SPOLM 2008

ISSN 2175-6295 Rio de Janeiro- Brasil, 05 e 06 de agosto de 2008.

LOCALIZAÇÃO DE FACILIDADES USANDO ALGORITMOS EVOLUTIVOS PARALELOS: UM ESTUDO VIA PROBLEMA DAS P-MEDIANAS

Sinaide Nunes Bezerra
Centro Federal de Educação Tecnológica de Minas Gerais, PPGMMC
Avenida Amazonas, 7675 - Nova Gameleira, Belo Horizonte, MG, Brasil
sinaide@dppg.cefetmg.br

João Francisco de Almeida Vitor
Centro Federal de Educação Tecnológica de Minas Gerais, PPGMMC
Avenida Amazonas, 7675 - Nova Gameleira, Belo Horizonte, MG, Brasil
joaofrancisco@dppg.cefetmg.br

Sérgio Ricardo de Souza
Centro Federal de Educação Tecnológica de Minas Gerais, PPGMMC
Avenida Amazonas, 7675 - Nova Gameleira, Belo Horizonte, MG, Brasil
sergio@dppg.cefetmg.br

RESUMO

O problema das p-medianas é uma das formulações mais utilizadas para a resolução de problemas de localização de facilidades. O presente trabalho apresenta um estudo comparativo entre aplicações das metaheurísticas Algoritmos Genéticos (AG), Otimização por Colônia de Partículas Discretas (*Discrete Particle Swarm Optimization* - DPSO) e *Jumping Frog Optimizatin* (JFO), em formulações seqüênciais e paralelas, à solução do problema das p-medianas. A metodologia utilizada consiste na aplicação da metaheurística ao problema das p-medianas, afim de encontrar os pontos de facilidade e, em seguida, gera-se a região de atendimento associada, conhecida como *cluster*. Para a geração dos *clusters*, utiliza-se a uma heurística de designação, que visa solucionar problemas de múltiplas facilidades, relacionando cada ponto de demanda à facilidade mais próxima. A geração de *clusters* é refinada pela Heurística de Locação-Alocação (HLA). Testes computacionais, realizados com instâncias da literatura, mostram que as soluções encontradas através da metaheurística JFO são superiores, em todos os casos, aos encontrados através de AG e DPSO.

Palavras-Chaves: Problema das p-Medianas; Algoritmos Evolutivos, Algoritmos Genéticos; *Discrete Particle Swarm Optimization*.

Abstract

The p-median problem is one of most important formulations of the facility location problem. This paper addresses a comparative study concerning the application of Genetic Algorithm (GA), Discrete Particle Swarm Optimization (DPSO) and Jumping Frog Optimization (JFO) metaheuristics, in sequential and parallel formulations, for solving the p-median problem. The adopted medothodology consists in applying the given metaheuristic to the considered problem, in order to find facility points and, in the sequel, get the associated

care region, known as cluster. For the generation of clusters, it is used an assigning heuristic, which aims to resolve problems of multiple facilities, relating each demand point to the closer facility. The generation of clusters is refined by using the Allocation-Location Heuristic. Computational tests are performed with instances from literature, showing that the results found with JFO metaheuristic are better than the ones from GA and DPSO metaheuristics.
Keywords: p-Median Problem; Evolutive Algorithms; Genetic Algorithms; *Discrete Particle Swarm Optimization*.

1. INTRODUÇÃO

Em vários problemas, pertencentes a áreas distintas de conhecimento, torna-se necessário localizar um determinado ponto, dentro de uma região estabelecida, mais adequado para se instalar um serviço específico. Esta questão pode estar associada à localização de fábricas, depósitos, escolas, hospitais, etc. A localização de serviços é importante, seja para o escoamento da produção de maneira mais satisfatória; para atender clientes de uma empresa de maneira mais satisfatória; para a instalação de antenas celulares, que tenham uma maior abrangência; para a instalação de hospitais em regiões que atendam ao maior número possível de habitantes, dentre outros objetivos a serem cumpridos.

Neste contexto, são propostos modelos matemáticos para resolver problemas desta classe. Um exemplo é o Problema das P-Medianas (*p-median problem* - PMP). O PMP é um dos problemas clássicos de localização, que procura localizar, dentro de uma região pré-estabelecida, os melhores locais para se instalar um serviço, denominado facilidade. O objetivo é localizar *p* vértices (denominados medianas) em uma rede contendo *n* vértices, de forma a obter a menor soma das distâncias de cada vértice até a mediana mais próxima. Uma vez localizadas as múltiplas medianas, são formados "*clusters*" (agrupamentos), sendo que cada mediana é alocada a certo número de vértices da demanda.

