Análise de desempenho da heurística Busca Local com permuta *All Pairs* aplicada ao problema de alocação de facilidades

Tarcísio Barroso Marques[1], **Lucas de Souza Siqueira**[2], **Rodrigo Oliveira Zacarias** [3]

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. Em uma nova abordagem do problema, foi imposta uma restrição de distância que informa se a facilidade mais próxima ao ponto de demanda poderá atendê-lo ou não. O trabalho apresentado neste artigo poderá ser usado por outras Meta-heurísticas, como um Algoritmo Genético, através a criação de uma população inicial já melhorada pelo processo da Busca Local, dentre outros.

Palavras-chave: Busca Local, Alocação de Facilidades, Algoritmo Genético, P-Medianas.

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. In a new approach to the problem, a distance constraint has been imposed which informs whether the facility closest to the demand point can meet it or not. The project presented in this article can be used by other Meta-heuristics as a Genetic Algorithm, through the creation of an initial population already improved by the Local Search process, among others.

Keywords: Local Search, Allocation of facilities, Genetic Algorithm, P-Medians.

[1] Instituto Federal de Educação, Ciência e Tecnologia Fluminense - Campus Itaperuna, BR 356, Km 3 s/n, E-mail: tarcisio.marques@iff.edu.br
[2] Instituto Federal de Educação, Ciência e Tecnologia Fluminense - Campus Itaperuna, BR 356, Km 3 s/n, E-mail: lucassouza.ti@gmail.com
[3] Instituto Federal de Educação, Ciência e Tecnologia Fluminense - Campus Itaperuna, BR 356, Km 3 s/n, E-mail: rodrigo.oliveira.zacarias@gmail.com

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, a pesquisa apresentada neste artigo, que foi fomentada pelo Instituto Federal Fluminense Campus Itaperuna-RJ, aborda 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; 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; SENNE, 2003).

Para a resolução desses problemas relacionados à alocação de facilidades, heurísticas vêm sendo estudadas por diversos pesquisadores. Um exemplo dessas heurísticas foi utilizado 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, baseada no problema de p-medianas, projetada especificamente 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.

Com base no trabalho de Marques et al. (2017)[4], que trata o problema de p-medianas

[4] MARQUES, T. B.; SIQUEIRA, L. S.; ZACARIAS, R. O. Busca Local com permuta All Pairs aplicada ao problema de alocação de facilidades. In: Computer on the Beach, 2017, Florianópolis. Anais do Computer on the Beach, 2017. p. 446-455. Disponível em: https://siaiap32.univali.br/seer/index.php/acotb/article/view/10574/5929. Acesso em: 18 jan. 2018.

utilizando uma Busca Local com permuta *All Pairs*, este artigo apresenta uma nova abordagem considerando uma restrição de distância entre uma facilidade e um ponto de demanda, no intuito de ampliar a análise apresentada no referido trabalho. Nesse caso, o objetivo é constatar se uma facilidade mais próxima a um ponto de demanda poderá atendê-lo ou não.

Além disso, as duas abordagens do problema também são submetidas a um Algoritmo Genético para fins comparativos do nível de eficiência apresentado entre esse algoritmo e o de Busca Local *All Pairs*. Em todos os testes, 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: é a restrição de distância (máxima distância permitida de uma facilidade a um ponto de demanda). Por exemplo: alcance de uma antena; distância máxima permitida de um posto de saúde aos bairros que ele atende, etc.

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).

$N_i = \{ j \mid d_{ij} \leq d\}$ é o conjunto de facilidades que cobre ou atende o ponto de demanda *i*. (conjunto de facilidades que podem atender ao ponto *i*).

$Q = \{q_1, \ldots, q_z \mid n \geq z \in P\}$: conjunto de pontos de demanda que não foram atendidos devido à restrição de distância *d*.

K = Constante de normalização.

Formulação:

Minimiza $f(x) = \sum_{i=1}^{n} \min\{d_{ij} \mid j \in N\} + K.\sum_{i=1}^{z} q_j$ r

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. A segunda parte da função objetivo $K.\sum_{i=1}^{z} q_j$ procura maximizar o atendimento aos pontos de demanda. A distância euclidiana d_{ij} é calculada da seguinte forma:

$$\sqrt{|(x_i - x_j)| + |(y_i - y_j)|}$$

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 do que 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. Mas vale 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.

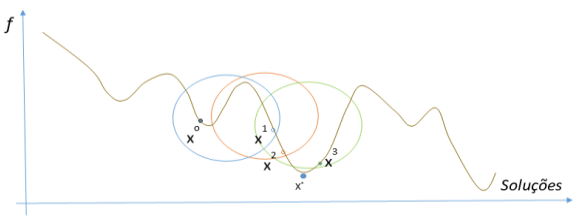

Figura 1 - Gráfico representativo da Busca Local.

