.

A seção 2 descreve o problema e faz uma breve revisão bibliográfica. A seção 3 apresenta os algoritmos da literatura e a seção 4, os algoritmos aqui propostos. Na seção 5, são descritos os

resultados computacionais e a análise dos mesmos e a seção 6 faz conclusões e sugestões para trabalhos futuros.

2. O problema de alocação dinâmica de espaços (PADE)

Considere um projeto dividido em *períodos* e, em cada um deles, existe um conjunto de *atividades* a ser realizado. Cada atividade requer um conjunto de *recursos*. Os recursos que não utilizados em um período por qualquer atividade são considerados *recursos ociosos*. O problema consiste em associar recursos a *locações* próprias, isto é, recursos requeridos por atividades em *espaços de trabalho* e recursos ociosos em *depósitos* de forma que a distância total percorrida pelos recursos entre as locações seja mínima.

Uma solução para o PADE, portanto, constitui-se de duas matrizes, uma matriz de atividades que relaciona períodos, espaços de trabalho e atividades e uma matriz de recursos ociosos, descrita por períodos e recursos ociosos.

A fim de exemplificar o problema, as Figuras 1 a 5 apresentam uma instância do problema. A Figura 1 representa um conjunto com 6 locações distribuídas em 3 espaços de trabalho e 3 depósitos.

Locaçao 1 (Espaço de trabalho 1)	**Locaçao 2** (Espaço de trabalho 2)	**Locaçao 3** (Espaço de trabalho 3)
Locaçao 4 Depósito 1	**Locaçao 5** Depósito 2	**Locaçao 6** Depósito 3

Figura 1 – Exemplo de locações

A Figura 2 representa o planejamento de um projeto 5 períodos onde 5 atividades devem ser realizadas.

Período	Atividades
1	1, 2
2	2, 3
3	4
4	4, 5
5	5

Figura 2 – Planejamento de atividades por período

A Figura 3a mostra os recursos necessários para a realização de cada atividade do projeto e a Figura 3b apresenta os recursos ociosos por período. Vemos (Figura 2) que, no período 1 deverão ser realizadas as atividades 1 e 2. Estas, por sua vez, demandam os recursos 1, 3 e 7 e 2 e 4, respectivamente. Desta forma, os demais recursos (5, 6, 8 e 9) são ociosos neste período.

Atividade	Recursos
1	1, 3, 7
2	2, 4
3	5, 6, 8
4	1, 2, 8
5	3, 6, 9

Período	Recursos Ociosos
1	5, 6, 8, 9
2	1, 3, 7, 9
3	3, 4, 5, 6, 7, 9
4	4, 5, 7
5	1, 2, 4, 5, 7, 8

Figura 3a) Recursos necessários para cada atividade e 3b) Recursos ociosos por período

Devemos considerar ainda a distância entre locações, apresentada na Figura 4.

	1	**2**	**3**	**4**	**5**	**6**
1	0	1	2	1	2	3
2	1	0	1	2	1	2
3	2	1	0	3	2	1
4	1	2	3	0	1	2
5	2	1	2	1	0	1
6	3	2	1	2	1	0

Figura 4 – Matriz de distância entre locações

Consideramos, neste exemplo, que a capacidade máxima de cada locação é de três recursos. Uma solução para esta instância é dada na Figura 5, com total de 22 unidades de distância percorrida pelos recursos. A alocação de atividades/recursos e o cálculo da distância podem ser entendidos da seguinte maneira: No período 1, as atividades 1 e 2 e seus recursos são associados ao espaço de trabalho 1 (ET1) e 2 (ET2), respectivamente. O recurso ocioso 9 foi associado ao depósito 2 (D2) e 5, 6, 8 ao depósito 3 (D3). No período 2, a atividade 2 continua em execução e, portanto, seus recursos 2 e 4 continuam alocados em ET2. A atividade 3 não é realizada neste período e seus recursos 5, 6 e 8 são retirados de D3 e associados a ES3, isto nos dá um custo de 3 unidades de distância. Os recursos ociosos 1, 3 e 7 foram retirados de ET1 e associados a D1 (custo = 3) e o recurso ocioso 9 não foi movimentado e continua no depósito D2. O custo total de movimentação de recursos no período 2 é igual a 6. No período 3, a atividade 4 foi associada a ET1, assim como seus recursos, 1, 2 e 8. Houve o transporte do recurso 2 que, no período anterior, estava alocado em ET2, com custo de 1 unidade e do recurso 8 de ET3 para ET1, com custo de 2 unidades. Além disso, o recurso 1 que estava no depósito D1 foi transportado para ET1 com custo de 1 unidade. Os recursos 4, 5 e 6 tornaram-se ociosos neste período e o custo de transporte destes é de 3 unidades. Continuando desta forma, obtemos o custo total da solução de 22 unidades de distância.