Os métodos exatos propostos para solucionar problemas assim como PMP encontram dificuldades em relação ao tempo computacional gasto na determinação da solução ótima. Problemas de localização são classificados como problemas NP-difíceis [18] e, neste caso, muitas heurísticas e metaheurísticas têm sido propostas para resolvê-lo, encontrando boas soluções em um tempo computacional aceitável.

Dentre as metaheurísticas propostas, pode-se destacar os métodos baseados em algoritmos evolutivos [1], como Algoritmo Genético (AG) e Otimização por Colônia de Partícula Discreta (*Discrete Particle Swarm Optimization* - DPSO). O AG é utilizado em problemas de localização de facilidades em [10] e [12]. Em [17] o DPSO é apresentado como proposta para resolver o PMP. Outras soluções através de formulações heurísticas diversas podem ser encontradas em [3], onde é apresentado o método AVS (*Adjusted Vertex Substitution*); [29] e [30], nos quais procedimentos de *path-relinking*, juntamente com algoritmos de busca local, são usados; e em [31], onde é usado o método denominado HC (*Heuristic Concentration*). Boas revisões referentes à solução do PMP são apresentadas em [26] e [28].

No presente trabalho, serão apresentadas implementações dos algoritmos evolutivos Algoritmo Genético (AG), DPSO e *Jumping Frog Optimization* (JFO) para a solução do PMP, afim de encontrar os pontos que melhor estão localizados dentro de uma região, que abranja uma região de atendimento específica, determinada pelos algoritmos de Gillet & Johnson [18], juntamente com a Heurística de Locação-Alocação [2] [25]. O algoritmo de Gillet & Johnson é descrito como uma proposta para o problema de roteamento de veículos com vários depósitos [36].

A metodologia adotada consiste em: inicialmente, será feita a abordagem de cada algoritmo em seu modo seqüencial e, em seguida, serão apresentadas as mudanças necessárias para utilização dos métodos em modo paralelo.

Na utilização do modo paralelo, os testes são realizados com 1, 2 e 4 processadores distintos. Apesar de executado em processadores diferentes, o processo tem natureza

cooperativa, tendo, como objetivo, refinar a solução encontrada em cada um dos processadores em separado. Desta forma, é proposta a implementação de uma arquitetura paralela, na qual os algoritmos envolvidos no processo de otimização ajudam-se mutuamente, em um modo cooperativo, visando o refinamento da solução encontrada por cada algoritmo para obter uma melhor solução global para o PMP. Instâncias de teste, encontradas em [24], são utilizadas para efeito de teste computacional das implementações realizadas, sendo comparadas as soluções geradas pelos algoritmos paralelos em relação aos modelos seqüenciais correspondentes.

O trabalho apresenta, na seção 2, as características do PMP; na seção 3, são descritos os algoritmos evolutivos AG e DPSO, no caso de abordagem para o PMP. A seção 4 trata das heurísticas de Gillet e Johnson e da HLA. Nas seções 5 e 6, são discutidas as implementações paralelas propostas para o AG, para o DPSO e para o JFO. A seção 7 mostra os resultados obtidos pelos métodos implementados e a seção 8 conclui o artigo, discutindo os resultados encontrados.

2. PROBLEMA DAS P-MEDIANAS

Problemas de localização e suas aplicações constituem uma importante linha de pesquisa operacional. Tais problemas buscam determinar pontos estratégicos onde serão sediadas facilidades para atender, da melhor maneira possível, um conjunto espacialmente distribuído de pontos de demanda.

O termo “facilidades”, utilizado nos problemas de localização, pode ser substituído por fábricas, depósitos, escolas, hospitais, etc., enquanto “clientes” (pontos de demanda) referem-se a depósitos, unidades de vendas, estudantes, etc. Em geral, várias facilidades podem ser localizadas e, por sua vez, alocadas aos seus clientes. Desta forma, tais problemas são também conhecidos como problemas de localização-alocação [24] [28].

O PMP pode ser descrito como um grafo, no qual cada vértice é visto como uma potencial mediana. 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 (medianas) de forma a minimizar a distância total a outros vértices do grafo [4] e [28].

A soma das menores distâncias entre o vértice v_i, pertencente a V, e todos os outros vértices do grafo é denominada número de transmissão (σ):

$$\sigma(v_j) = \sum_{i=1}^{n} w_i d(v_i, v_j);\ v_i, v_j \in V$$

no qual, n é o número total de vértices do grafo; $d(v_i, v_j)$ é a menor distância entre v_i e v_j; w_i, o peso associado ao vértice v_i.

Assim, v_m é uma mediana se, dentre todos os vértices do grafo, é aquele que produz a menor soma total das distâncias, desde a sua própria posição até cada um dos vértices do grafo. Deste modo, forma-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 [28], abrangendo o máximo possível de arcos com a menor distância entre os pontos [37].

