Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 9645-9657 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

JUMPING FROG OPTIMIZATION E ALGORITMO GENÉTICO APLICADOS À SOLUÇÃO DO PROBLEMA DAS P-MEDIANAS

Anderson M. de Vasconcelos, Sérgio R. de Souza, João F. de A. Vitor, Sinaide N. Bezerra and Cynthia da S. Barbosa

Centro Federal de Educação Tecnológica de Minas Gerais, Av. Amazonas, 7675, 30510-000, Belo Horizonte - MG - Brasil, andersonmv@gmail.com,sergio@dppg.cefetmg.br

Keywords: Jump Frog Optimization; Algoritmo Genético; p-Medianas.

Abstract. Este artigo propõe um estudo comparativo entre aplicações das metaheurísticas *Jump Frog Optimization* e *Algoritmo Genético Híbrido* aplicados à solução do Problema das p-Medianas. O problema das p-medianas tem por objetivo determinar *p* nós, denominado medianas, em um grafo de *n* vértices, minimizando a distância total a partir de outros nós do grafo. A metodologia utilizada consiste na implementação do Algoritmo de Otimização por Saltos de Rãs (JFO) e Algoritmo Genético com a Busca Local (AG-BL) para melhoria do cromossomo gerado. A técnica utilizada para Busca Local é Descida Randômica. Após conhecidos os pontos onde serão fixados as medianas, é aplicado o algoritmo de Gillet e Johnson (G&J) para a geração das regiões de atendimento, conhecida como $cluster$, juntamente com a Heurística de Localização-Alocação (HLA). HLA - são métodos utilizados para melhorar a localização da mediana na sua região de atendimento. Os resultados obtidos são analisados e comparados com os encontrados na literatura e mostram que as soluções encontradas através da metaheurística JFO são superiores às encontradas pelo AG.

1 INTRODUÇÃO

Otimização por Enxame de Partículas (*Particle Swarm Optimization - PSO*) é uma meta-heurística proposta inicialmente por Kennedy and Eberhart (1995). Foi inspirada no comportamento cooperativo de vários tipos de animais: revoadas de pássaros, enxames de abelhas e cardumes de peixes, os quais são denominados de partículas. Para encontrar alimentos, eles utilizam suas próprias experiências e as experiências do bando.

O PSO desenvolve-se a partir de um conjunto de partículas $P(t)$, geradas randomicamente, as quais são candidatas à solução do problema de otimização. Cada partícula é tratada como um ponto no espaço N-dimensional. Assim, segundo Shi and Eberhart (1998), temos que:

- $X_i = (x_{i1}, x_{i2}, \ldots, x_{iN})$: partícula i, candidata à solução do problema, na qual x_i representa uma variável da solução;
- $P_i = (p_{i1}, p_{i2}, \ldots, p_{iN})$: melhor posição prévia de cada partícula (posição que possui o melhor valor de aptidão para a solução do problema), também chamada de $pbest_i$. A melhor partícula prévia dentre todas as partículas da população é representada pelo símbolo g e mostra a melhor posição ocupada pelas partículas do bando;
- $V_i = (v_{i1}, v_{i2}, \ldots, v_{iN})$: velocidade ou taxa de variação da posição da partícula i.

Algoritmo PSO
Iniciar randomicamente a posição e a velocidade das N partículas da população;
Faça
Avaliar indivíduos através do cálculo da função objetivo: *f(x)*;
Atualizar $pbest_i$;
Se $f(x_i) < f(pbest_i)$ então
$pbest_i = x_i$;
Atualizar $gbest$;
Se $pbest_i < f(gbest)$ então
$gbest = pbest_i$;
Atualizar a velocidade de cada partícula;
Atualizar a posição de cada partícula;
Enquanto (critério de parada não atingido)
Fim_Algoritmo.

Figure 1: Estrutura básica do algoritmo PSO.

1.1 Algoritmo JFO (Jump Frog Optimization)

Uma nova proposta de adaptação do método PSO para problemas de otimização discretos é apresentada em García and Pérez (2007). Esta versão, denominada Otimização por Saltos de Rãs (*Jump Frog Optimization* - JFO), tem sua inspiração no movimento de um grupo de rãs saltando de pedra em pedra em um certo intervalo de tempo.

