10th Brazilian Congress on Computational Intelligence (CBIC'2011), November 8 to 11, 2011, Fortaleza, Ceará Brazil
© Brazilian Society on Computational Intelligence (SBIC)

RESOLUÇÃO DO PROBLEMA DAS P-MEDIANAS POR MEIO DE ALGORITMOS BASEADOS EM GRASP, ILS E MULTI-START

Gustavo Marques Zeferino, Flaviana M. de S. Amorim, Marcone Jamilson Freitas Souza, Moacir F. de F. Filho e Sérgio Ricardo de Souza

Centro Federal de Educação Tecnológica de Minas Gerais – CEFET/MG.
CEP: 30.510-000 -- Belo Horizonte -- MG – Brasil.
zeferino@lsi.cefetmg.br, flaviana@dppg.cefetmg.br, marcone@iceb.ufop.br, franca@des.cefetmg.br e sergio@dppg.cefetmg.br

Abstract – This work addresses the *p*-Median Problem by the algorithms based on metaheuristics *Greedy Randomized Adaptive Search Procedure* (GRASP), *Iterated Local Search* (ILS) e *Multi-Start*. These algorithms use the Fast Swap-Based algorithm as local search. The computational experiments were performed with two sets of instances of literature and the results showed that ILS algorithm is the best in relation to the time of execution and the final solution quality.
Keywords – *p*-Median Problem, GRASP, Iterated Local Search, Multi-Start.

Resumo – Este trabalho aborda o Problema das *p*-Medianas por meio de algoritmos baseados nas metaheurísticas *Greedy Randomized Adaptive Search Procedure* (GRASP), *Iterated Local Search* (ILS) e *Multi-Start*. Esses algoritmos utilizam, como método de busca local, o algoritmo *Fast Swap-based Local Search*. Os experimentos computacionais foram realizados com dois conjuntos de instâncias da literatura e mostraram que o algoritmo ILS apresenta o melhor desempenho em termos de tempo de execução e qualidade da solução.
Palavras chave – Problema das p-Medianas, GRASP, Iterated Local Search, Multi-Start.

1 Introdução

Este trabalho tem seu foco no Problema das *p*-Medianas (PPM) não-capacitado, em que o objetivo é escolher, dentre *n* nós de um grafo, um conjunto de *p* nós, denominados medianas, de modo a minimizar a soma das distâncias de cada nó restante até o nó mediana mais próximo. Na versão capacitada deste Problema, a cada nó mediana do grafo é associado um peso de capacidade máxima a ser satisfeito pela mediana escolhida. Neste caso, a soma das demandas de todos os nós cobertos por uma mediana não deve ultrapassar a capacidade de atendimento da mesma [8]. Exemplos de aplicações do PPM podem ser vistos em diversos problemas, como Sistemas de Distribuição [7], Atendimento de Clientes de uma Rede com Demanda Probabilística [4] e Sistemas de Informações Geográficas [8], dentre outros. Uma revisão a respeito do PPM pode ser encontrada em [11].

O PPM é um problema da classe NP-difícil [6] e, portanto, heurísticas são as alternativas utilizadas para resolver instâncias de maior porte do problema.

O restante deste artigo está estruturado como segue. Na seção 2 é apresentada a definição e uma formulação formal do problema. A seção 3 apresenta a metodologia adotada para resolver o PPM. Na seção 4 são apresentados os testes realizados com estes algoritmos utilizando várias classes de instâncias da literatura, sendo também realizada uma comparação gráfica de desempenho entre estes métodos. Por último, é feita a conclusão do trabalho na seção 5.

2 Definição do Problema

Seja $G = (V, A)$ um grafo não direcionado ponderado, em que V é o conjunto dos vértices do grafo e A é o conjunto das arestas, tendo associado a cada aresta o valor da distância (ou custo de viagem) entre dois vértices adjacentes. O objetivo do problema é encontrar p vértices, de forma a minimizar a distância entre estes p vértices selecionados e os vértices restantes. Cada vértice restante é ligado ou associado a somente um dos p vértices escolhidos, sendo este o vértice mais próximo, denominado como mediana.