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

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
```

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 ($f(x) \in N \mid f(x) \leq m$). O pseudocódigo é mostrado no Algoritmo 1.

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.

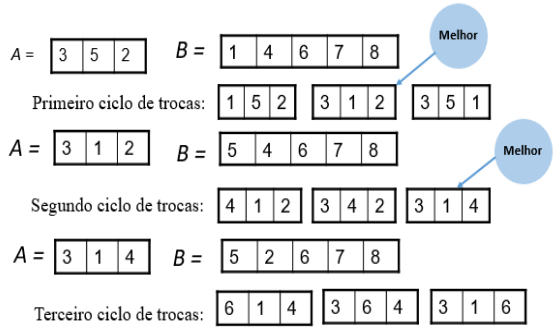

Figura 2 - Lógica do Algoritmo de Busca Local *All Pairs* aplicada ao problema de alocação de facilidades.

O pseudocódigo exibido no Algoritmo 2 retrata a Busca Local com a permuta All Pairs aplicada ao problema. Neste 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*. Esta 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){
01 G ← 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. Também foi estipulado uma distância máxima *d*, na qual consiste no limite em que o candidato não pode ultrapassar do ponto de demanda em que ele atenderia, sofrendo uma penalização *p* nesse caso.

Os pontos de demandas foram 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, notou-se uma diferença considerável no custo da solução, onde o valor desta encontrada pela Busca Local ultrapassou o valor da solução ótima em pelo menos 40% em todos os testes.

Tabela 1 - Custo das melhores soluções encontradas pelo algoritmo de Busca Local

Nº	*Prob. n_m_k*	*f*(x)*	*f(x)*
1	10_10_2	752,45	2161,47
2	20_20_4	1669,57	2644,97
3	50_50_4	3895,52	6371,55
4	100_100_6	7261,84	10283,17
5	200_200_6	15161,57	21300,36
6	300_300_6	32999,23	76033,46
7	400_400_6	30463,02	43708,80
8	500_500_6	37212,48	54686,39
9	700_700_4	52949,29	95373,13
10	1000_1000_6	76745,46	109692,85

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*.

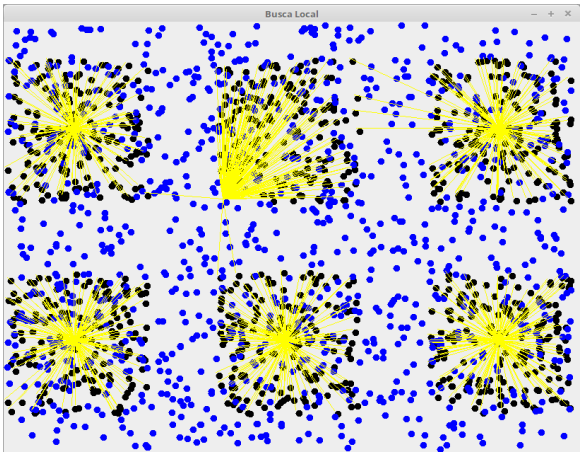

Figura 3 - Solução ótima onde todas as facilidades (pontos azuis) utilizadas ficaram bem localizadas, e todos pontos de demanda (pontos pretos) foram atendidos.

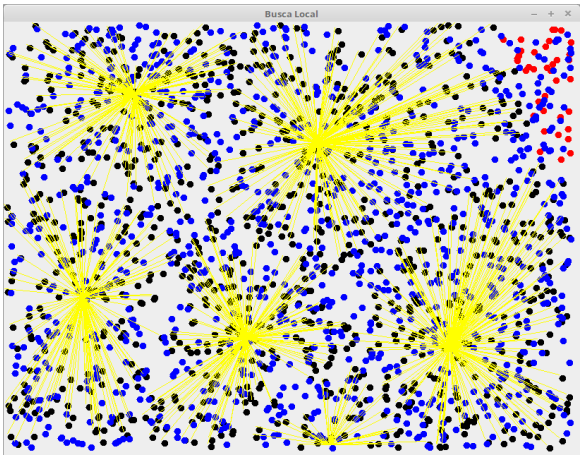

Figura 4 - Solução não foi a melhor, pois algumas demandas ficaram mal localizadas e houve pontos de demanda não atendidos na extremidade superior (pontos vermelhos).

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	2161,469684	32	6
2	20_20_4	2644,969829	64	10

3	50_50_4	6371,545944	368	59
4	100_100_6	10283,17094	1128	173
5	200_200_6	21300,35547	2328	456
6	300_300_6	32999,2296	3528	655
7	400_400_6	43708,79689	4728	942
8	500_500_6	54686,39257	5928	1076
9	700_700_4	95373,13133	5568	1024
