Busca Local com permuta *All Pairs* aplicada ao problema de alocação de facilidades.

Tarcísio Barroso Marques[1], Lucas de Souza Siqueira[1], Rodrigo Oliveira Zacarias[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia Fluminense (IF Fluminense)
Campus Itaperuna – 28.300-000 – Itaperuna – RJ – Brasil

tarcisio.marques@iff.edu.br, {lucassouza.ti,rodrigo.olizac}@gmail.com

Abstract. *This article presents a Local Search that makes the All Pairs exchange, applied to the problem of allocation of facilities, treated as the problem of p-medians. The objective is to allocate a fixed number of facilities (medians) at strategic points in a region to minimize the total distance involved, between the open facilities and the demand points served, applying the Local Search and aiming at the improvement of the initial solution. The project presented in this article can be used by other Meta-heuristics as the Genetic Algorithm, through the creation of an initial population already improved by the Local Search process, among others.*

Resumo. *Este artigo apresenta uma Busca Local que faz a permuta All Pairs, aplicada ao problema de alocação de facilidades, tratado como o problema das p-medianas. O objetivo é alocar um número fixo de facilidades (medianas) em pontos estratégicos de uma região para minimizar a distância total envolvida, entre as facilidades abertas e os pontos de demanda atendidos, aplicando-se a Busca Local, visando a melhoria da solução inicial. O trabalho apresentado neste artigo poderá ser usado por outras Meta-heurísticas, como o Algoritmo Genético, através a criação de uma população inicial já melhorada pelo processo da Busca Local, dentre outros.*

1. Introdução

O processo decisório é uma atividade presente no cotidiano de muitos profissionais, principalmente daqueles que ocupam funções estratégicas dentro de uma organização. No ambiente dos negócios, para um empresário que deseja expandir suas atividades, decidir sobre onde instalar uma nova filial ou como melhor atender seu mercado consumidor envolve soluções que podem gerar altos custos se não forem otimizadas.

Nesse contexto, este trabalho, fomentado pelo Instituto Federal Fluminense *Campus* Itaperuna-RJ, apresenta o problema de alocação de facilidades. Alocar facilidades consiste na decisão sobre onde posicionar espacialmente bens ou serviços de forma que a distância entre o ponto de alocação e os pontos de mercado consumidor seja o menor possível, com o objetivo de minimizar os custos de transporte ou transmissão, respeitando os critérios ou regras de negócio da organização [Ribeiro e Arroyo 2008].

Segundo Ribeiro (2008), facilidade é um termo utilizado para simbolizar escolas, postos de saúde, postos de gasolina, hospitais, centros de distribuição, áreas de lazer, etc. Cada facilidade possui um cliente ou usuário que será o consumidor dos bens ou serviços oferecidos. Dessa forma, clientes podem ser grupos de pessoas, bairros, cidades, depósitos, entre outros. As facilidades podem ser divididas em duas categorias: as que representam novos pontos a serem instalados em uma região e as que representam a seleção de um ponto dentro de um conjunto de centros já existentes [Lorena et al. 2001].

Dentre as formulações que envolvem os problemas de alocação de facilidades, o problema das p-medianas destaca-se pelo grande número de aplicações práticas. Suas primeiras formulações foram realizadas por Hakimi (1964, 1965) e têm o objetivo de minimizar a soma total dos custos de alocação entre facilidades e pontos de demanda. O problema pode se apresentar de outras formas, tais como em relação à capacidade máxima de atendimento ou o custo fixo na localização das facilidades [Lorena e Senne 2003].

Para a resolução desses problemas relacionados à alocação de facilidades, heurísticas vêm sendo estudadas por diversos pesquisadores. Um exemplo dessas heurísticas foi utilizada no trabalho desenvolvido por Capdeville e Vianna (2013), que propuseram duas implementações da heurística GRASP para solucionar um problema de alocação de pontos de acesso de rede sem fio no espaço *indoor* de uma instituição de ensino, no intuito de fornecer uma maior cobertura de sinal para os usuários, considerando como critérios os conceitos de radiofrequência.