2.1.Considerações sobre problema:

As seguintes considerações sobre o problema foram propostas em McKendall e Jaramillo (2006).

a) As locações (espaços de trabalho e depósitos) e as distâncias entre elas são conhecidas;
b) Os recursos necessários para a realização de uma atividade são conhecidos a priori e são determinados respeitando a relação de precedências entre atividades;
c) Somente uma atividade pode ser realizada em cada espaço de trabalho em um determinado período;

Matriz de Atividades			
Período	**ET1**	**ET2**	**ET3**
1	**Ativ. 1** (1,3,7)	**Ativ. 2** (2,4)	
2		**Ativ. 2** (2,4)	**Ativ. 3** (5,6,8)
3	**Ativ. 4** (1,2,8)		
4	**Ativ. 4** (1,2,8)	**Ativ. 5** (3,6,9)	
5		**Ativ. 5** (3,6,9)	

Matriz de Recursos Ociosos			
Período	**D1**	**D2**	**D3**
1		9	5,6,8
2	1,3,7	9	
3	3,7	4,9	5,6
4	7	4	5
5	1,2,7	4,8	5

Figura 5 – Uma solução para o PADE

d) A capacidade do espaço de trabalho associado a uma atividade é suficiente para guardar seus respectivos recursos;
e) Cada atividade requer somente um espaço de trabalho e pelo menos um recurso;
f) Se uma atividade é realizada em múltiplos períodos, esta atividade deve ser associada ao mesmo espaço de trabalho em cada um destes períodos.
g) Se um recurso é ocioso em múltiplos períodos, este recurso deve ser associado ao mesmo depósito em cada um destes períodos;
h) As capacidades dos depósitos são conhecidas;
i) O objetivo é minimizar o custo de transporte dos recursos entre locações.

2.2. Formulação matemática

O PADE pode ser escrito como um problema de programação quadrática inteira utilizando a notação apresentada nas seções 2.2.1 a 2.2.4.

2.2.1. Índices

- J = número total de atividades (j = 1, 2, …, J);
- T = número total de recursos (t = 1, 2, …, T);
- P = número total de períodos (p = 1, 2, …, P);
- N = número total de locações (espaços de trabalho e depósitos);
- L = conjunto de locações, $|L| = N$ (L = 1, 2, …, N);
- W = conjunto de espaços de trabalho, onde $W \subset L$ ($w \in W$);
- S = conjunto de depósitos, onde $S \subset L$ ($s \in S$) e $W \cup S = L$;
- Rj = conjunto de recursos necessários para realizar a atividade j;
- Ip = conjunto de recursos ociosos no período p;
- Ap = conjunto de atividades no período p.

2.2.2. Parâmetros de entrada:

- dkl = distância entre as locações k e l;
- Cs = capacidade do depósito s;
- NRj = número de recursos necessários para realizar a atividade j.

2.2.3. Variáveis de decisão:

- $xptk$ = 1 se no período p, o recurso t está associado ao depósito k
  = 0 caso contrário;
- yjw = 1 se a atividade j é realizada no espaço de trabalho w
  = 0 caso contrário.