2.1 Formulação Matemática

O Problema das *p*-Medianas pode ser formulado, segundo [2], como segue:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} w_i d_{ij} x_{ij}$$

$$\text{sujeito a: } \sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i$$

$$x_{ij} \le y_j \qquad \forall i, j$$

$$\sum_{j=1}^{n} y_j = p,$$

© Brazilian Society on Computational Intelligence (SBIC)

$$x_{ij} = 1 \; ou \; 0 \quad \forall i, j,$$
$$y_j = 1 \; ou \; 0 \quad \forall j$$

em que:

p = número de medianas a serem instaladas;
n = número total de nós de demanda;
w_i = demanda do nó i;
d_{ij} = distância entre o nó i e j;

$x_{ij} = \begin{cases} 1, \text{ se o nó i é atendido pela mediana do nó j;} \\ 0, \text{ caso contrário.} \end{cases}$

$y_j = \begin{cases} 1, \text{ se a mediana é instalada no nó j;} \\ 0, \text{ caso contrário.} \end{cases}$

A primeira restrição da formulação matemática do problema assegura que um nó de demanda é atendido por uma única mediana. A segunda afirma que um nó de demanda somente é atendido por uma mediana que esteja instalada. E a terceira restrição garante que são designadas apenas p medianas. As demais restrições definem que as variáveis envolvidas são binárias.

3 Metodologia

Esta seção apresenta os procedimentos propostos para a solução do PPM. Inicialmente, é apresentada a estrutura de dados utilizada para a representação de uma solução. Em seguida, são detalhados, brevemente, as três metaheurísticas implementadas para a solução do PPM, a saber, *Greedy Randomized Adaptative Search Procedure* (GRASP), *Iterated Local Search* (ILS) e *Multi-Start*. Por fim, é apresentada a técnica de busca local utilizada como heurística de refinamento dessas metaheurísticas.

3.1 Representação da Solução

Para representar uma solução foi definida uma matriz com *n* colunas e três linhas. A primeira linha contém os índices dos vértices, sendo que as *p* primeiras posições representam os índices dos vértices que são classificados como medianas. Cada configuração destes *p* primeiros índices, desconsiderando ordem, representa uma possível solução para o problema. A segunda linha armazena o índice da mediana que atende o vértice com índice definido na primeira linha e a terceira linha armazena a distância entre a mediana e o vértice.

Para explorar o espaço de soluções foram utilizados movimentos de troca, que consistem em trocar um vértice que é mediana por um que não o seja.

3.2 Metaheurísticas

Foram propostos três algoritmos metaheurísticos, os quais são detalhados nas seções seguintes.

3.2.1 GRASP

Greedy Randomized Adaptative Search Procedure -- GRASP [13] é uma metaheurística que consiste na aplicação iterativa de duas fases: construção e refinamento, retornando a melhor das soluções obtidas ao longo da busca.

Na fase de construção, é gerada uma solução parcialmente gulosa por meio de uma função guia. A aleatoriedade da construção é controlada por um parâmetro real $\alpha \in [0,1]$. Para $\alpha = 1$, tem-se uma solução totalmente aleatória; para $\alpha = 0$, tem-se uma solução gulosa. Após a construção da solução, é aplicado o método de busca local apresentado na seção 3.3 para refinar a solução construída.

A construção de uma solução no método GRASP consiste em inserir elementos, obedecendo a um valor calculado pela função guia g(.) e a uma regra de seleção que contém um fator aleatório. Para o PPM, este método consiste em selecionar somente p vértices para serem medianas.

Para cada inserção, é definida uma lista C com os vértices remanescentes. Para cada vértice, é calculado um valor, através da função guia g(t), que consiste no somatório das distâncias do vértice $t \in C$ para todos os demais vértices $i \in C \setminus \{t\}$. Alguns destes vértices são selecionados para uma outra lista, chamada de Lista Restrita de Candidatos (LRC), pela seguinte regra:

$$LRC = \{t \in C \mid g(t) \leq g(t_{\min}) + \alpha(g(t_{\max}) - g(t_{\min}))\}$$

© Brazilian Society on Computational Intelligence (SBIC)

Com $t_{\max} = \arg\max_{t \in C} g(t)$ e $t_{\min} = \arg\min_{t \in C} g(t).$

Em que t_{max} , t_{min} e α é um valor real no intervalo [0, 1] definido pelo usuário. Com a LRC construída, é calculado o valor de aptidão para cada um dos vértices pertencentes a ela. A aptidão de um elemento t da lista LRC é calculada pela expressão:

$$f(t) = \frac{1/g(t)}{\sum_{t \in LRC} 1/g(t)}$$

A aptidão representa a probabilidade do vértice t da lista LRC ser escolhido como mediana. Desta forma, é gerado uma variável aleatória e com ela é feito um sorteio para a definição de qual vértice será escolhido como mediana. Todo o processo é repetido até que todas as *p* medianas sejam escolhidas.