No caso de Brondani et al. (2013), é proposta uma heurística projetada especificamente, baseada no problema de p-medianas, para localizar um número de unidades de lazer na zona norte da cidade do Rio de Janeiro. Essa heurística é implementada com o auxílio do algoritmo de Floyd e com base na heurística de Pearl (1984), tendo resultados bastante satisfatórios.

Neste artigo será tratado o problema de p-medianas utilizando uma Busca Local que faz uma permuta *All Pairs*, um algoritmo que define uma vizinhança composta por um conjunto de soluções com características “mais próximas”. Neste sentido, o objetivo é alocar estrategicamente em pontos de uma região um número fixo de facilidades para minimizar a distância total relacionada às facilidades usadas e os pontos de demanda atendidos. Primeiramente, uma solução randômica é gerada e melhorada por meio da iteração com os elementos vizinhos até chegar a uma solução eficiente e de baixo custo computacional, formulando uma heurística de melhoria.

Para isso, foram utilizadas instâncias de problemas de médio e de grande porte, sendo algumas delas criadas manualmente no intuito de se conhecer a solução para que os resultados possam ser comparados.

2. Definição do Problema

Nesta seção apresenta-se a definição do problema de localização de facilidades, abordado neste trabalho.

Notações:

$P = \{1,...,n\}$: conjunto de pontos de demanda (um ponto de demanda pode ser um bairro ou um quarteirão de um bairro por exemplo);

$C = \{1,...,m\}$: conjunto de pontos potenciais onde podem ser alocadas as facilidades (se no ponto $j \in C$ é alocada uma facilidade, então é dito que a facilidade j é aberta, caso contrário a facilidade j está fechada);
$A = \{a_1, \ldots, a_k \mid m \geq k \in C\}$: conjunto de facilidades abertas.
$B = \{b_1, \ldots, b_y \mid m \geq y \in C\}$: conjunto de facilidades fechadas.
c_j: variável de decisão $\in \{0,1\}$. Se a facilidade j é aberta tem-se $a_j = 1$, caso contrário $a_j = 0$.
d_{ij}: distância (euclidiana) do ponto i $\in P$ com coordenadas (x_i, y_i) ao ponto j $\in A$ com coordenadas (x_j, y_j);

Formulação:

Minimizar $f(x) = \sum_{i=1}^{n} \min\{d_{ij} \mid j \in A\}$

A função objetivo em $f: \Omega \in \Re$ (Ω = conjunto das soluções viáveis, $\Re$ = conjunto dos números reais) busca aproximar os pontos de demanda às facilidades a serem abertas atendendo um ponto de demanda $p_i \in P$ pela facilidade $a_j \in A$ que esteja mais próxima.

3. A heurística Busca Local aplicada ao problema de alocação de facilidades.

No ramo da Ciência da Computação, a busca local é uma heurística computacional utilizada para resolver problemas de difícil otimização. A heurística consiste em percorrer uma vizinhança, que contém soluções com características semelhantes à solução atual, objetivando encontrar melhores resultados para uma função, normalmente chamada de *Função Objetivo*. Caso essa função melhore, outras buscas podem ser realizadas a partir desta melhoria, uma vez que resultados ainda melhores podem ser encontrados. Essa busca é realizada através de permutas com os vizinhos. É importante ressaltar que a *Busca Local* não faz todas as permutas possíveis ao problema e também, em muitos casos, encontra soluções piores a atual, dependendo muito do contexto e da vizinhança analisada.

A Figura 1 ilustra este universo de possibilidades. Supondo que se deseja minimizar uma função objetivo f, a solução atual x_0é testada e após várias iterações com os vizinhos, encontra-se uma solução melhor, x_1 que passa a ser a solução atual. Novas buscas são realizadas, agora com nova vizinhança e, com isso, a solução evolui até que se encontre x_3. No entanto, é importante observar que ao analisar a curva que representa o universo das possibilidades, entre a solução x_2 e x_3 existe um ponto mínimo x^* que a Busca Local não é capaz de encontrar, o que não quer dizer que a solução não tenha evoluído.