No algoritmo JFO, um conjunto de partículas se locomove no espaço de busca discreto, saltando de uma solução a outra, sem ocupar posições intermediárias. Como as partículas podem ocupar apenas um único conjunto de posições, definido pelo domínio do problema, a idéia

de velocidade (adotada na versão contínua do algoritmo PSO) é substituída por saltos esporádicos e aleatórios no espaço de busca.

O algoritmo JFO mantém as características básicas do algoritmo PSO contínuo desenvolvido por Kennedy and Eberhart (1995), guardando informações sobre suas melhores posições prévias, individuais e coletivas. A mudança proposta pelo algoritmo JFO encontra-se na atualização da posição de cada partícula. Neste procedimento, a velocidade não é considerada, logo, a matriz de velocidade V não existe. As matrizes X, B e G se mantêm conforme a versão original do algoritmo PSO.

Esse método consiste em realizar movimentos aleatórios, com probabilidade c_1; movimentos de aproximação da melhor posição prévia da partícula (b_i), com probabilidade c_2; movimentos de aproximação da melhor posição na vizinhança da partícula (g_i), com probabilidade c_3; e movimentos de aproximação da melhor posição global (g^*), com probabilidade c_4. conforme a equação (1).

$$x_{i+1} = c_1 x_i \oplus c_2 b_i \oplus c_3 g_i \oplus c_4 g^* \tag{1}$$

A figura 2 apresenta a soma vetorial, conforme equação (1). Nesta figura, s indica a posição do valor ótimo para o problema.

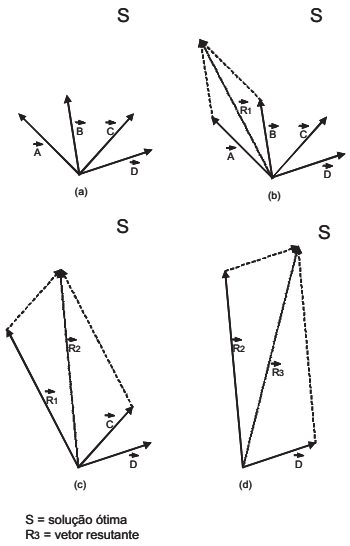

Figure 2: Comportamento do algoritmo JFO: Soma vetorial ilustrativa.

O comportamento deste algoritmo torna-se mais claro a partir do exemplo a seguir. Seja a figura 2, que representa a soma dos vetores $\vec{A}$, $\vec{B}$, $\vec{C}$ e $\vec{D}$, dada pelas equações (2) e (3).

$$\begin{aligned} \vec{A} &= c_1 x_i; \qquad \vec{B} = c_2 b_i; \\ \vec{C} &= c_3 g_i; \qquad \vec{D} = c_4 g^*. \end{aligned} \tag{2}$$

O vetor resultante $\vec{R_3}$ é dado por:

$$\vec{R_3} = \vec{A} + \vec{B} + \vec{C} + \vec{D} \tag{3}$$

Estes movimentos permitem substituir uma solução por outra dentro do espaço de busca. Assim, a atualização da posição de cada partícula é efetuada conforme o procedimento a seguir: Considere um conjunto composto pelas posições das partículas, x_i, com $i = 1, 2, \ldots, t$, sendo t

o tamanho do bando, e um intervalo $U \in [0, 1]$, composto pelas probabilidades de movimentos c_j, com $j = 1, 2, 3, 4$. A relação entre estes valores é mostrada pela figura 3.

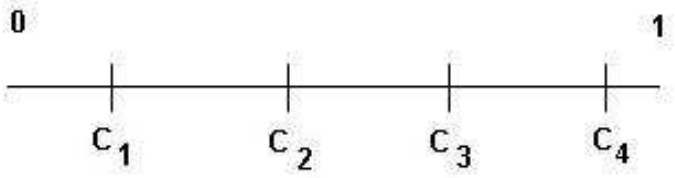

Figure 3: Segmentos c_1, c_2, c_3 e c_4.

Cada partícula x é composta por k atributos. O tamanho da partícula corresponde à quantidade de facilidades procuradas; os valores dos atributos são distintos e variam segundo $1, 2, \ldots, n$, sendo n correspondente ao maior valor do domínio discreto. A estrutura de movimentos do algoritmo JFO é descrita pela figura 4. Nesta figura, Mov é a probabilidade de acontecer um movimento de troca correspondente ao atrator p_i, e Movimento() é a função de troca do elemento atrator em p_i por um outro elemento escolhido randomicamente.