Após a construção, a solução gerada por este procedimento é melhorada pela heurística de refinamento descrita na seção 3.3.

Esse processo de construção e refinamento é aplicado 10 vezes, retornando-se a melhor das soluções obtidas.

3.2.2 ILS

A metaheurística *Iterated Local Search* – ILS [9] consiste em partir de uma solução inicial s_0, previamente obtida a partir da utilização de um algoritmo de construção; aplicar um procedimento de busca local a essa solução; e, para escapar do ótimo local gerado, aplicar perturbações nesse ótimo local. No algoritmo ILS implementado, a solução inicial é o melhor resultado obtido dentre 20 iterações da fase de construção GRASP.

Em seguida, foi aplicada a fase de refinamento do ILS. A busca local utilizada é a descrita na seção 3.3. Para modificar a solução s obtida, são feitas perturbações na mesma, gerando-se as soluções perturbadas s'. Sobre essas soluções, é aplicado o procedimento de refinamento, gerando um ótimo local s". A decisão sobre qual solução será aplicada a próxima perturbação é feita pelo critério de aceitação. O critério de aceitação adotado é o de melhora no valor da solução corrente s, isto é, uma solução s" é aceita para ser a nova solução s corrente se f(s") < f(s).

No ILS, a intensificação é obtida em perturbações feitas na solução corrente. Já a diversificação é obtida quando se aceita qualquer solução s" e são aplicadas perturbações maiores na solução ótima corrente. O êxito deste método está diretamente associado à definição do procedimento de busca local, do procedimento de perturbação aplicado à solução atual e do critério de aceitação das soluções. Uma perturbação no algoritmo implementado consiste em executar um movimento de troca aleatória entre uma mediana alocada (ou "aberta") e uma mediana candidata (ou "fechada"). Além disso, as perturbações são executadas em níveis, isto é, para cada nível i de perturbação, são realizadas Nivel_iter iterações em relação à solução ótima local corrente. Este valor foi fixado em Nivel_iter = 20. As perturbações são realizadas de Nivel_min_per a Nivel_max_per níveis. Neste trabalho, estes valores foram fixados em Nivel_min_per = 3 e Nivel_max_per = 10.

3.2.3 Multi-Start

Multi-Start [10] é uma metaheurística simples, consistindo em gerar soluções aleatórias e melhorá-las, por meio de uma heurística de refinamento. O método de busca local adotado é o descrito na seção 3.3. A solução inicial é gerada de forma aleatória com distribuição uniforme. Cada vez que uma melhor solução é encontrada, ela é armazenada.

Como critério de parada, foi adotado o número de 100 iterações sem melhora, ou seja, o método para quando a melhor solução se mantém inalterada por 100 iterações.

3.3 Busca local

Para este trabalho foi implementada a busca local que realiza sempre a melhor troca possível e que repete este processo até que não exista mais alguma troca que melhore o valor da função objetivo (distância entre as medianas e os vértices). Esta heurística é conhecida como Método da Descida (ou *Best Improvement Method*). Neste trabalho, foi implementada um versão deste método, descrita em [12], conhecida como *Fast Swap-based Heuristic*. A heurística calcula o ganho, para cada vértice que não é mediana, caso esta se torne uma mediana, e calcula, para cada mediana, a perda, caso esta deixe de ser mediana. Além disto, é calculado um fator que considera o impacto total na solução caso se realize uma determinada troca. Com estes valores, é possível estimar o ganho de cada troca e efetivar aquela que possui o maior ganho. A grande vantagem deste método é que não é necessário realizar a troca em si para estimar esses valores. Um dos motivos é que as informações calculadas não são recalculadas a cada iteração e, sim, somente atualizadas.

4 Resultados

Os algoritmos GRASP, ILS e *Multi-Start* foram implementados na linguagem C++ e testados em um computador com processador Pentium Intel(R) Core(TM)2 Quad Q8400, com clock de 2.66 GHz, 3,7 GB de RAM, Kernel Linux 2.6.32-30-generic, compilador GCC versão 4.4.3 e sistema operacional Ubuntu 10.04 64-bits. Para testar os algoritmos, foram usadas duas classes de instâncias: *OR-Library* [3] e *Koerkel* [2]. Para cada instância, cada algoritmo foi executado 50 vezes.