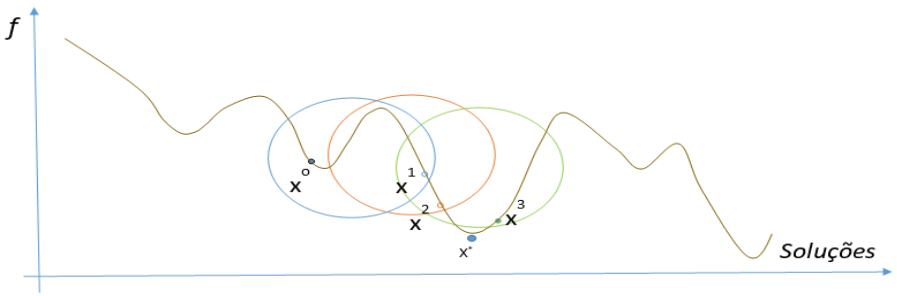

Figura 1. Gráfico representativo da Busca Local.

A busca local é aplicada a diversos tipos de problema, dentre eles pode-se citar: problema de coloração de grafo; problema do caixeiro viajante; problema de alocação de facilidades.

No trabalho de Lintzmayer et al. (2011), é apresentado um algoritmo heurístico baseado em colônias de formigas artificiais com busca local que tem como objetivo apresentar soluções para o problema de coloração de grafos. Arya et al. (2004) analisaram heurísticas de busca local para solução do problema das p-medianas e alocação de facilidades. Foi definido um intervalo de localidade para a busca local minimizar o problema com a máxima taxa de otimização.

Neste trabalho, o objetivo foi alocar um número fixo de facilidades em pontos estratégicos de uma região, minimizando a soma de todas as distâncias de cada ponto de demanda a sua facilidade mais próxima (mediana).

3.1. Geração da solução inicial

A busca local implementada parte de uma solução inicial gerada de forma aleatória. Após isso, há o início do processo de troca com os vizinhos. Seja *m,* o total de locais candidatos à instalação das facilidades, *k* o número de facilidades a serem abertas, *A* o conjunto que contém as facilidades abertas e *f(x)* a função que gera números inteiros pseudoaleatórios, representando uma facilidade sorteada para ser aberta ($f(x) \in N \mid f(x) \leq m$). Isso é apresentado no Algoritmo 1, em pseudocódigo.

Algoritmo 1. Geração aleatória de solução inicial.

```
Aleatoria(m, k) {
 01 A ← ∅, i ← 0;
 02 while (i < k) {
 03     usado ← false;
 04     aux ← f(x);
 05     for y ← 0 tom {
 06         if(A_y = aux)usado ← true;
 07     }
 08     if ( ! usado) {
 09         A_i ← aux;
 10         i++;
 11     }
 12 }
End Aleatoria
```

3.2. A Busca Local com a permuta *All Pairs*.

A Busca Local implementada inicialmente gera solução que consiste em, aleatoriamente, abrir certo número *k* de facilidades, dentro do universo de locais candidatos à instalação das mesmas. Dessa forma, tem-se um vetor *A*, representando as facilidades abertas e um vetor *B* representando as fechadas. A partir deste ponto, são aplicadas permutas entre *A* e *B*, com todos os pares vizinhos, *All Pairs*. A cada ciclo de trocas, a função objetivo é atualizada caso a solução tenha melhorado e uma nova busca é realizada. Todo esse processo pode ser visualizado na Figura 2.

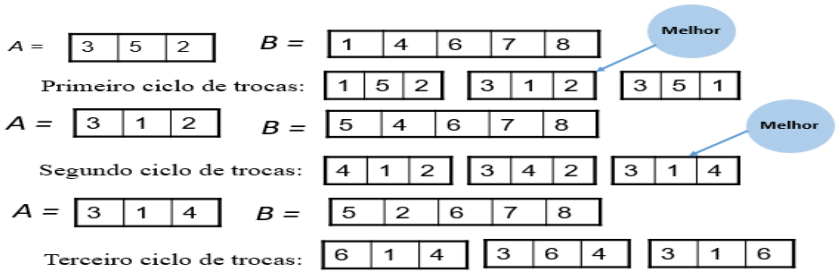

Figura 2. Lógica do Algoritmo de Busca Local *All Pairs* aplicada ao problema de alocação de facilidades.