Em certos casos podem existir restrições sobre a capacidade de atendimento das facilidades. Neste tipo de problema, considera-se que cada cliente possui associado um peso de demanda (w_i) a ser satisfeito pela facilidade escolhida para atendê-lo. A soma das demandas de todos os clientes cobertos por uma facilidade não deve superar a capacidade de atendimento da mesma. Quando esse tipo de condicionante estiver presente, diz-se tratar de um problema de localização com restrição de capacidade [24].

Neste trabalho, serão apresentados os algoritmos utilizados para solução do problema, nos quais será adotada que o peso de demanda (w_i) é tomado como sendo igual a 1 (um) para todos os vértices do grafo, significando que apenas as distâncias euclidianas serão consideradas. Assim, o problema é considerado sem restrição de capacidade.

A localização de p-medianas é reconhecida como um problema NP-difícil [18],

sendo, assim, impossível resolvê-lo em tempo polinomial.. Devido à complexidade, algoritmos exatos encontram dificuldades para resolver tais problemas de dimensões encontradas no mundo real [14]. Os problemas de pequeno porte podem ser solucionados por algoritmo exatos, através da aplicação do método exaustivo, que examina todas as combinações possíveis no grafo.

2.1. FORMULAÇÃO POR PROGRAMAÇÃO INTEIRA

Considere um conjunto *L* com *m* localizações potenciais para compor *p* facilidades e um conjunto U contendo localizações para *n* clientes (pontos de demanda). Considere também uma matriz D (*n* x *m*), composta pelas distâncias (custos para que um cliente *i* seja satisfeito por uma facilidade *j*), para todo $j \in L$ e $i \in U$.

O PMP, modelado como problema de programação inteira, é empregado nesse trabalho. Considerando o peso de demanda relativo a cada um dos vértices igual a 1 (problema não capacitado), a formulação matemática, adaptada conforme [14] e [24] é:

$$\text{Minimizar } Z = \sum_{j=1}^{n}\sum_{i=1}^{m} d_{ij}x_{ij}\text{'} \tag{1}$$

$$\text{Sujeito a: } \sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in U \tag{2}$$

$$x_{ij} \leq x_{jj}, \forall i \in U, \forall j \in L \tag{3}$$

$$\sum_{j=1}^{n} x_{jj} = p \tag{4}$$

$$x_{ij}, x_{jj}, \in \{0,1\}, \forall i \in U, \forall j \in L \tag{5}$$

para:

- Z: a função objetivo a ser minimizada;
- *p*: número de facilidades (medianas) a serem localizadas;
- $[d_{ij}]_{nxm}$: matriz de custo (distância), com $d_{ij} = 0$, $\forall i \in U$, $\forall j \in L$;
- $[x_{ij}]_{nxm}$: matriz de alocação, com:

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

e

$$x_{jj} = \begin{cases} 1, & \text{se o vértice } j \text{ é uma facilidade;} \\ 0, & \text{caso contrário.} \end{cases}$$

Quanto às restrições, pode dizer que:

- as restrições (2) e (3) garantem que cada vértice x_i (ponto de demanda) é alocado a somente uma facilidade x_j.
- a restrição (4) determina o número exato de medianas a serem localizadas (*p*);
- a restrição (5) corresponde às condições de integralidade, ou seja, as variáveis de decisão x_{ij} e x_{jj} podem assumir somente valores 0 ou 1.

3. ALGORITMOS EVOLUTIVOS

Os algoritmos evolutivos simulam processos naturais de sobrevivência e reprodução das espécies. Assim como na natureza, indivíduos competem entre si por recursos e pela reprodução. Deste modo, indivíduos mais aptos têm maior probabilidade de se reproduzir e transmitir suas características genéticas às novas gerações e indivíduos menos aptos tendem a ser descartados. Se enquadram neste contexto os algoritmos genéticos (AG) [10], [19], [38], PSO e DPSO [17], [35], [39] e JFO [17]. A seguir, são apresentados os algoritmos AG e PSO, sendo que o PSO aplicado ao PMP apresenta-se na formulação discreta, denominada *Discrete*

Particle Swarm Optmization (DPSO).

3.1. ALGORITMO GENÉTICO

Em síntese, AGs são técnicas metaheurísticas pertencentes à classe dos algoritmos evolutivos. Como todo algoritmo evolutivo, o AG trabalha com uma população de indivíduos no espaço de busca do problema. Em seu algoritmo, a evolução da população se dá através da melhoria genética de seus indivíduos, a cada geração, rumo a melhores soluções, em média.

O termo "genético" tem sua base na Biologia e diz respeito à maneira como as possíveis soluções para o problema tratado são codificadas em cromossomos, estrutura composta por uma cadeia finita de elementos, os genes.