Algoritmo Movimentos_JFO
Passo 1: Eleger aleatoriamente um atributo p_i da partícula x_i,
sendo p_i denominado atrator da partícula x_i;
Passo 2: Dividir o intervalo U em quatro intervalos com probabilidades
c_1, c_2, c_3 e c_4, sendo $c_4 = 1 - (c_1 + c_2 + c_3)$;
Passo 3: Gerar um número aleatório $rand \in [0\ \ 1]$ e verificar o
segmento ao qual ele pertence dentro do intervalo U, indicando
a probabilidade de movimento c_j associada;
Passo 4: Gerar outro número aleatório $Rand \in [0\ \ 1]$.
Passo 5: Aplicar movimentos, a cada iteração:
$Mov = Rand * \ p_i * \ c_j$
Se ($Mov > 1$) então aplicar Movimento()
Passo 6: Atualizar a posição das partículas:
Se $Mov > 1$, então p_i é atualizado com uma posição
aleatória $X - \{p_i \cup x_i\}$;
Senão terminar iteração.
Fim_Algoritmo.

Figure 4: Algoritmo de atualização das partículas para JFO.

Assim, o algoritmo efetua várias trocas e movimentos no espaço de busca, promovendo uma varredura intensa nas soluções possíveis. O algoritmo JFO encontra-se na figura 5.

1.2 Algoritmo Genético - AG

Algoritmo genético (AG), descrito no trabalho de Holland (1975), é uma metaheurística pertencente a uma família de modelos computacionais inspirados na evolução natural das espécies. Semelhante ao PSO, o AG é uma metaheurística populacional, inicializada com uma população gerada aleatoriamente, composta por candidatos a soluções do problema.

Segundo Zuben and Castro (2004), o AG representa uma das técnicas mais empregadas dentre as componentes da Computação Evolutiva. Goldberg (1989) afirma que o AG é um algoritmo

Algoritmo JFO
Iniciar randomicamente a posição das *N* partículas da população;
Iniciar X, B *e* G;
Faça
Atualizar X;
Para i $\leftarrow$ 1 até *N* faça
Avaliar as aptidões das partículas em X através de $f(X_i)$;
Atualizar B, segundo algoritmo Movimentos_JFO
Se $f(X_i) < f(B_i)$, então $f(B_i) = X_i$;
Atualizar G. Se $B_i < f(G)$, então *G* = B_i;
Fim_Para;
Enquanto (critério de parada não atingido)
Fim_Algoritmo.

Figure 5: Algoritmo JFO.

robusto, eficiente e eficaz para vários tipos de problemas.

Os termos empregados pela metaheurística Algoritmo Genético podem ser definidos por:

- P(t) = $(x_1^t, x_2^t, \ldots, x_n^t)$: população, conjunto de indivíduos (cromossomos) pertencentes ao espaço de busca;
- $x_i^t = (g_1^t, g_2^t, \ldots, g_n^t)$: cromossomo candidato à solução do problema de otimização;
- g_j^t: gene, representa uma variável da solução;
- Alelo: refere-se ao conjunto de valores possíveis que um gene pode assumir;
- Aptidão (*fitness*): mede a adequação de um cromossomo para solução do problema tratado (geralmente associada ao valor da função objetivo específica do problema).

A figura 6 apresenta a estrutura básica de um Algoritmo Genético.

Algoritmo (AG)
Inicie a população;
Avalie a população
Enquanto (critério de parada não for atingido) faça
Selecione indivíduos para próxima população;
Aplique cruzamento e mutação;
Avalie a população;
Fim_Enquanto;
Fim_Algoritmo.

Figure 6: Pseudocódigo Algoritmo AG Básico.

A população inicial de indivíduos pode ser gerada de várias maneiras. Na maioria das vezes, é realizada de forma aleatória. Segundo Grefenstette (1986), o tamanho da população afeta o desempenho global e a eficiência do algoritmo, e varia com a classe de problema em estudo.

A avaliação consiste em verificar o valor da aptidão de cada indivíduo da população. É através do cálculo da aptidão que se mede o quanto um indivíduo está próximo da solução desejada e qual a probabilidade dele vir a reproduzir.

Após a avaliação de todos os indivíduos da população, tem-se o processo da seleção de indivíduos para reprodução. Terminado a seleção, novos indivíduos são criados a partir dos operadores genéticos de cruzamento e mutação.