O pseudocódigo exibido no Algoritmo 2 retrata a Busca Local com a permuta *All Pairs* aplicada ao problema. Nesse algoritmo, são fornecidos o vetor inicial *A,* que contém *k* facilidades abertas aleatoriamente, o vetor *B*, que contém *y* facilidades fechadas, e também o número mínimo de iterações desejadas, *iter*. Essa função realiza a troca de cada elemento com o vizinho adiante, e verifica-se se houve melhora na função objetivo $f(A)$, em relação à melhor das soluções encontradas até o momento, linha 07. Ao final de cada ciclo de troca, caso tenha havido melhoria, é atualizado definitivamente o vetor Great (G) e também o vetor das facilidades abertas A, linha 10. Ao final de todo o processo, é retornada a melhor solução encontrada.

Algoritmo 2. Busca Local com permuta *All Pairs*.

```
allPairs (A, B, iter){
01G ← A, H ←∅, S ←∅, melhora ← true, cont ← 0;
02 while ((melhora) or (cont<iter )){
03     melhora ← false;
04     for (i ← 0 tok){
05         for (j ← 0 toy){
06             troca ← A_i; A_i ← B_j; B_j ← troca; cont + +;
07             if(f(A)<f(G)) { H ← A; S ← B; melhora ← true; }
08             troca ← A_i; A_i ← B_j; B_j ← troca;
09         }
10         if (melhora){G ← H; A ← G; B ← S}
11     }
12 }
13 retorne G;
EndallPairs;
```

4. Testes Computacionais

Para testar a heurística implementada, foram geradas várias instâncias de problemas para alocar *k* facilidades em *m* locais candidatos de forma a atender *n* pontos de demanda, onde estes estão alocados de forma agrupada em blocos fazendo com que a solução ideal fique mais dedutível. Estas instâncias foram divididas em dois conjuntos de problemas:

Conjunto 1: problemas construídos manualmente onde se conhecem as soluções ótimas. Nesses problemas, foram gerados pontos estrategicamente posicionados com o objetivo de obter problemas de difícil solução.

Conjunto 2: problemas gerados de forma aleatória. Para esses problemas, as soluções ótimas não são conhecidas.

Todos os testes foram realizados em um computador core I5 com velocidade de processamento de 2.50 GHz e 06 GB de memória RAM. O algoritmo foi desenvolvido na linguagem Java, sendo executado no sistema operacional Linux, distribuição Linux Mint 18 Cinnamon de 64 bits.

A Tabela 1 exibe 10 instâncias do *Conjunto 1*, comparando o valor da solução ótima $f^*(x)$, e os valores $f(x)$ das soluções obtidas pela busca local. Nos 10 problemas testados, foram encontradas soluções muito próximas das ótimas onde, em 3 casos, as mesmas não tiveram nem 1% de diferença.

Tabela 1. Custo das melhores soluções encontradas pela Busca Local.

Nº	Prob. n_m_k	f*(x)	f(x)
1	10_10_2	826,99	1820,26
2	20_20_4	1507,72	1994,32
3	50_50_4	3745,89	4058,07
4	100_100_6	7897,03	8521,56
5	200_200_6	14856,22	15477,76
6	300_300_6	23429,17	23813,91
7	400_400_6	30107,75	30444,22
8	500_500_6	37858,69	38187,19
9	700_700_4	53096,03	53513,91
10	1000_1000_6	75666,21	76235,63

A Figura 3 exibe a solução gráfica ideal da instância de problema nº 10, seguida pela Figura 4, que mostra visualmente porque a melhor solução não foi alcançada através do *Conjunto 2*.

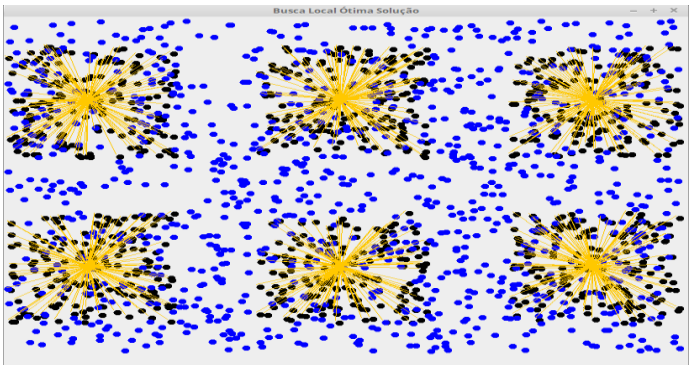

Figura 3. Solução ótima onde todas as facilidades utilizadas ficaram bem localizadas.