© Brazilian Society on Computational Intelligence (SBIC)

As tabelas em que são apresentados os resultados obtidos têm a coluna Instância que é subdividida em quatro outras colunas: a subcoluna Nome, que expressa o nome da instância; a subcoluna n, que representa o número de vértices contido na instância; a subcoluna p, representando os números de medianas que devem ser instaladas; e a subcoluna Ótimo, na qual está o valor ótimo encontrado na literatura correspondente a cada instância. Nas outras três colunas, são apresentados os resultados obtidos para cada algoritmo implementado GRASP, ILS e *Multi-Start*, com suas respectivas subcolunas Melhor, que lista o melhor resultado encontrado; subcoluna Erro %, que informa a porcentagem com que a média dos valores obtidos ficaram distantes do valor ótimo; e a subcoluna Tempo, indicando o tempo médio gasto pelos algoritmos, em segundos.

Pela Tabela 1, que contêm os resultados da instância *OR-Library*, verifica-se que o algoritmo ILS é capaz de encontrar todos os valores ótimos da classe de instâncias analisada. O algoritmo GRASP não alcançou o resultado ótimo apenas na instância pmed30. O algoritmo *Multi-Start*, por sua vez, não alcançou o resultado ótimo nas instâncias pmed19, pmed25, pmed30 e pmed40. Na tabela 2, estão os resultados obtidos para as instâncias *Koerkel*, em que é visto que o ILS alcançou valores menores que os da literatura para os conjuntos K1000-2 e K-1000-4. O algoritmo GRASP não alcançou nenhum dos resultados. E o algoritmo *Multi-Start*, por sua vez, alcançou um resultado menor para a instância K1000-2.

Com relação aos valores de erro médio encontrados, o algoritmo ILS apresenta valores menores ou iguais a 0,01% para 27 das instâncias *OR-Library* e, para todas as instâncias *Koerkel*. Entre 0,01% e 0,1% são encontrados 11 resultados para as instâncias *OR-Library*; e maior que 0,1% para 2 resultados nas instâncias *OR-Library*. Quanto ao algoritmo GRASP, os valores de erro médio menores ou igual a 0,01% são encontrados em 32 instâncias *OR-Library* e, em todas as instâncias *Koerkel*; entre 0,01% e 0,1% para 5 instâncias *OR-Library*; e maior que 0,1% é visto em 2 instâncias *OR-Library*. Por último, no algoritmo *Multi-Start*, os valores de erro médio menores ou igual a 0,01% são encontrados em 32 instâncias *OR-Library* e em todas as instâncias *Koerkel*; entre 0,01% e 0,1% para seis instâncias *OR-Library*; e maior que 0,1% para duas instâncias *OR-Library*.

Com relação ao tempo, nas instâncias *OR-Library*, os três algoritmos alcançaram o valor rapidamente para as instâncias com n = 100. O ILS obteve o menor tempo na pmed4 *OR-Library* e na K1000-2, com os tempos de 0,2649 e 12,2418 segundos, respectivamente. O GRASP obteve em 0,3276 segundos o valor ótimo da instância pmed1. Porém, seu menor tempo nas instâncias de *Koerkel* foi de 22,9774 segundos na K1000-2, sem alcançar o ótimo. Já o algoritmo *Multi-Start* obteve seu menor tempo de 0,7380 e 33,2020 segundos para alcançar o valor ótimo nas instâncias pmed1 e K1000-2, respectivamente. Nesta última chegou a diminuir o valor da literatura.

De forma a comparar esses algoritmos com relação ao tempo necessário para encontrar um valor alvo, foram feitos experimentos, segundo a abordagem indicada em [1]. Para execução dos experimentos, utilizou-se a instância pmed30 *OR-Library* e a instância K1000-4 *Koerkel*, cujos valores ótimos são, respectivamente, 1989 e 32.110.068. Cada algoritmo foi executado 50 vezes, sendo interrompido após alcançar o valor alvo, no caso, os valores 2000 e 32.493.567, respectivamente. Não foram permitidos tempos de execução repetidos; assim, os tempos repetidos foram descartados e uma nova execução foi feita.