A população inicial de indivíduos pode ser gerada de várias maneiras, sendo, na maioria das vezes, realizada de forma aleatória. Segundo [19], o tamanho da população afeta o desempenho global e a eficiência do algoritmo. Uma população grande fornece uma cobertura representativa do problema, além de prevenir convergências prematuras para soluções locais em vez de globais. Em contrapartida, são necessários mais recursos computacionais.

A avaliação consiste em verificar o valor da aptidão de cada indivíduo da população. É através do cálculo da aptidão que se mede quão próximo um indivíduo está da solução desejada ou quão boa é está solução.

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

Há várias formas possíveis de se fazer o cruzamento. O cruzamento simples, chamado cruzamento de um ponto, prevê apenas um corte, escolhido com probabilidade uniforme sobre o comprimento do cromossomo, [38]. A partir do ponto de corte, cada cromossomo pai é dividido em duas partes, assim os genes dos dois pais são combinados de forma a gerar cromossomos filhos

Terminada a recombinação por cruzamento, os cromossomos filhos podem sofrer mutação. Este operador genético permite introduzir novos indivíduos na população, contribuindo para a manutenção da diversidade da população. A mutação altera arbitrariamente um ou mais genes, resultando em um novo cromossomo. Neste trabalho é utilizado a programação com números inteiros (discretos), sendo o tamanho do cromossomo igual a quantidade de medianas procuradas.

No trabalho, ao seleção do AG é apresentada em [12].

Passo 1 – Inicialização

Construir a população inicial, gerando uma lista R = (r_1, r_2, ..., r_m) com m cromossomos viáveis de p elementos cada, escolhidos randomicamente entre os *v* vértices do grafo:

Calcular C_1 = Aptidão(r_1), $\forall\ r_t \in R$;

Ordenar R de modo que $C_1 \leq C_2 \leq ... \leq C_m$;

Faça k = 0, e o máximo de iterações it_{max} e o máximo de iterações sem melhora 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

Elimine de R o pior cromossomo e selecione dois cromossomos r_i = Seleção(R) e r_j = Selecão(R), $r_i \neq r_j$;

$$Seleção(R) = \left\{ r_j \in R \,/\, j = \left\lceil \frac{-1+\sqrt{1+4.rnd(n^2+n)}}{2} \right\rceil \right\}, \text{ onde rnd} \in [0,1];$$

Passso 4 – Reprodução

Fazer o cruzamento ($r_{i,}$ r_j) = ($r_{x,}$ r_y);

Passso 5 -

Se r_x e r_y forem cromossomos viáveis fazer:

$$\begin{cases} r_m = r_x, \text{se } aptidão(r_x) \leq aptidão(r_y) \\ r_m = r_y \text{ caso contrário} \end{cases}$$

Ir ao passo 7;

Passso 6 – Mutação

Caso r_x ou r_y não viáveis, escolher r_x ou r_y, fazer a mutação produzindo r_m.

Passso 7 – Aptidão

Calcular Cm = *aptidão*(r_m).

Inserir o cromossomo em R, mantendo a ordem crescente de aptidão.

Fazer k = k+1 e voltar ao Passo 2

3.2. OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS

Otimização por Enxame de Partículas (*Particle Swarm Optimization* - PSO) é uma técnica metaheurística baseada no comportamento social de peixes, aves, dentre outros animais, proposta por [21]. Seus autores se inspiraram no comportamento cooperativo de vários tipos de animais, dando atenção especial aos modelos baseados em revoada de pássaros. Nesses modelos, cada pássaro adapta sua posição e velocidade para, essencialmente buscar uma posição mais promissora no espaço de busca. A busca por alimento ou pelo ninho e a interação entre os pássaros (chamados de partículas no algoritmo) ao longo do vôo são modelados como mecanismos de otimização. Neste sentido, encontrar o local com comida ou o ninho corresponde a localizar a melhor solução do problema.

O PSO possui muitas similaridades com o AG. Apesar de não possuir operadores genéticos, sua população evolui no espaço de busca, através da melhoria das posições das partículas, a cada iteração, rumo a melhores soluções, em média.

O algoritmo proposto por [34], considera o espaço de busca contínuo, significando que as partículas podem ocupar quaisquer posições dentro do domínio do problema. Cada partícula é tratada como um ponto no espaço N-dimensional.

O algoritmo possui, inicialmente, uma população de soluções potenciais (partículas) com suas respectivas posições aleatórias. A cada uma delas é atribuída uma velocidade e as partículas passam a se movimentar pelo espaço de busca. Cada uma das partículas possue "memória", armazenando nesta a sua melhor posição prévia (*pbest*) e também a melhor posição prévia já alcançada pelo enxame (*gbest*).