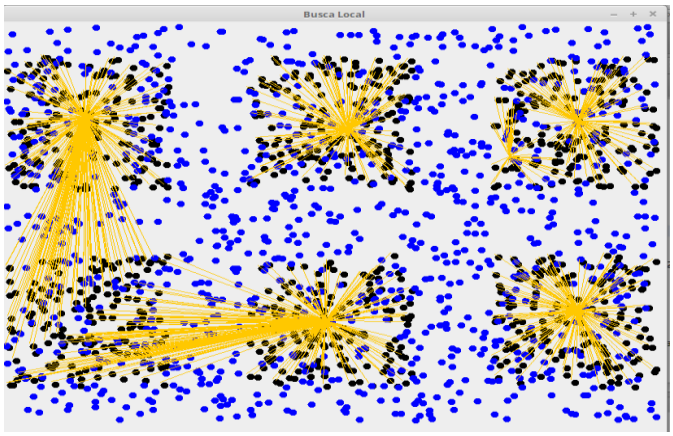

Figura 4. Solução não foi a melhor, pois uma demanda ficou muito próxima da outra (extremidade superior direita da imagem).

A Tabela 2 exibe informações das instâncias de problemas do *Conjunto 2*, onde foram computadas as seguintes informações: o valor da função objetivo *f(x)*; o número total de iterações realizadas com os vizinhos, *tot_i,* e o tempo computacional gasto em milissegundos, *t*.

Tabela 2. Informações das instâncias do *Conjunto 2*.

Nº	Prob. n_m_k	*f(x)*	***tot_i***	***t***
1	10_10_2	1820,26	8	32
2	20_20_4	1994,32	2	64
3	50_50_4	4058,07	10	368
4	100_100_6	8521,56	25	1128
5	200_200_6	15477,8	46	2328
6	300_300_6	23813,9	66	3528
7	400_400_6	30444,2	116	4728
8	500_500_6	38187,2	133	5928
9	700_700_4	53513,9	229	8328
10	1000_1000_6	76235,6	426	11928

A Tabela 3 mostra informações da instância de problema de nº 5 do Conjunto 1, para que seja possível visualizar passo a passo as melhorias ocorridas na função objetivo *f(x)* em determinada iteração *iter*, além do tempo computacional gasto *t*. Tais informações também estão sintetizadas no gráfico exibido na Figura 5.

Tabela 3. Melhoras obtidas com a busca local na instância de problema nº 5.

iter	f(x)	t	iter	f(x)	t
3	30962,3	7	624	21878,5	28
4	30776,4	7	630	21493,1	28
26	30284,7	9	648	21236,9	29
34	29764,6	10	659	21206,9	29
35	29657,3	10	777	19692,4	32
45	29648,8	11	789	18735,1	33
51	29105,7	12	814	18491,9	33
195	27982,9	16	855	18358,7	33
196	27235,7	16	976	17332,7	34
204	26130,6	16	980	17239,4	35
406	25521,3	22	981	16281,9	35
428	25514,8	23	998	16014,9	35
443	25069,8	23	1109	15985,4	36
471	24343,6	24	1115	15853,4	36
587	23476,2	27	1198	15850,7	36
598	23187,9	27	1370	15608,8	38
613	22447,2	28	1372	15477,8	38