Nas figuras 1 e 2, verifica-se que o algoritmo ILS é o que alcança o valor alvo mais rapidamente nas classes de instâncias em estudo. Na instância pmed30 *OR-Library*, enquanto o valor alvo é alcançado instantaneamente pelo ILS, este mesmo alvo é alcançado pelo GRASP em cerca de 250 segundos e pelo *Multi-Start* em quase 800 segundos. Este resultado é também corroborado pelos valores apresentados na Tabela 1 para a mesma instância, uma vez que o algoritmo ILS alcançou, em 50 execuções, o valor ótimo em 209,152 segundos, tendo um erro médio de 0,1398%. Semelhantemente aos resultados apresentado acima, o ILS na instância *Koerkel*, K1000-4, alcança o alvo rapidamente, o mesmo só é alcançado pelo GRASP após 100 segundos e pelo *Multi-Start* depois dos 250 segundos. Este resultado é também corroborado pelos valores apresentados na Tabela 2.

5 Conclusões

Este trabalho tratou o Problema das p-Medianas não-capacitado por meio de três algoritmos, baseados em GRASP, ILS e *Multi-Start*.

O algoritmo GRASP consiste na aplicação iterativa de duas fases, construção e refinamento, retornando a melhor das soluções obtidas ao longo da busca. Nas instâncias *OR-Library* alcançou valores ótimos em 97,5% e nas instâncias *Koerkel* não alcançou nenhum valor ótimo. Os maiores erros percentuais foram de 2.569,103 e 0,0153 nas respectivas instâncias pmed30 e K1000-6, tendo o erro médio, dentre cada uma das classes de instâncias dado por 0,0208 e 0,0087.

O algoritmo ILS implementado utiliza, para geração da solução inicial, a fase de construção da metaheurística GRASP. Ele apresentou resultados superiores aos encontrados para as implementações de *Multi-Start* e GRASP, alcançando valores ótimos em 100% dos 40 conjuntos de instâncias *OR-Library*, já nas instâncias *Koerkel* não alcançou nenhum valor ótimo. Os maiores erros percentuais foram de 0,1398 e 0,0088 nas respectivas instâncias pmed30 e K1000-12, tendo o erro médio, dentre cada uma das classes de instâncias dado por 0,0187 e 0,0049.

O algoritmo *Multi-Start*, mesmo sendo baseado em uma metaheurística simples, em que sua solução inicial, neste trabalho, é gerada de forma aleatória, com distribuição uniforme, conseguiu alcançar, dentre os 40 conjuntos das instâncias *OR-Library*, 90% e nas instâncias *Koerkel* não alcançou nenhum valor ótimo. Os maiores erros percentuais foram de 0,3519 e 0,0346 nas respectivas instâncias pmed30 e K1000-2, tendo o erro médio, dentre cada uma das classes de instâncias dado por 0,0206 e 0,0115.

Estes resultados mostram a superioridade da implementação realizada para o algoritmo baseado em ILS sobre os demais. Esta afirmativa se baseia no melhor desempenho no tocante ao número de valores ótimos alcançados, em que é

© Brazilian Society on Computational Intelligence (SBIC)

seguida pela GRASP e, por fim, pela *Multi-Start*. A superioridade da ILS é corroborada, por fim, pelos gráficos *time-to-target* apresentados nas figuras 1 e 2, em que o ILS alcança o valor alvo em tempo inferior ao dos demais métodos.

A principal vantagem do algoritmo ILS e do *Multi-Start*, é o menor tempo de processamento e, a simplicidade da implementação, respectivamente. Para as instâncias *OR-Library* e *Koerkel* a principal vantagem do GRASP reside na baixa variabilidade das soluções finais; e no ILS, no menor tempo de processamento requerido.