Neste trabalho é utilizado o operador de cruzamento chamado de $OrderCrossover$ (OX). Este operador gera seus descendentes. Selecionando uma subsequência de um caminho de um pai, preservando a ordem relativa da sequência do outro pai.

Terminada a recombinação por cruzamento, os cromossomos filhos podem sofrer mutação, o que irá introduzir novos indivíduos na população, contribuindo para a manutenção de sua diversidade. A mutação altera arbitrariamente um ou mais genes, resultando em um novo cromossomo.

Quando acontece o cruzamento, pode ocorrer inconsistência em algum dos cromossomos. Genes repetidos podem ser atribuídos a um mesmo indivíduo. Sejam r_i={2,4,15,6,21} e r_j={8, 12, 2, 5, 19}, r_i e $r_j \subset V$. Aplicado após o cruzamento, tem-se os descendentes r_x={2, 12, 2, 5, 21} e r_j={8, 4, 15, 6, 19}. Os genes 1 e 3 em r_x são iguais, logo, r_x não é viável, e assim, não é uma solução candidata para o PMP.

Quando um cromossomo C não é viável, o mesmo não pode ser considerado como um descendente válido. Para solucionar este problema, deve-se verificar os genes repetidos em C e obter aleatoriamente em V um gem que torne C viável, e $C \subset V$. Aplicando este procedimento no cromossomo de exemplo r_x, temos: r_x = {11, 12, 2, 5 21}. O primeiro gene de r_x foi escolhido de forma aleatória em V, produzindo assim um cromossomo viável. No algoritmo apresentado, a viabilidade dos cromossomos descendentes, r_x e r_y, é verificada após cada cruzamento.

No algoritmo apresentado na fig. 7 no $passo5$, r_x ou r_y é fixada em 50% para cada crosssomo. Após a escolha de um cromossomo, a mutação ocorre mediante uma probabilidade denominada probabilidde de mutação. Esta probabilidade é um dos parâmetros que podem influenciar no funcionameto do AG.

Neste trabalho, é apresentado um Algoritmo Genético com Busca Local, um método iterativo proposto inicialmente por Feo and Resende (1995). Ele consiste, basicamente, na construção de uma solução inicial aleatória em que são gerados elemento a elemento, e outra fase de busca local, que consiste em efetuar trocas dos genes em um cromossomo, com o objetivo de melhorar sua aptidão. Em seguida, é retornada a melhor solução encontrada.

2 PROBLEMA DE LOCALIZAÇÃO

O problema das p-Medianas faz parte do conjunto dos problemas de localização, importante linha de pesquisa de Otimização Combinatória. Tais problemas buscam determinar pontos estratégicos onde serão alocadas as medianas, de forma a atender, da melhor maneira possível, um conjunto espacialmente distribuído.

O objetivo deste trabalho é testar técnicas capazes de solucionar tais problemas de maneira mais eficiente e adequada.

Senne and Lorena (2003) propõem um modelo de programação inteira binária para a solução

Algoritmo AGPMP

Passo1 - Inicialização;

construir a população inicial, gerar uma lista $R = \{r_1, r_2, ...r_m\}$;
com m cromossomos viáveis de p genes cada, escolhidos aleatoriamete em V.
Calcular C_1 = Aptidão (r_i)
Ordenar *R* de modo que $C_1 \leq C_2 \leq ... \leq C_m$;
defina *k*=0, it_{max} e it_{sm}

Passo 2 - Teste

Testar se *k* $\geq it_{max}$ ou *k* $\geq it_{sm}$ então PARE e apresente o cromossomo r_1

Passo 3 - Seleção;

r_i = Seleção(R) e r_j = Seleção(R), $\forall\ r_i \neq r_j$;

Seleção(R) = $\left\{ r_j \in \Re \ / \ j = \left[\frac{-1+\sqrt{1+4.rnd(n^2+n)}}{2} \right] \right\}$

Passo 4 - Cruzamento;

fazer o cruzamento$(ri, rj) = (rx, ry)$;

Passo 5 - Mutação;

Escolher r_x ou r_y;
Aplicar mutação no cromossomo escolhido;

Passo 6 - Aptidão;

Calcular a aptidão de r_x e r_y;
Se aptidão(r_x) $<$ aptidão(r_y)então s = aptidão(r_x);
Se aptidão(s) $< C_m$ então
$C_m =$ aptidão(s);
R_m = (s);
Senão
Busca_Local(s);

Passo 7 - Ordenação;

Ordenar R, mantendo a ordem crescente de aptidões;
Fazer k=k+1 e voltar ao Passo 2;

Fim Algoritmo.