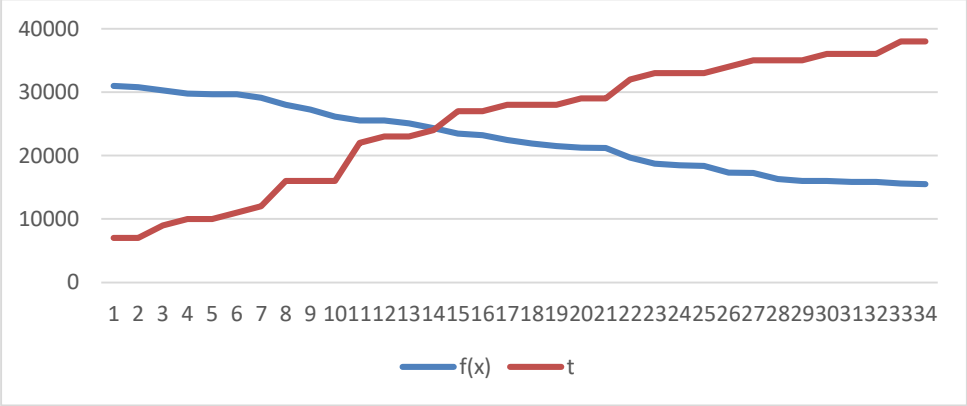

Figura 5. Evolução da função objetivo x, tempo computacional em milissegundos.

Para aprimorar a avaliação do desempenho da heurística Busca Local *All Pairs*, também foram executados testes com algumas das mesmas instâncias de problema com o Algoritmo Genético. Selecionou-se a instância de nº 7 do Conjunto 1, onde implementou-se o Algoritmo Genético com uma população de 700 indivíduos criados aleatoriamente, e como critério de parada utilizou-se o valor de 200 iterações sem que houvesse atualização da população.

Na Figura 6 pode-se observar que os resultados ficaram similares, porém a Busca Local teve como *f(x)* o valor de 37.506,89, sendo neste caso mais vantajoso que o Algoritmo Genético que resultou no valor final de 43.082,56.

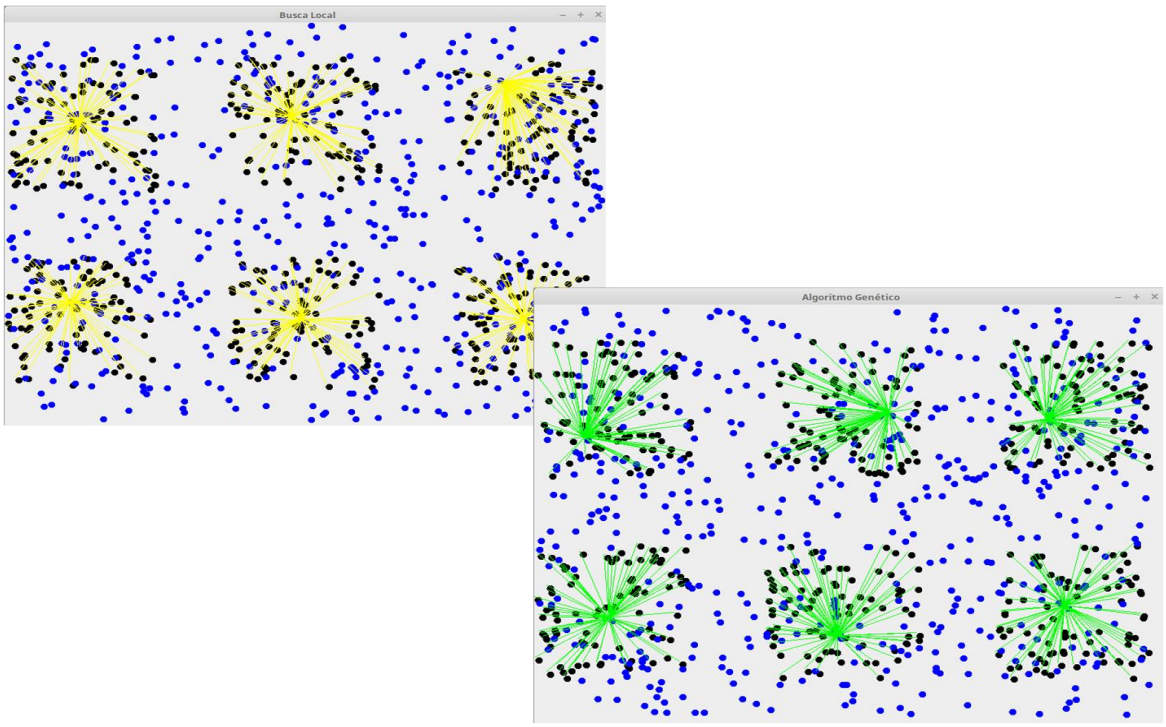

Figura 6. À esquerda, plotagem com Busca Local *All Pairs*, e à direita plotagem com Algoritmo Genético.