A cada iteração, a atualização da partícula *i* dá-se pelo acréscimo da velocidade (taxa de variação da posição), em todas as dimensões, fazendo com que ela tenda gradualmente para seus melhores valores históricos, *pbest* e *gbest*.

As partículas são atualizadas conforme Eq.3.1 e 3.2, apresentadas em [34]:

$$v_{in}^{t+1} = wv_{in}^{t} + c_1 r_1 (pbest_{in} - x_{in}^{t}) + c_2 r_2 (pbest_{in} - x_{in}^{t}) \quad (6)$$

$$x_{in}^{t+1} = x_{in}^{t} + v_{in}^{t+1} \quad (7)$$

sendo

w: peso inercial, controla o impacto da velocidade prévia na velocidade corrente;

c_1 e c_2: constantes positivas, parâmetros cognitivo e social, respectivamente;

r_1 e r_2: números randômicos distribuídos uniformemente no intervalo [0-1];

t = 1, 2, ..., $iter_{max}$: número de iteração corrente, com o limite máximo igual a $iter_{max}$.

O algoritmo básico para o PSO é mostrado a seguir:

Passo1: Inicie randomicamente a posição e a velocidade das N partículas da população;

Passo2: Avaliar as partículas;

Passo3: Atualizar pbest$_i$. Se $f(x_i) < f(pbest_i)$, então $pbest_i = x_i$;
Passo4: Atualizar gbest. Se $f(pbest_i) < f(gbest)$, então gbest = $pbest_i$;
Passo5: Atualizar a velocidade de cada partícula conforme Eq. (3.1);
Passo6: Atualizar a posição de cada partícula conforme Eq. (3.2);
Passo5: Enquanto critério de parada não atingido) volte ao passo 2;

3.3. OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS DISCRETO

Muitos problemas de otimização são aplicados a espaço de características discretas. Exemplos típicos incluem problemas de ordenação e arranjos de elementos discretos e problemas combinatoriais [22]. É neste cenário que o algoritmo PSO discreto, denominado *Discrete Particle Swarm Optmization* (DPSO), é aplicado para a solução do PMP neste trabalho [14].

De acordo com [22], no espaço discreto, uma partícula (indivíduo da população, $\vec{X}(i) = (x_{(i,1)}, x_{(i,2)}, ..., x_{(i,n)})$) pode ocupar vértices de um hipercubo e mover-se suavemente através da variação das coordenadas de suas variáveis. No algoritmo DPSO, o enxame de partículas trabalha através do ajuste de trajetórias a partir da manipulação das coordenadas de $x_{(i,d)}$.

O DPSO mantém a estrutura básica aplicada ao PSO clássico descrito na subseção 3.2, entretanto, para operar em espaço discreto, algumas adequações são necessárias.

No algoritmo DPSO proposto por [13], a partícula $X(i)$ possui k atributos. Cada atributo é identificado por um único número inteiro ou índice. Inicialmente os atributos são escolhidos de forma aleatória da seqüência $I = \{1, 2, 3, ..., n\}$, onde n corresponde ao limite do domínio discreto, um de cada vez, e sem repetição. Cada partícula $X(i)$ é associada a uma matriz, V^i_{2xn}, denominada probabilidade proporcional. Assim, $V^i_{2xn} = \begin{pmatrix} \text{probabilidade de escolha} \\ \text{índices dos atributos} \end{pmatrix}$, sendo que os n elementos da primeira linha definem a probabilidade de uma partícula ser ou não escolhida para compor o próximo candidato a solução. A segunda linha apresenta os índices dos atributos relativos a cada uma das probabilidades.

A matriz V^i_{2xn} é então usada para atualizar a nova partícula associada a $X(i)$. Segundo os critérios de atualização inicialmente todos os elementos da primeira linha de V^i_{1xn} recebem o mesmo valor, 1, indicando que todas as partículas têm a mesma probabilidade de serem escolhidas,ficando assim, $V^i_{2xn} = \begin{pmatrix} 1 & 1 & 1...1 \\ 1 & 2 & 3...n \end{pmatrix}$. Em seguida três parâmetros, α, β e γ, associados a $X(i)$, $B(i)$ e G, respectivamente, são acrescidos à primeira linha de V^i. Esses parâmetros determinam a contribuição de $X(i)$, $B(i)$ e G na formação da próxima partícula. Assim, todos os atributos cujos índices estão presentes em $X(i)$ têm sua probabilidade aumentada por α, o mesmo ocorrendo com todos os atributos em $B(i)$, que têm sua probabilidade aumentada por β, o mesmo ocorre com os atributos presentes em G, que são aumentados de γ. Em seguida a primeira linha de V^i é multiplicada por um parâmetro randômico uniforme no intervalo [0-1], gerado de forma independente para cada um dos atributos, em seguida ordena-se V^i em relação à primeira linha de forma decrescente. Os k primeiros índices são escolhidos para compor a nova partícula.