2.2.4. Formulação:

Minimizar:

$$\sum_{t=1}^{T}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{p=1}^{P-1} d_{kl}\, x_{ptk}\, x_{p+1tl} \qquad (1)$$

Sujeito a:

$$\sum_{\forall s \in S} x_{pts} = 1, \forall p, \forall t \in I_p, \qquad (2)$$

$$\sum_{\forall s \in S} x_{pts} \le C_s, \forall s \in S, \forall p, \qquad (3)$$

$$\sum_{\forall w \in W} y_{jw} = 1, \forall j, \quad (4)$$

$$\sum_{\forall j \in A_p} y_{jw} \leq 1, \forall w \in W, \forall p, \quad (5)$$

$$\sum_{t \in R_j} x_{ptk} = NR_j y_{jw}, \forall p, \forall j \in A_p, \forall w \in W, \quad (6)$$

$$x_{ptk} = 0 \text{ ou } 1, \forall p, \forall t, Ak \in L, \quad (7)$$

$$x_{jw} = 0 \text{ ou } 1, \forall j, \forall w \in W. \quad (8)$$

A função objetivo (1) minimiza o custo de transporte dos recursos entre espaço de trabalhos e depósitos durante os períodos onde as atividades serão realizadas. As restrições (2) garantem que todos os recursos ociosos serão alocados a um depósito em um período e as restrições (3) garantem que a capacidade dos depósitos será respeitada. As restrições (4) garantem que haverá um espaço de trabalho para cada atividade e as restriçoes (5) garantem que uma atividade será realizada em um espaço de trabalho. A equação (6) garante que os recursos necessários a realização de uma atividade serão associados ao mesmo espaço de trabalho que a atividade tiver sido associada e as restrições (7) e (8) indica que as variáveis de decisão do problema devem ser binárias.

3. Algoritmos da Literatura

Em uma breve revisão bibliográfica, encontramos somente dois trabalhos tratando do PADE. O primeiro (McKendall et al., 2006) define o problema e propõe duas heurísticas baseadas em *simulated annealing* para obter soluções aproximadas para um conjunto de instâncias gerado aleatoriamente. No segundo trabalho (McKendall e Jaramillo, 2006), os autores propõem algoritmos de construção e uma busca tabu para o problema e comparam os resultados obtidos com os da literatura, utilizando o mesmo conjunto de instâncias. Eles reportam que a busca tabu proposta supera os resultados apresentados na literatura em qualidade de solução e tempo computacional.

Apresentaremos, nesta seção, um algoritmo de construção e busca tabu descritos por McKendall e Jaramillo (2006).

3.1. Algoritmo de Construção First Assignment (FA)

Este algoritmo constrói uma solução para o PADE em duas fases: a primeira alocando atividades e a segunda, os recursos ociosos. Na primeira fase, para cada período, a primeira atividade do período é associada ao primeiro espaço de trabalho disponível, a segunda atividade ao segundo espaço disponível e assim por diante, concluindo esta fase quando o processo tiver sido feito para todos os períodos. Se uma atividade é realizada em períodos consecutivos, ela deve ser associada ao mesmo espaço de trabalho em cada um destes períodos (seção 2.1, item f). Na segunda fase do algoritmo, a idéia é a mesma, o primeiro recurso ocioso do período é alocado ao primeiro depósito disponível e assim por diante, devendo-se respeitar a regra descrita na seção 2.1, item g.

3.2. Busca Tabu

Os autores definem três tipos de movimentos, a partir de uma solução construída. O movimento 1 equivale a troca de locações (espaços de trabalho ou depósitos) de duas ou mais atividades (ou recursos) em um ou mais períodos. O movimento 2 é definido como a remoção de uma atividade (ou recurso) de uma locação e associação desta atividade (ou recurso) em outra locação que esteja disponível em um ou mais períodos. O movimento 3 é uma

associação dos movimentos 1 e 2.

Foram definidas também duas listas tabu: uma lista de atividades e uma lista de recursos ociosos. As listas tabu guardam os movimentos recentes.

O critério de aspiração é definido como um movimento que gere uma solução de custo menor que a melhor solução encontrada.