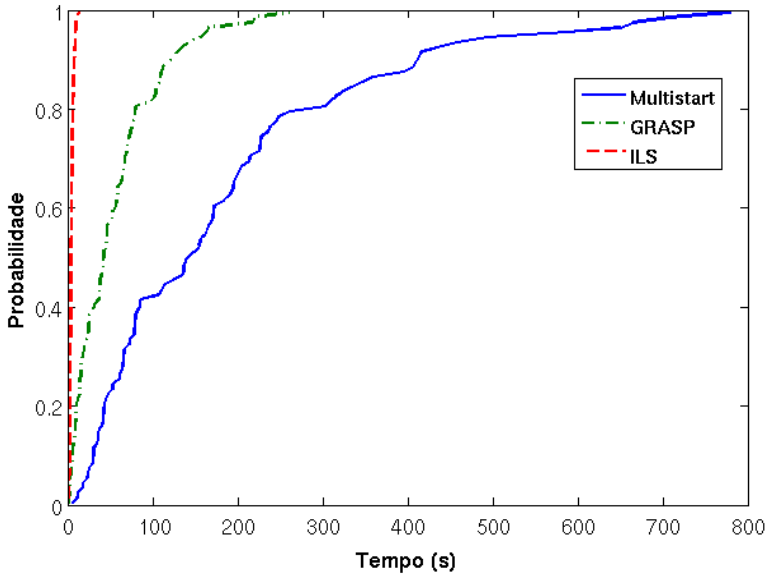

Figura 1: *Time-to-target* da instância pmed30 com alvo igual a 2000.

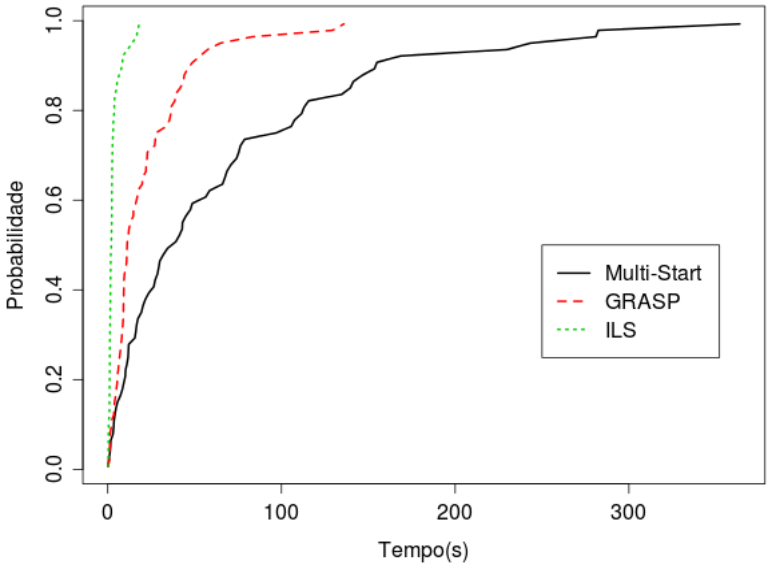

Figura 2: *Time-to-target* da instância K1000-4 com alvo igual a 32.493.567.

Agradecimentos

Os autores agradecem ao CEFET-MG, à CAPES, à FAPEMIG e ao CNPq pelo apoio ao desenvolvimento deste trabalho.

Referências

[1] Aiex, R. M.; Resende, M. G. C.; Ribeiro, C. C. (2002). Probability distribution of solution time in GRASP: An experimental investigation. *Journal of Heuristics*, 8:343–373.

© Brazilian Society on Computational Intelligence (SBIC)

[2] Alp, O.; Erkut, E.; Drezner, D. (2003). An efficient genetic algorithm for the p-median problem. *Annals of Operations Research*, 122:21–42.

[3] Beasley, J. (1985). A note on solving large p-median problems. European Journal of Operational Research, 31:270–273.

[4] Berman, O.; Wang, J. (2010). The network p-median problem with discrete probabilistic demand weights. *Computer & Operations Research*, 37(8): 1455–1463.

[5] Christofides, N. G. T. (1975). *An algorithmic approach*. New York: Academic Press Inc.

[6] Kariv, O.; Hakimi, L. (1979). An algorithmic approach to network location problems. ii: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3): 539–560.

[7] Klose, A.; Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research*, 162: 4–29.

[8] Lorena, L. A. N.; Senne, E. L. F.; Paiva, J. A. C.; Pereira, M. A. (2001). Integração de modelos de localização a sistemas de informações geográficas. *Gestão e Produção*, 8(2): 185-195.

[9] Lourenço, H. R.; Martin, O. C.; Stutzle, T. (2003). *Iterated local search*. In Glover, F. and Kochenberger, G., editors, Handbook of Metaheuristics, p. 321–353. Kluwer Academic Publishers, Boston.

[10] Martí, R. (2003). *Multi-start methods*. In Glover, F. and Kochenberger, G., editors, Handbook of Metaheuristics, p. 355–367. Kluwer Academic Publishers, Boston.

[11] Mladenovic, N.; Brimberg, J.; Hansen, P.; Moreno-Pérez, J. A. (2007). The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, 179(3): 927 – 939.