3.4. ALGORITMO JUMPING FROG OPTIMIZATION

Uma nova proposta de adaptação do método PSO para problemas de otimização discretos é proposta por [17]. Esta versão, denominada otimização por salto de rã (*Jumping Frog Optimization* - JFO), tem sua inspiração no movimento de um grupo de rãs saltando de pedra em pedra em certo intervalo de tempo.

No algoritmo JFO, um conjunto de partículas se locomove no espaço de busca discreto dando saltos 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 (como componente de atualização da posição da partícula) é substituída por saltos esporádicos e aleatórios no espaço de busca.

O algoritmo JFO mantém as características básicas do algoritmo PSO contínuo [21], guardando informações sobre suas melhores posições prévias, individuais e coletivas. A mudança proposta pelo JFO encontra-se na atualização da posição de cada partícula, que consiste em realizar movimentos aleatórios de melhora da partícula. Tais movimentos permitem substituir uma solução por outra dentro do espaço de busca. Assim, a atualização da posição de cada partícula é efetuada conforme procedimento abaixo:

Considere um conjunto X composto pelas posições das partículas, x_j, com $i = 1, 2, ..., n$ (n corresponde ao número de posições domínio discreto) e um intervalo unitário I composto pelas probabilidades de movimentos, c_j, com $j = 1, 2, 3, 4$. A atualização da partícula é obtida a partir dos seguintes passos:

Passo1: Eleger aleatoriamente uma posição $p_j \in X$, denominada atrator da partícula i;

Passo2: Dividir o intervalo unitário I, em quatro intervalos com probabilidades c_1, c_2, c_3 e $c_4 = 1-(c_1 + c_2 + c_3)$, $(c_1 + c_2 + c_3 = 1)$.

Passo3: Gerar um número aleatório, *rand*, no intervalo [0,1] e verificar o segmento ao qual ele pertence dentro do intervalo I, indicando a probabilidade de movimentos (c_j) associada;

Passo4: Gerar outro número aleatório, *Rand*, no intervalo [0,1];

Passo5: Aplicar movimentos, a cada iteração, segundo *ProbMov* de tal forma que:
Se *ProbMov* > 1, então aplicar movimento.
Sendo $ProbMov = Rand \cdot p_i \cdot c_j$, com a probabilidade de movimentos, c_j, correspondente ao atrator p_i.

Passo 6: Atualizar a posição das partículas como segue:
Se *ProbMov* > 1, então p_i é atualizado com uma posição aleatória: $X - \{p_i\}$
Senão terminar iteração.

4. GERAÇÃO DOS CLUSTERS

O método que permite determinar a região atendida por cada facilidade (*cluster*) é implementado pelo algoritmo de designação de Gillet e Johnson [19]. Este método visa solucionar problemas de múltiplas facilidades, relacionando cada ponto de demanda à facilidade mais próxima. O algoritmo é descrito a seguir:

Passo1: Calcula-se a distância entre cada nó ainda não designado até cada mediana;

Passo2: 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;

Passo3: 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;

Passo4: Designar os nós i para as medianas mais próximas, até sua capacidade de atendimento esgotar. 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.

Após terminado o algoritmo de determinação, é aplicada a Heurística de Locação-Alocação (HLA), proposta em [24], que busca melhorar, pelo uso de uma busca local, a distância dentro de um cluster já formado alterando a mediana com um vértice a ela designado. A HLA é empregada em trabalhos de [2] e [21].

A seguir segue o pseudo-código descrito em [2] e adaptado neste trabalho para o PMP não capacitado.

Dados J conjunto das medianas = $\{j_1, ..., j_p\}$ e

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 $k = 1, ..., p$ faça
        Para $i = 1, ..., |C_k|$ faça
            Troque mediana j_k 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 ← *novasol;*
    Fim_Se;
Fim_Enquanto;

5. COMPUTAÇÃO PARALELA

A idéia básica da maioria dos programas paralelos é dividir uma tarefa em partes menores e solucionar as partes simultaneamente, usando múltiplos processadores. Esta abordagem de dividir-e-conquistar pode ser aplicada em Algoritmos Genéticos de muitos modos diferentes. Alguns métodos de paralelismo usam uma única população, enquanto outros dividem a população em várias subpopulações relativamente isoladas [7]. Devido às similaridades dos métodos AG e DPSO, a abordagem de dividir-e-conquistar, descrita para o AG, aplica-se de forma análoga ao DPSO paralelo.