A partir de uma solução inicial, a busca tabu avalia todos os movimentos de atividade e recursos ociosos e identifica aquele que retorna menor custo. Se esse movimento for não tabu ou um movimento permitido pelo critério de aspiração então ele é executado e a solução correspondente é definida como solução corrente na próxima iteração. Caso contrário, o próximo melhor movimento não tabu é considerado como solução corrente na próxima iteração.

4. Algoritmos Propostos

Nesta seção são propostos novos módulos para as etapas de construção e busca local para uma heurística GRASP aplicado ao PADE. Propomos dois métodos de construção e dois métodos de busca local.

A metaheurística GRASP foi proposta em 1995 por Feo e Resende (1995) e apesar de recente tem se mostrado uma das melhores heurísticas da literatura (Silva et al., 2006; Silva et al., 2007). GRASP é um método iterativo onde cada iteração é composta de duas etapas: construção de uma solução e busca local. A fase de construção do GRASP é iterativa, gulosa, randômica e adaptativa. Ela é iterativa porque constrói uma solução elemento a elemento e é adaptativa pois a escolha do próximo elemento da solução parcial é influenciada pelas escolhas anteriores. Para a seleção do próximo elemento da solução parcial, a princípio todos os candidatos são considerados, contudo o número de candidatos pode em muitos casos ser extremamente elevado, por isso, normalmente, é considerada apenas uma lista restrita de candidatos (LRC). O tamanho da LRC é $p = 1 + \alpha\ (a - 1)$, onde a é o número total de elementos candidatos e α um parâmetro de entrada. A partir de uma LRC, seleciona-se um elemento deste conjunto aleatoriamente e não necessariamente o melhor. Esta escolha aleatória permite que este procedimento possa ser usado várias vezes para obter soluções distintas. Pela forma como são construídas as soluções na etapa inicial, estas não representam na maioria dos casos um ótimo local, daí ser fortemente recomendada uma etapa de busca local. As iterações do GRASP são totalmente independentes, esse é o motivo de ser também conhecida como uma heurística do tipo *multistart*. O critério de parada do GRASP normalmente é o número máximo de iterações e a solução final GRASP é a melhor solução obtida ao final de sua execução.

4.1. Algoritmo de construção gFA

O algoritmo de construção é baseado na heurística FA apresentada na seção 3.1. Uma solução é construída da seguinte forma: Para cada atividade a do período p é criada uma LRC contendo os espaços de trabalho disponíveis. Um espaço de trabalho é escolhido aleatoriamente a partir da LRC. A quantidade de elementos da LRC é definida pelo número de espaços disponíveis para a atividade e α é sempre igual a 1.

4.2. Algoritmo de construção gFA1

Difere de gFA na ordenação e tamanho da LRC. Nesta heurística, para cada atividade a do período p são examinados os espaços de trabalho disponíveis. Uma LRC para cada atividade é criada, ordenada em ordem crescente da soma das distâncias para os outros espaços de trabalho. Isto é, o espaço de trabalho disponível mais bem localizado é o primeiro elemento da

LRC e assim por diante. O tamanho da LRC é determinado por $min\{disp, \alpha \times tam\_et\}$, onde *disp* corresponde a quantidade de espaços de trabalho disponíveis e *tam_et*, ao número de locações destinadas a alocação de atividades. Um espaço de trabalho é escolhido aleatoriamente a partir da LRC.

4.3. Algoritmo de busca local bG

O algoritmo de busca local bG utiliza a mesma vizinhança da busca tabu de McKendall e Jaramillo (2006) descrito na seção 3.3. A busca bG avalia os movimentos possíveis de atividades e recursos considerando a solução corrente. O melhor movimento é realizado e esta passa a ser a solução corrente na próxima iteração. A busca pára quando é feito um número de iterações sem melhora.

4.4. Algoritmo de busca local bG1

É uma variação de bG. Considerando-se uma solução corrente, são avaliados todos os movimentos de atividades e, para cada um, usada a Heurística de Alocação Rândomica, definida em McKendall e Jaramillo (2006) para realocar os recursos ociosos, considerando as alterações feitas pelos movimentos na matriz de atividades. O melhor movimento é realizado e a solução correspondente passa a ser a solução corrente. O critério de parada é o mesmo de bG..