10	1000_1000_6	109692,85	17892	2685

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***
4	69680,41	6
496	68063,5	150
497	61749,97	151
498	61358,58	151
728	61227,25	167
861	61209,5	186
990	56359,67	195
1096	56058,19	200
1105	56043,47	201
1487	50591,37	247
1538	50525,63	255
1550	50055,05	256
1863	50054,02	298
3461	38568,74	757
3462	38397,03	757
4452	37224,62	899
4881	37212,48	959

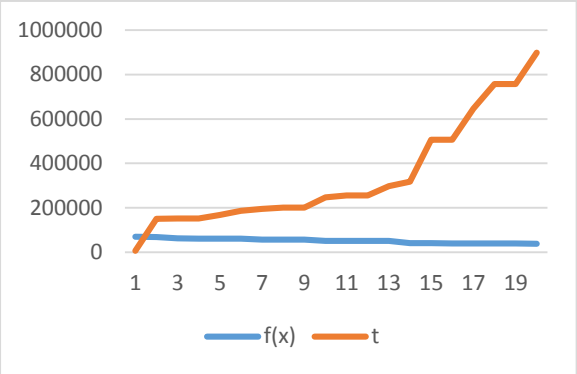

Figura 5 - Evolução da função objetivo x tempo computacional em microssegundos.

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 10 do Conjunto 1, onde implementou-se o Algoritmo Genético com uma população de 500 indivíduos criados aleatoriamente, e como critério de parada utilizou-se o valor de 100 iterações sem que houvesse melhorias nessa população.

O valor-limite para a distância do ponto de demanda e facilidade foi definido em 300, acrescentando uma penalidade de 300 no custo da solução, para cada ponto que ultrapassasse a distância máxima. Tal critério foi utilizado para ambos testes.

Nas Figuras Figura ***6*** e Figura ***7*** pode-se comparar o resultado do Algoritmo Genético e da Busca Local, onde a mesma demonstrou-se mais vantajosa com o *f*(x)* o valor de 76.745,46, enquanto no Algoritmo Genético encontrou-se o valor de 84.176,90.

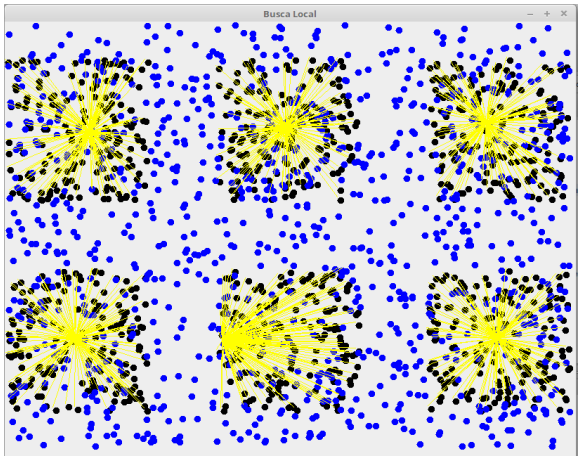

Figura 6 - Plotagem com Busca Local *All Pairs*, com pontos de demanda gerados estrategicamente alocados.

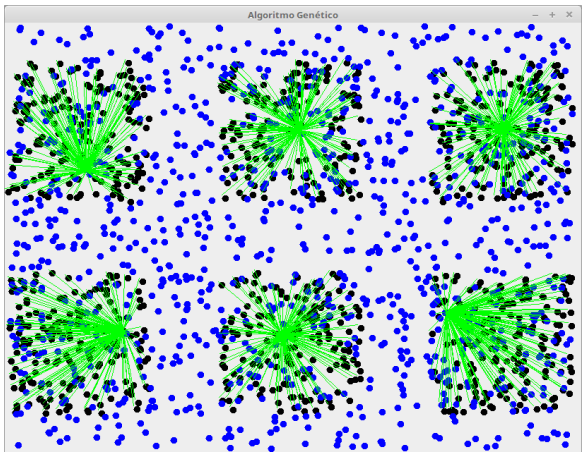

Figura 7 - Plotagem com Algoritmo Genético aplicado ao Conjunto 1.

As Figuras Figura ***8*** e Figura ***9*** exibem visualmente as soluções encontradas pela Busca Local e Algoritmo Genético para a instância de problema nº 10 do Conjunto 2. A Busca Local *All Pairs* encontrou o valor de 109.692,85 para a função objetivo *f(x)*. Já o Algoritmo Genético também teve um desempenho inferior neste caso, obtendo o valor de 116.661,42.

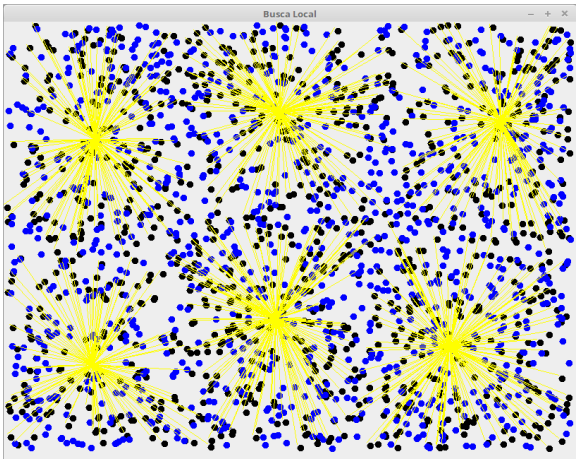

Figura 8 - Plotagem com Busca Local *All Pairs* aplicado ao Conjunto 2 na instância de nº 10.

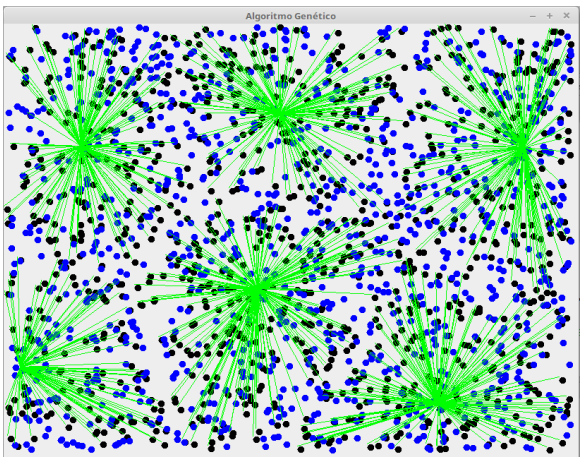

Figura 9 - Algoritmo Genético aplicado à instância de nº 10 no Conjunto 2.

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. Além disso, afim de aumentar o grau de complexidade do problema, foi implementada uma restrição de distância, na qual o custo era penalizado por cada ponto de demanda que ultrapasse o limite de distância de sua facilidade.