Também foram realizados testes na instância de problema nº 10 do conjunto 2, gerados de forma aleatória. A Busca Local *All Pairs* encontrou o valor de 113.618,93 para a função objetivo. Já o Algoritmo Genético teve um desempenho ligeiramente superior, obtendo o valor de 113.434,61.

5. Conclusões

Neste trabalho foi realizada uma análise da heurística de melhoria Busca Local, aplicando a permuta *All Pairs*, como solução do problema de alocação de facilidades. As instâncias de problemas foram divididas em dois conjuntos: o primeiro contendo pontos de demanda e facilidades estrategicamente posicionados, de forma a se conhecer a solução ótima, e o segundo onde estes eram posicionados aleatoriamente.

Conforme os dados demonstraram, há uma melhoria significativa quando a Busca Local é aplicada a um conjunto de dados maior, pois a diferença em porcentagem dos valores da solução fica menor conforme é aumentado o número de pontos de demanda e candidatos a facilidades. Apesar da Busca Local não obter sucesso para encontrar exatamente a melhor solução, os resultados são satisfatórios ao se considerar a proximidade que a mesma consegue alcançar. Ao comparar com outra técnica como o Algoritmo Genético, destacou-se que a eficiência da mesma teve vantagem quando os pontos de demanda foram gerados agrupados por blocos, começando a demonstrar desvantagem em um cenário totalmente aleatório.

Referências

Arya, V., Garga, N., Khandekar, R., Meyerson, K. M., Pandit, V. (2004). "Local Search Heuristics for K -Median and Facility Location Problems", Vol. 33, No 3, pp. 544-562.

Brondani, André Ebling. França, Francisca Andrea Macedo. Júnior, Roberto Velasco Kopp. Netto, Paulo Oswaldo Boaventura. Jurkiewicz, Samuel. (2013) "Alocação de unidades urbanas de lazer por um modelo de p-medianas." Revista Eletrônica Pesquisa Operacional para o Desenvolvimento, Rio de Janeiro, v. 5, n. 2, p. 209-223, mai./ago.

Capdeville, Rebeca Marcilio de Araújo, Vianna, Dalessandro Soares. (2013) "Heurísticas GRASP para o problema de alocação de pontos de acesso em uma rede sem fio em ambiente indoor". Revista Eletrônica Sistemas & Gestão, v. 8, n. 1, p. 86-93.

Hakimi, S. L. (1964) "Optimum distribution of switching centers and the medians of a graph. Operations Research", 12, 450-459.

Hakimi, S. L. (1965) "Optimum distribution of switching centers in a communication network and some related graph theoretic problems". Operations Research, 13, 462-475.

Lintzmayer, C. N., Mulati, M. H., Silva, A. F., (2011). "RT-ColorAnt: Um Algoritmo Heurístico Baseado em Colônia de Formigas Artificiais com Busca Local para Colorir Grafos", XLIII Simpósio Brasileiro de Pesquisa Operacional, pp. 1666-1677.

Lorena, L. A. N. Senne, E. L. Paiva, J. A. (2001) "Integração de modelos de localização a sistemas de informações geográficas. Gestão & Produção", São Carlos, v. 8, n. 2.

Lorena, L. Senne, E. (2003) "Local search heuristics for capacitated p-median problems, Networks and Spatial Economics". v.3, n.4, pp. 407-419.

Ribeiro, W. S. Arroyo, J. E. C. (2008) "Metaheurística GRASP biobjetivo para um problema de localização de facilidades". In Anais do Encontro Nacional de Engenharia de Produção, Rio de Janeiro.

Ribeiro, P. C. F. (2008) "Um enfoque na localização de facilidade baseado em testes de redução e heurísticas ADD/DROP". 2008. 37 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação), Faculdade Lourenço Filho, Fortaleza, Ceará, Brasil.

Pearl, P. (1984) "Heuristics: Intelligent Search for Computer Problem Solving." New York: AddisonWesley.