4.5. Algoritmos GRASP para o PADE

Combinamos os algoritmos descritos nas seções 4.1 e 4.2 e formamos as seguintes heurísticas GRASP, apresentadas na Tabela 1.

Algoritmo	Construção	Busca Local
grasp1	gFA	bG
grasp2	gFA	bG1
grasp3	gFA1	bG1

Tabela 1 – Algoritmos GRASP para o PADE

5. Resultados Computacionais

Utilizando as 96 instâncias (P01 – P96) propostas por (McKendall et al., 2006) foi realizada uma bateria de testes. Todos os algoritmos descritos neste trabalho foram implementados, a saber, heurísticas FA, MFA, RC, MRC e a busca tabu. Neste momento, verificamos que seguindo fielmente a descrição dos autores, incluindo os parâmtros descritos no trabalho, as heurísticas RC e MRC geram soluções inviáveis para o problema e os algoritmos FA e MFA geram soluções de custo diferente aos reportados no trabalho. Buscando esclarecimento, entramos em contato com os autores e não obtivemos resposta. Desta forma, consideramos neste trabalho, para efeito de comparação, somente a heurística FA e a busca tabu implementados de McKendall e Jaramillo (2006).

Toda implementação foi feita utilizando linguagem de programação C usando a versão 4.0.3 do compilador GNU gcc. Os experimentos foram feitos em um Dual Intel Pentium 4, 3192 Mhz com 1010 Mbytes de RAM usando sistema operacional Ubuntu Linux. 6.0.4.

Os algoritmos propostos foram rodados utilizando os seguintes parâmetros: grasp1 e grasp2 com 500 iterações e α igual a 1 e grasp3 com 500 iterações e valor de α igual a 0.6. O número de iterações sem melhora das buscas locais propostas é igual a 5. Esses parâmetros foram definidos após testes preliminares.

Os resultados apresentados daqui em diante são demonstrados como diferença média entre os

resultados obtidos pelos algoritmos. O cálculo da diferença foi feito utilizando-se a fórmula:

média[(*valor_obtido_algoritmo* - *menor_valor*) / *valor_obtido_algoritmo**100], (9)

onde:

valor_obtido_algoritmo é o custo obtido pelo algoritmo para a instância *i* (*i* = [1, 96]) e,

menor_valor é o menor custo obtido comparando-se todos os algoritmos para instância *i*.

Apresentamos, na Tabela 2, os resultados médios dos algoritmos propostos grasp1, grasp2 e grasp3 comparados ao algoritmo de construção FA + busca tabu, ambos de McKendall e Jaramillo (2006) implementados, a que denominamos FA+BT. A coluna 1 indica o algoritmo testado e a coluna 2, a diferença média percentual, calculada conforme a equação 9.

Algoritmo	**Diferença média (%)**
FA+BT	15,4
grasp1	1,8
grasp2	0,9
grasp3	2,2

Tabela 2 – Diferença média entre os algoritmos comparados.

Verificamos, pela Tabela 2, que nenhum dos algoritmos obtiveram a melhor solução em todas as instâncias testadas, mas o algoritmo proposto grasp2 obteve, em média, os melhores resultados quando comparado aos demais. Podemos notar que, a diferença percentual média do algoritmo da literatura implementado FA+BT é de 15,4% e dos algoritmos propostos é de, no máximo, 2,2%.

Em McKendall et. Al (2006) são reportados resultados exatos para 25 das 96 instâncias (P1 – P24 e P27). Os algoritmos propostos grasp1, grasp2, grasp3 obtiveram solução ótima 22, 23, 25 vezes, respectivamente para estas 25 instâncias onde o ótimo é conhecido, enquanto FA+BT obteve solução exata em 3 instâncias. Das 96 instâncias testadas, os algoritmos propostos grasp1, grasp2, grasp3 obtiveram melhor resultado em 46, 63, 46 vezes, respectivamente.