[12] Resende, M.; Werneck, R. (2003). A fast swap-based local search procedure for location problems. *Technical Report* TD-5R3KBH, AT&T Labs Research.

[13] Resende, M. G. C.; Feo, T. A. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 9: 849–859.

[14] Teitz, M. B.; Bart, P. (1968). Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, 16(5): 955-961.

[15] Whitaker, R. A. (1983). A fast algorithm for the greedy interchange of large-scale clustering and median location problems. *INFOR*, 21:95-108.

© Brazilian Society on Computational Intelligence (SBIC)

Apêndice:

Instâncias				GRASP			ILS			Multi-Start		
Nome	n	p	Ótimo	Melhor	Erro %	Tempo (s)	Melhor	Erro %	Tempo (s)	Melhor	Erro %	Tempo (s)
pmed1	100	5	**5819**	**5819**	0,000	0,33	**5819**	0,000	0,39	**5819**	0,000	0,74
pmed2	100	10	**4093**	**4093**	0,000	0,36	**4093**	0,000	0,30	**4093**	0,000	10,44
pmed3	100	10	**4250**	**4250**	0,000	0,35	**4250**	0,000	0,28	**4250**	0,000	0,90
pmed4	100	20	**3034**	**3034**	0,000	0,50	**3034**	0,000	0,26	**3034**	0,000	14,97
pmed5	100	33	**1355**	**1355**	0,000	0,70	**1355**	0,004	0,34	**1355**	0,000	22,14
pmed6	200	5	**7824**	**7824**	0,000	11,82	**7824**	0,000	15,53	**7824**	0,000	29,27
pmed7	200	10	**5631**	**5631**	0,000	12,87	**5631**	0,000	0,91	**5631**	0,000	39,87
pmed8	200	20	**4445**	**4445**	0,000	18,77	**4445**	0,000	0,80	**4445**	0,000	63,67
pmed9	200	40	**2734**	**2734**	0,000	34,52	**2734**	0,003	0,93	**2734**	0,000	108,81
pmed10	200	67	**1255**	**1255**	0,000	59,68	**1255**	0,000	13,14	**1255**	0,000	179,49
pmed11	300	5	**7696**	**7696**	0,000	25,34	**7696**	0,000	34,81	**7696**	0,000	72,45
pmed12	300	10	**6634**	**6634**	0,000	30,48	**6634**	0,000	22,95	**6634**	0,000	107,24
pmed13	300	30	**4374**	**4374**	0,000	57,92	**4374**	0,000	17,50	**4374**	0,000	224,63
pmed14	300	60	**2968**	**2968**	0,000	127,96	**2968**	0,011	23,61	**2968**	0,000	563,67
pmed15	300	100	**1729**	**1729**	0,074	291,27	**1729**	0,042	35,37	**1729**	0,046	1273,13
pmed16	400	5	**8162**	**8162**	0,000	51,59	**8162**	0,000	68,55	**8162**	0,000	137,71
pmed17	400	10	**6999**	**6999**	0,000	53,78	**6999**	0,000	40,15	**6999**	0,000	218,77
pmed18	400	40	**4809**	**4809**	0,000	148,10	**4809**	0,018	32,23	**4809**	0,004	729,10
pmed19	400	80	**2845**	**2845**	0,030	374,04	**2845**	0,051	45,32	2846	0,049	1704,17
pmed20	400	133	**1789**	**1789**	0,002	583,10	**1789**	0,025	71,36	**1789**	0,000	2072,72
pmed21	500	5	**9138**	**9138**	0,000	67,18	**9138**	0,000	84,47	**9138**	0,000	208,04
pmed22	500	10	**8579**	**8579**	0,000	89,44	**8579**	0,000	63,31	**8579**	0,000	366,73
pmed23	500	50	**4619**	**4619**	0,000	260,77	**4619**	0,005	52,04	**4619**	0,000	1306,81
pmed24	500	100	**2961**	**2961**	0,010	753,96	**2961**	0,053	76,27	**2961**	0,007	2849,70
pmed25	500	167	**1828**	**1828**	0,169	1465,79	**1828**	0,132	135,36	1829	0,170	5194,98
pmed26	600	5	**9917**	**9917**	0,000	105,14	**9917**	0,000	145,55	**9917**	0,000	311,74
pmed27	600	10	**8307**	**8307**	0,000	128,10	**8307**	0,000	94,46	**8307**	0,000	549,04
pmed28	600	60	**4498**	**4498**	0,007	582,25	**4498**	0,033	78,61	**4498**	0,009	2838,59
pmed29	600	120	**3033**	**3033**	0,053	1428,23	**3033**	0,071	121,56	**3033**	0,063	5778,29
pmed30	600	200	**1989**	1993	0,390	2569,10	**1989**	0,140	209,15	1992	0,352	9302,32
pmed31	700	5	**10086**	**10086**	0,000	139,66	**10086**	0,000	189,79	**10086**	0,000	451,82
pmed32	700	10	**9297**	**9297**	0,000	173,68	**9297**	0,000	124,23	**9297**	0,000	798,66
pmed33	700	70	**4700**	**4700**	0,003	944,96	**4700**	0,018	113,86	**4700**	0,000	4933,34