Figure 7: Pseudocódigo do Algoritmo Genético para o PMP *(p-Median Problem)*

do problema da p-Mediana como demonstrado a seguir:

$$\text{Q(z)=Min} \quad \sum_{i \in N} \sum_{j \in N} d_{ij} x_{ij} \tag{4}$$

$$\text{S.a:} \quad \sum_{i \in N} x_{ij} = 1; j \in N \tag{5}$$

$$\sum_{i \in N} x_{ii} = p \tag{6}$$

$$x_{ij} \leq x_{ii}; i, j \in N \tag{7}$$

$$x_{ij} \in \{0, 1\}; i, j \in N \tag{8}$$

em que:

- N=$\{1, ..., n\}$, cada elemento de *N* representa um vértice
- $[d_{ij}]_{nxn}$ é uma matriz simétrica de custos (ou distância), com $d_{ii} = 0, \forall i \in N$;
- $[x_{ij}]_{nxn}$ é uma matriz de alocação, com $x_{ij} = 1$ se o nó j é alocado à mediana i, e $x_{ii} = 0$, caso contrário;
- p é o número de medias;
- n é o número de nós.

As restrições (5) e (7) garantem que cada nó j é alocado a apenas um nó i, o qual deve ser uma mediana. A restrição (6) determina o número exato de medianas a ser localizado e a restrição (8) corresponde às condições de integridade.

Segundo Senne and Lorena (2003), esta formulação do problema das p-medianas é NP-difícil. A formulação de *4* a *7* pode ser resolvido apenas para instâncias de pequeno porte. Essas limitações existem devido ao aumento exponencial do tempo de resolução à medida que aumentam os dados de entrada. Segundo Woeginger (2003), é possível projetar algoritmos que sejam significativamente mais rápidos do que a busca exaustiva. Mesmo assim, esses algoritmos continuam sendo não polinomiais, fato que justifica o uso de metaheurísticas neste trabalho.

Encontramos aplicações de metaheurísticas para a solução das p-medianas no trabalho de Mladenovic et al. (2007) que, examinam o problema das p-medianas, com o objetivo de proporcionar uma visão geral sobre seus avanços e como resolvê-lo. Steiner et al. (2004), apresentam uma metaheurística aplicada ao problema das p-Medianas. Além de utilizarem os operadores convencionais, utilizam também um novo operador chamado de hipermutação heurística. Também comparam o AG com outro algoritmo de busca tabu. Osman and Erhan (2003) propõem um novo algoritmo genético para um problema de localização. O algoritmo é relativamente simples e gera boas soluções rapidamente. A evolução é facilitada por uma heurística gulosa.

O PMP é representado através de um grafo composto por um conjunto de vértices *V*, Cada vértice é visto como um potencial local para se instalar uma facilidade ou simplesmente medianas. Seja, então, um grafo não direcionado $G(V, A)$, $|V| = n$, onde V são os vértices e A, as arestas. O PMP consiste em determinar um conjunto com p vértices, formando-se um conjunto V_p, sendo $V_p \subset V$ tal que $|V_p| = p$, logo V_p é a solução para o problema das p-medianas, agregando o máximo possível de arcos com a menor distância entre os pontos. Logo, V_p contém os elementos (vértices) que indicam a localização de cada facilidade.

3 GERAÇÃO DE CLUSTERS

O método empregado na determinação da região atendida por cada uma das facilidade $(cluster)$ é implementado segundo o algoritmo de designação de Gillet e Johnson, conforme Smiderle et al. (2004). A sua execução é repetida até que todos os nós estejam relacionados a uma determinada mediana.

Após terminado o algoritmo de determinação, é aplicada a heurística de Localização (HLA), proposta em Lorena (2001), que objetiva melhorar, pelo uso da busca local, a distância dentro de um $cluster$ já formado, alterando a mediana com um vértice a ela designado. A HLA é empregada no trabalho de Arakaki and Lorena (2006). A $Fig.9$ apresenta o pseudo-código descrito em Arakaki and Lorena (2006) e adaptado neste trabalho para o PMP não capacitado.

Em alguns casos, a busca por localizar as facilidades, não faz uso a geração da região de atendimento Beasley (1990). Neste caso, os algoritmos G & J e HLA, não serão executados.