Quanto ao tempo computacional exigido, as heurísticas GRASP propostas se mostraram muito mais rápidas. Dentre estas, o algoritmo grasp2 é o mais eficiente em se tratando de tempo de processamento. Por exemplo, para a instância P96, enquanto os algoritmos GRASP propostos, grasp1, grasp2, grasp3 necessitam de, aproximadamente, 19, 8, 9 minutos para chegarem a uma solução, o algoritmo FA+BT, leva, aproximadamente, 1.260 minutos para obter uma solução de custo 639. O algoritmo proposto mais rápido, grasp2, obtém em 8 minutos uma solução de custo 597, para a mesma instância. É necessário, no entanto, citar que a comparação de tempo entre a metaheurística GRASP e metaheurística busca tabu pode não ser a mais eficaz, já que as iterações do GRASP são diferentes da busca tabu. Formas alternativas de avaliar os tempos seria, talvez, um tempo fixo para ambas as estratégias ou ainda, computar o tempo de processamento até que um valor alvo fosse encontrada em cada uma delas. Tais simulações, ora em andamento, já vislumbram, no entanto, novamente uma maior eficiencia das heuristicas GRASP aqui propostas em relação a Busca Tabu da literatura.

6. Conclusões

O objetivo principal destre trabalho foi o de investigar o impacto da metaheurística GRASP aplicando-se algoritmos de construção e busca local para o Problema de Alocação Dinâmica de Espaços. Para isto, foram utilizadas adaptações de heurísticas e um conjunto de instâncias da literatura.

Na extensa bateria de testes realizados, conseguimos mostrar que as heurísticas GRASP propostas são competitivas para tratar o problema, tendo o algoritmo proposto grasp2 alcançado os melhores resultados para instâncias testadas em tempo computacional satisfatório.

Acreditamos, e pretendemos, em trabalhos futuros, poder sofisticar o método utilizando outros conceitos que tem se apresentado promissores quando aplicados a problemas da área de Pesquisa Operacional, como a idéia de GRASP Reativo (Prais e Ribeiro, 2000), reconexão por caminhos (Laguna e Martí, 1999; Silva et al., 2007), GRASP paralelo (Pardalos et al., 1995), filtro de soluções (Silva et al., 2006), entre outros.

Referências

FEO, T.A. & RESENDE, M.G.C. *Greedy randomized adaptive search procedures.* Journal of Global Optimization. Vol. 6, p. 109-133, 1995.

LAGUNA, M. & MARTÍ, R. *GRASP and path relinking for 2-layer straight line crossing minimization.* INFORMS Journal on Computing. Vol.11, p.44-52, 1999.

MCKENDALL, A.R. & JARAMILLO, J.R. *A tabu search heuristic for the dynamic space allocation problem*. Computers & Operations Research. Vol. 33, p.768-789, 2006.

MCKENDALL, A.R.; NOBLE, J.S. & KLEIN, C.M. *Simulated annealing heuristics for maninging resources during planned outages at eletric power plants.* Computers & Operations Research. Vol.32, p.107-125, 2006.

PARDALOS, P.; PITSOULIS, L.; MAVRIDOU, T. & RESENDE, M.G.C. *Parallel search for combinatorial optimization: genetic algorithms, simulated annealing and GRASP*. Parallel Algorithms for Irregularly Structured Problems, Springer Verlag, p.317-331, 1995.

PRAIS, M. & RIBEIRO, C.C. *Reactive GRASP: An application to a matriz decomposition problem in TDMA traffic assignment.* INFORMS Journal on Computing. Vol.12, p.164-176, 2000.

RESENDE, M.G.C & RIBEIRO, C.C. *Greedy randomized adaptive search procedures.* Handbook of Metaheuristics. Kluwer Academic Publishers, p.219-249, 2003.

SILVA, G.C., OCHI, L.S. & MARTINS, S.L. *Proposta e Avaliação de Heurísticas GRASP para o Problema da Diversidade Máxima.* Pesquisa Operacional. Vol. 26(2), p.321-360, 2006.

SILVA, G.C., ANDRADE; M.R.Q., OCHI, L.S.; MARTINS, S.L. & PLASTINO, A. *New heuristics for the maximum diversitt problem*. Journal of Heuristics - SPRINGER, aguarda impressão, 2007.