© Brazilian Society on Computational Intelligence (SBIC)

pmed34	700	140	**3013**	**3013**	0,052	2357,22	**3013**	0,066	182,14	**3013**	0,066	8320,78
pmed35	800	5	**10400**	**10400**	0,000	244,13	**10400**	0,000	248,46	**10400**	0,000	601,83
pmed36	800	10	**9934**	**9934**	0,000	289,07	**9934**	0,000	160,74	**9934**	0,000	1066,43
pmed37	800	80	**5057**	**5057**	0,008	1581,09	**5057**	0,026	135,22	**5057**	0,016	7917,05
pmed38	900	5	**11060**	**11060**	0,000	269,55	**11060**	0,000	296,02	**11060**	0,000	787,13
pmed39	900	10	**9423**	**9423**	0,000	274,28	**9423**	0,000	162,91	**9423**	0,000	1345,97
pmed40	900	90	**5128**	**5128**	0,036	2271,41	**5128**	0,050	185,56	5129	0,043	10862,55

Tabela 1: Resultados obtidos para cada Metaheurística, aplicadas nas instâncias *OR-Library* [3]. Para cada instância e método, foram realizados 50 execuções.

Instâncias				GRASP			ILS			Multi-Start		
Nome	n	p	Ótimo	Melhor	Erro %	Tempo (s)	Melhor	Erro %	Tempo (s)	Melhor	Erro %	Tempo (s)
K1000-2	1000	2	**46.118.255**	46.120.842	0,007	22,98	**46.118.254**	0,001	12,24	**46.118.254**	0,035	33,20
K1000-4	1000	4	**32.110.068**	32.174.931	0,010	54,42	32.117.422	0,006	18,29	32.252.865	0,008	42,68
K1000-6	1000	6	**26.007.551**	26.232.909	0,015	75,59	**26.007.550**	0,007	17,68	26.330.005	0,016	53,22
K1000-8	1000	8	**22.251.618**	22.569.395	0,014	93,69	22.270.385	0,007	18,06	22.604.481	0,016	58,75
K1000-10	1000	10	**19.706.508**	20.146.728	0,008	142,90	19.767.587	0,007	18,85	20.139.178	0,014	74,14
K1000-12	1000	12	**17.804.044**	18.226.542	0,011	165,32	17.818.307	0,009	18,27	18.313.325	0,009	90,71
K1000-17	1000	17	**14.785.148**	15.136.437	0,012	186,19	14.827.214	0,006	18,17	15.190.052	0,011	99,63
K1000-25	1000	25	**12.004.788**	12.376.835	0,007	388,78	12.028.966	0,006	17,91	12.431.766	0,007	130,84
K1000-50	1000	50	**8.036.540**	8.337.125	0,009	492,27	8.071.403	0,003	21,10	8.370.953	0,006	248,13
K1000-111	1000	111	**4.810.938**	5.047.303	0,005	1016,07	4.817.217	0,004	32,47	5.044.472	0,008	595,04
K1000-143	1000	143	**4.019.654**	4.193.801	0,007	1460,51	4.028.969	0,003	39,38	4.201.144	0,008	772,54
K1000-200	1000	200	**3.117.365**	3.250.501	0,003	2529,02	3.125.671	0,004	44,50	3.241.712	0,008	1489,55
K1000-333	1000	333	**1.923.368**	1.987.126	0,005	3957,13	1.928.882	0,002	77,11	1.984.547	0,005	3667,57

Tabela 2: Resultados obtidos para cada Metaheurística, aplicadas nas instâncias *Koerkel* [2]. Para cada instância e método, foram realizados 50 execuções.