Algoritmo Gillet & Johnson
Passo 1 - Calcula-se a distância entre cada nó ainda não designado até cada uma das medianas;
Passo 2 - Para cada nó *i* (não designado) do passo anterior, obter t_i^1 como sendo a mediana mais próxima de *i* e t_i^2 a segunda mediana mais próxima de *i*, com distâncias c_i^1 e c_i^2, respectivamente;
Passo 3 - Para todos os nós i, calcular a razão $r_i = c_i^1/c_i^2$.
Ordenar os nós i de acordo com os valores de r_i, em ordem crescente;
Passo 4 - Designar os nós *i* para as medianas mais próximas, até que sua capacidade de atendimento se esgote.
Voltar ao passo 1 até que todos os nós estejam designados.
O algoritmo será executado até que todos os nós estejam designados a uma determinada mediana.

Figure 8: Pseudocódigo para determinação de *cluster* de Gillet & Johnson.

Algoritmo HLA
Dados *J* conjuntos das medianas = $\{j_1, \ldots, j_p\}$
$C_k$conjunto de vértices do agrupamento k = $\{v_1,\ldots, v_{|Ck|}\}$
μ_{kj} soma das distâncias da mediana j aos vértices do agrupamento k
$|C_k|$ cardinalidade de C_k
Enquanto (solução-inicial melhora) faça
Para C_k = 1, ..., p faça
Para i = 1, ..., $|C_k|$ faça
Troque mediana *jk* por um vértice v_i;
Calcule novo μ_{kj} ;
Se novo μ_{kj} for melhor que antigo μ_{kj} então
Atualiza μ_{kj};
Guarda nova mediana;
Fim_Se;
Fim_Para;
Fim_Para;
Atualiza *J*;
Calcula o valor *novasol* que corresponde as realocações;
Se *novasol* for melhor do que solução-inicial então
Faça solução-inicial $\leftarrow novasol$;
Fim_Se;
Fim_Enquanto;
Fim_Algoritmo.

Figure 9: Pseudocódigo da HLA.

4 RESULTADOS COMPUTACIONAIS E ANÁLISE

Para a verificação da eficácia do AG-BL e o JFO foram utilizados as instâncias descritas a seguir:

- Instância 1 - Grupo de instâncias descritas em Beasley (1990) para o PMP não capacitado.

- Instância 2 - Grupo de instâncias descritas em Lorena (2001) com 324 e 818 nós para o PMP não capacitado.

- Instância 3 - Grupo de instâncias descritas em Lorena (1999) implementado para métodos exatos com 3282 nós e testado neste trabalho para até 1000 medianas.

A Tabela 1, apresenta os resultados produzidos pelas rotinas AG, AG-BL e JFO para Instância 1 sem a geração de $clusters$, com 100, 200, 400, 600 e 900 nós e 5, 67, 133, 200 e 90 medianas respectivamente. Para o AG foram fixados os seguintes parâmetros: Probabilidade de mutação = 5% , iteração máxima (it_max) = 1000, iteração sem melhora (it_sm) = 50 e percentual de busca $(perc\_busca)$= 80%. Os resultados da metaheurística AG, foram extraídas de Bezerra (2008).

O algoritmo foi implementado em linguagem de programação C++ Builder-6, e executado 10 vezes para cada mediana em um computador Intel Core 2 Duo com 3 GB de memória RAM e sistema operacional Windows XP.

Os resultados obtidos para a Instância 1 são apresentados na Tabela 1. Nesta tabela, tem-se que: n, quantidade total de pontos; p, quantidade de medianas; s, solução encontrada por Lorena (2001); alg, algoritmo; σ, melhor solução encontrada; m_σ, solução média; t_m, tempo médio em segundos das soluções.

Table 1: Tabela para grupo de instâncias 1, sem geração de $cluster$

Instâncias	n	p	s	alg	s	ms	tm(s)
pmed1	100	5	5819	AG	5893	6175	2,43
				AG_BL	5819	5823	2,95
				JFO	5819	5820	2,67
pmed10	200	67	1255	AG	1474	1567	15,96
				AG_BL	1267	2161	161,1
				JFO	1258	1265	2,75
pmed20	400	133	1789	AG	2337	2433	26,18
				AG_BL	1831	1819	4231
				JFO	1825	1832	348,56
pmed30	600	200	1989	AG	2686	2744	41,28
				AG_BL	2304	2044	21378
				JFO	2047	2057	161,9
pmed40	900	90	5128	AG	6017	6141	26,03
				AG_BL	5246	5297,9	3746
				JFO	5240	5823	156

A combinação dos métodos AG e busca local AG-BL permite sair de ótimos locais mais facilmente. Quando o AG converge para um ótimo local, e dele não consegue escapar através do cruzamento ou mutação, é auxiliado pela busca local, que, nesta situação diversifica os cromossomos dos indivíduos da população, alterando de forma contundente a combinação de genes dos cromossomos.