Conforme os dados demonstraram, houve uma diferença acentuada quando a Busca Local foi aplicada a um conjunto de dados estrategicamente posicionados. As mesmas instâncias de problemas sem a restrição de distância máxima, não divergiram na mesma proporção, uma vez que a tendência da diferença do custo do mesmo cenário aplicado aos conjuntos era se aproximar. Ao se comparar com o Algoritmo Genético, notou-se um desempenho ligeiramente melhor da Busca Local nos dois conjuntos, demonstrando que para este tipo de problema o seu uso foi eficiente.

Referências

ARYA, V.; GARGA, N.; KHANDEKAR, R., MEYERSON, K. M.; PANDIT, V. Local Search Heuristics for K-Median and Facility Location Problems, Vol. 33, No 3, pp. 544-562, 2004.

BRONDANI, A. E.; FRANÇA, F. A. M.; KOPP JÚNIOR, R. V.; NETTO, P. O. B.; JURKIEWICZ, S. 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, 2013.

CAPDEVILLE, R. M. A.; VIANNA, D. S. 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, 2013.

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

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

LINTZMAYER, C. N.; MULATI, M. H.; SILVA, A. F. 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, 2011.

LORENA, L. A. N; SENNE, E. L; PAIVA, J. A. Integração de modelos de localização a sistemas de informações geográficas. Gestão & Produção, São Carlos, v. 8, n. 2, 2001.

LORENA, L; SENNE, E. Local search heuristics for capacitated p-median problems, Networks and Spatial Economics. v.3, n.4, pp. 407-419, 2003.

MARQUES, T. B.; SIQUEIRA, L. S.; ZACARIAS, R. O. Busca Local com permuta All Pairs aplicada ao problema de alocação de facilidades. In: Computer on the Beach, 2017, Florianópolis. *Anais do Computer on the Beach, 2017*. p. 446-455. Disponível em: https://siaiap32.univali.br/seer/index.php/acotb/article/view/10574/5929. Acesso em: 18 jan. 2018.

RIBEIRO, W. S.; ARROYO, J. E. C. Metaheurística GRASP biobjetivo para um problema de localização de facilidades. *Anais do Encontro Nacional de Engenharia de Produção*, Rio de Janeiro, 2008.

RIBEIRO, P. C. F. Um enfoque na localização de facilidade baseado em testes de redução e heurísticas *ADD/DROP*. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação), Faculdade Lourenço Filho, Fortaleza/CE, 2008.

PEARL, P. *Heuristics: Intelligent Search for Computer Problem Solving*. New York: AddisonWesley, 1984.