Estratégias de busca paralelas executam, freqüentemente, uma exploração mais completa no espaço de solução do problema quando comparadas àquelas geradas pelas metaheurísticas seqüenciais correspondentes [14]. A qualidade da solução pode ser ampliada, quando é possível intensificar a varredura do espaço de busca através da comunicação entre os melhores indivíduos (possíveis soluções) de cada população, através do processo de migração.

Neste trabalho é utilizado o modelo denominado modelo de ilhas ou granularidade grossa [1], [5] e [8]. O modelo de paralelismo aplicado ao DPSO é análogo ao AG [5].

5.1. MODELO DE ILHAS

Neste modelo, o procedimento seqüencial do AG é executado em subpopulações (ilhas), que trocam informações através da migração dos seus melhores indivíduos (possíveis soluções) de cada geração. O modelo de ilha introduz o processo de migração, usado para enviar indivíduos de uma subpopulação para outra. O processo de migração é, em geral, acionado apenas quando ocorre estagnação da evolução de uma das subpopulações, permitindo a diversificação da mesma. O critério de parada local de um determinado processo é baseado em uma condição global que envolve todos os processos individuais.

Segundo [9], o processo de migração é controlado através de vários parâmetros. Tais parâmetros permitem determinar o destino dos indivíduos, a freqüência da migração e o número de indivíduos que serão trocados (taxa de migração).

Segundo [6], a migração de indivíduos de uma população para outra garante que nenhum nó perca muito tempo trabalhando com uma população cujo resultado seja ruim, pois esta será melhorada com a chegada de indivíduos de ilhas vizinhas. O fato de cada nó trabalhar a maior parte do tempo de forma independente acarreta um baixo tráfego de dados

na rede, proporcionando alta velocidade de comunicação entre os processadores.

6. ARQUITUTURA PROPOSTA

A implementação proposta neste trabalho inspira-se em [23], [25] e [32]. No modelo, um processador central, denominado Gerenciador, envia os parâmetros de funcionamento para cada algoritmo em cada ilha (processador). Em ilhas distintas, os algoritmos executam os procedimentos de otimização de forma independente, com as características pertinentes a cada método. Cada um dos algoritmos busca a melhor solução para o problema e sempre que uma solução, melhor que a já conhecida, é encontrada por algum algoritmo, inicia-se o processo de migração desta solução para o Gerenciador. Desta forma, o Gerenciador mantém uma lista de tamanho fixo com os melhores indivíduos já encontrados. Trata-se no gerenciador como indivíduo, mas também poderá ser uma partícula, pois está relacionado ao DPSO. Com isso, o termo "indivíduo" é usado de forma genérica. Inicialmente, está lista é gerada aleatoriamente. Ao longo do processo de migração, o pior indivíduo é substituído, mantendo-se assim o elitismo do processo.

Os eventos ou comandos entre Gerenciador e algoritmos ocorrem através de comunicação via *socket*. *Sockets* são mecanismos usados para prover o envio e recepção de mensagens e que pode ser definido como "um tipo de arquivo usado em uma rede de computadores para comunicação entre várias máquinas" [25]. Estas mensagens são comandos ou eventos que trafegam nos dois sentidos, Gerenciado para algoritmos ou algoritmos para Gerenciador. Os trabalhos [25] e [15] apresentam implementações que utiliza-se sockets como forma de comunicação.

Quando todos os processadores (algoritmos) estão prontos, o Gerenciador envia o comando "Iniciar". Ao encontrar uma solução melhor que a já conhecida, o processador a envia para o Gerenciador, dando início ao processo de emigração. Caso o processador não evolua mais sua população, informa o estado em que está estagnado através do evento "Estagnado"; o Gerenciador, então, procura na lista se existe alguma solução que seja melhor que a já encontrada pelo processador estagnado e a envia para esse processador através do processo "Imigrar". A imigração é acionada apenas quando um dos algoritmos requisitar novos indivíduos, com o objetivo de diversificar sua população, que se encontra temporariamente estagnada. A quantidade de indivíduos requisitados pode variar entre um (o melhor) ou vários indivíduos. Ao receber a solicitação, o processador mestre envia novos indivíduos para o algoritmo que fez a solicitação.

Arquiteturas paralelas são largamente empregadas em problemas de otimização juntamente com algoritmos evolutivos como AG e PSO. No caso de AG, encontra-se utilizado em diversos trabalhos, como [1], [6], [11], [15]. No caso do PSO, há relatos de implementações paralelas em [5], [27] e [33].