Nota-se porém que os resultados com o algoritmo JFO são melhores em comparação ao AG.

A Tabela 2, apresenta os resultados entre AG, AG-BL e JFO para Instância 2, com 324 nós, com 5, 10, 20 e 50 medianas. Os parâmetros de probabilidade de mutação, (it_max), (it_sm) e $(perc\_busca)$ são os mesmos utilizados para Instancia 1.

Nos testes com a Instância 2 os resultados são melhores com JFO e AG-BL. Os algoritmos para geração dos clusters tornam-se métodos de refinamento da solução encontrada, principalmente a HLA, que procura melhor a solução a partir de uma solução já encontrada.

Mesmo com o emprego dos algoritmos $G\&J$ e HLA no refinamento das soluções abtidas com o AG, essas técnicas não obtiveram sucesso nas isntâncias estudadas. Por outro lado, o

Table 2: Tabela para grupo de instâncias 2

Instâncias	n	p	s	alg	s	ms	tm(s)
pmedian324	324	5	122518	AG	127920	128554	16,22
				AG-BL	127078	129894	371,34
				JFO	126721	128556	41,44
		10	79256,36	AG	828553	85610	14,5
				AG-BL	82447	800976	681
				JFO	80463	80795	26,27
		20	54533,11	AG	60109	61954	15,68
				AG-BL	55573	54876	3104,43
				JFO	55428	57967	581
		50	32101,52	AG	38657	39793	19,6
				AG-BL	33496	32764	7125
				JFO	33396	33549	1756

AG-BL refinado pelas algoritmos $G\&J$ e HLA, mostram melhores soluções em relação ao AG.

Para as Instância 2 com 818 nós $(tabela3)$, os resultados os resultados são apresentados para os algortimos AG, AG-BL e JFO.

A Tabela 4, apresenta os resultados entre AG, AG-BL e JFO para Instância 3, com 3282 nós, com 5, 10, 20, 50, 100 e 500 medianas.

Table 3: Tabela para grupo de instâncias 2

Instâncias	n	p	s	alg	s	ms	tm(s)
pmedian818	818	5	605856,1	AG	637874	647937	18,3
				AG-BL	628739	635784	15353
				JFO	619745	627694	8294
		10	251717,8	AG	310971	329524	23,66
				AG-BL	306954	326465	14941
				JFO	288167	637594	212,55

Table 4: Tabela para grupo de instâncias 3

Instâncias	n	p	s	alg	s	ms	tm(s)
pmedian3282	3282	5	6381119	AG	6668893,7	6617009	23812,02
				AG-BL	6568893,2	6637284,9	18647,9
				JFO	6494538,6	6528367,4	1240,85
		10	3914249,8	AG	5413822,5	5368845,5	41433,04
				AG-BL	4724048,0	4932412,3	37324,77
				JFO	4593223,5	4637101,8	3250,93
		20	2350502,5	AG	3231004,3	3268941,4	47625,19
				AG-BL	2849826,9	2894786,2	43562,78
				JFO	2704734,5	2787853,4	9586,25
		50	1308957,3	AG	2109554,5	2269358,7	82203,73
				AG-BL	1739877,4	1846038,3	72836,36
				JFO	1618571,9	1689686,6	68954,93
		100	841380,81	AG	1365457,3	1402433,6	108530,69
				AG-BL	1137594,7	1168756,9	186619,33
				JFO	1020928,5	1053270,4	279188,78
		500	332954,84	AG	376397,4	3800741,3	299284,82
				AG-BL	372364,8	3784936,7	287399,52
				JFO	367629,8	3688739,4	316568,85
		1000	194813,50	AG	269251,18	270153,12	200333,33
				AG-BL	263586,38	263494,23	181569,30
				JFO	241751,14	246836,52	108078,10

5 CONSIDERAÇÕES FINAIS

Este trabalho apresenta os resultados do desempenho das metatheurísticas *Jump Frog Optimization* (JFO), do *Algoritmo Genético* (AG), e do *Algoritmo Genético com Busca Local*

$(AG - BL)$, na busca por melhores soluções para o problema das p−Medianas.

Os métodos são apresentados e aplicados às instâncias descritas em Beasley (1990), Lorena (2001) e Lorena (1999). Nomeadas neste trabalho como instância1, Instância2 e Instância3, respectivamente. No caso da Instância1, não são gerados os $clusters$, ficando a geração dos $cluster$ nas instâncias 2 e 3.

Os resultados obtidos na Instância1 são comparados aos valores encontrados por Bezerra (2008) e os resultados encontrados para Instância 2 e Instância 3, são comparados aos resultados descritos em Bezerra (2008), Lorena (2001) e Lorena (1999).

As soluções da rotina AG-BL mantém-se, em todos os casos, superiores ao AG. Esse melhor refinamento está diretamente relacionado ao maior tempo computacional necessário, quando comparado com o tempo computacional despendido pelo AG. Já o Algoritmo JFO apresentou melhores resultados em todos os testes que foram realizados em comparação ao AG e AG-BL.

Analisando os resultados, percebemos que o AG "pára" em um ótimo local, sem condições de melhora, enquanto que o AG_BL consegue escapar de um ótimo local através da rotina da rotina apresentada anteriormente nesTE trabalho.

Presentemente estão sendo testados instâncias maiores que às apresentadas neste trabalho. No futuro, será utilizado o AG com Busca Local Adaptativo com o objetivo de aproximar ou até mesmo superar os resultados advindos do JFO.

REFERENCES

Arakaki R.G.I. and Lorena L.A.N. Uma heurística de localização-alocação (hla) para problemas de localização de facilidades. 2006.

Beasley J.E. Or-library: distributing test problems by electronic mail, operational research society. 1990.

Bezerra S.N. *Algoritmos Evolutivos Paralelos Aplicados ao Problema das p-medianas*. Master's Thesis, Centro Federal de Educação Tecnológica de Minas Gerais, 2008.

Feo T.A. and Resende M.G.C. Greedy randamized adaptative search procedures. *Journal of Global Optimization*, 6:109 133, 1995.

García F.J.M. and Pérez J.A.M. Optimización por enjambre para la p-mediana continua y discreta. *Quinto Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados.*, 2007.

Goldberg D.E. Genetic algorithms in search optimization e machine learning. 1989.

Grefenstette J.J. Optimization of control parameters for genetc algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 1086, 1986.

Holland J.H. Adaptation in natural and artificial systems. *Univ. Michigan Press, Ann Arbor, MI*, 1975.

Kennedy J. and Eberhart R. Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks Perth, Australia. IEEE Service Center*, pages 1942–1984, 1995.

Lorena L.A.N.; Senne E.P.J.M.S. Integração de um modelo de p-medianas a sistemas de informações geográficas. In *31ž Simpósio Brasileiro de Pesquisa Operacional, Juiz de Fora, MG p. 635-647*. 1999.

Lorena L.A.N. Integração de modelos de localização a sistemas de informações geográficas. *Operations Research*, 2001.

Mladenovic N., Hansen P., and Perez J.A.M. The p-median problem: A survey of metaheuristic approaches. *European Journal of Operational Research*, .:927–939, 2007.

Osman A. and Erhan E. An efficient genetic algorithm for the p-median problem. *Annals of Operations Research*, 122:21 42, 2003.

Senne E.L.F. and Lorena L.A.N. Abordagens complementares para problemas de p-medianas. *Revista Produção - SciELO Brasil*, ., 2003.

Shi Y. and Eberhart R.C. A modified particle swarm optimizer. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. 1998.

Smiderle A., A. S.M.T., and V.E. W. Técnicas da pesquisa operacional aplicadas a um problema de cobertura de arcos. *TEMA, Sociedade Brasileira de Matemática Aplicada e Computacional*, 5:347–356, 2004.

Steiner M.T.A., Correa E.S., and Freitas A.A. A genetic algorithm for solving a capacitated p-median problem. *SpringerLink - Numerical Algorithms - Kluwer Academic Publishers. Printed in the Netherlands*, 2004.

Woeginger G.J. Exact algorithms for np-hard problems a survey. *Springer*, 2570/2003:185 207, 2003.

Zuben F.V. and Castro L. Computação evolutiva: Uma "nova" forma de resolver problemas. *DCA/FEEC/Unicamp.*, ., 2004.