Em síntese, o processo funciona da seguinte forma: Os algoritmos rodam em processadores diferentes, recebem os respectivos parâmetros do Gerenciados, são iniciados através do comando "Iniciar" enviado pelo Gerenciador para todos os processadores, cada processador carrega os dados de um repositório central, assim que algum encontra uma solução de melhora está é enviada ao Gerenciador que a armazena, caso seja melhor que alguma já conhecida. Quando o algoritmo atinge a quantidade de iterações determinada para solicitar novos indivíduos, it_mig, o comando "Solicitar imigração" é disparado, com isso o Gerenciador envia seus melhores indivíduos com a quantidade determinada pelo parâmetro qtd_imigrantes e o algoritmo continua o processo de melhora da solução. Se it_estagnado é atingido, o estado "Estagnado" é informado ao Gerenciador, que procura indivíduos melhores que os já encontrados pelo processador parado; caso encontre, este informa ao processador estagnado que, por sua vez, dispara o comando "Solicitar imigração". O processamento se repete até que it_sm ou it_max seja atingido.

A verificação da função de aptidão se dá quando o algoritmo gera um conjunto V candidato à solução. A partir deste conjunto, é aplicada a heurística de designação de Gillet &

Johnson [19], gerando assim, os primeiros clusters; em seguida aplica-se a HLA, afim de refinar a solução, procurando diminuir o custo total da mesma.

7. TESTES COMPUTACIONAIS

Para a verificação da eficácia do AG, DPSO e JFO, utilizou-se a instância descrita em [24] com 324 pontos no modelo não capacitado. O algoritmo foi implementado em linguagem de programação C++ Builder-5, e executado 20 vezes para cada mediana em computador Pentium de 3 GHz com 1 GB de memória RAM e sistema operacional Windows XP, Para o teste com vários computadores, a configuração é a mesma, além dos computadores para executar os algoritmos, ilhas, utilizou-se um a mais, posto como repositório .

Nos testes, utilizou-se 50 pássaros para DPSO e JFO e 200 indivíduos para AG no modo seqüencial. No modo paralelo, a estratégia adotada neste trabalho é manter o mesmo número de iteração por ilha (processador) igual ao utilizado no modo seqüencial e dividir os números de indivíduos/partículas por processador. Como número máximo de iterações, it_max, utiliza-se 1000. Adota-se neste trabalho dois critérios de parada: (a) o número máximo de iterações (it_max); (b) o número máximo de iterações sem melhora (it_sm).Desta forma, após um conjunto de iterações consecutivas, sem melhora da solução (it_sm), ou até atingir it_max, o programa deve parar. Neste trabalho, utilizou-se it_sm = 0,2 * it_max, ou seja, it_sm igual a 200 iterações. Além dos parâmetros mencionados, deve-se informar também quando ocorre a imigração para as ilhas e em que momento uma população de uma ilha é considerada estagnada. Neste trabalho, a imigração de novos indivíduos para uma ilha ocorre quando está atingir it_mig e é considerada estagnada quando atingir it_estaganado, sendo seus valores dados por: a) it_mig = 0,1 * it_sm $\Rightarrow$ it_mig = 20; b) it_estagnado = 0,3 * it_sm $\Rightarrow$ it_estagnado = 60.

Além destes parâmetros, é necessário informar a quantidade que irá migrar para as ilhas, denominado qtd_imigrantes. Neste trabalho a quantidade de imigrantes utilizada é 40% do tamanho da população por ilha. Para exemplificar, se o tamanho da população é 100 indivíduos então serão migrados 40.

Os resultados obtidos com a estrutura proposta estão apresentados nas Tabelas 1, 2 e 3. Nestas tabelas, tem-se que: n, quantidade total de pontos; p, quantidade de medianas; s, solução encontrada por [24]; alg, algoritmo; pop, tamanho da população; σ, melhor solução encontrada; m_σ, solução média; t_m, tempo médio em segundos das soluções.

Tabela 1 – Valores para um processador.

n	p	s	alg	pop	σ	m_σ	t_m(s)
324	5	122518	DPSO	50	127504	128296,55	217,82
			JFO	50	127006	129739,05	450,15
			AG	200	127920	128554,18	16,22
	10	79256,36	DPSO	50	82805,5	86709,21	151,57
			JFO	50	80277,8	80689	865,89
			AG	200	82853,2	85610,63	14,5
	20	54533,11	DPSO	50	60337,3	62566,44	198,09
			JFO	50	55428,9	56057,52	2437,35
			AG	200	60109,6	61954,04	15,68
	50	32101,52	DPSO	50	39049,2	40571,86	211,85
			JFO	50	33295,4	33458	9228,58
			AG	200	38657	39793,26	19,6

Tabela 2 – Valores para dois processadores.

n	p	s	alg	pop	σ	m_σ	t_m(s)

324	5	122518	DPSO	50	127637	128147,66	123,68
			JFO	50	127117	127550,23	126,97
			AG	200	127707	128546,96	87,13
	10	79256,36	DPSO	50	84542,7	86288,64	103,56
			JFO	50	80458,3	80783,62	423,93
			AG	200	82571,1	86924,31	47,81
	20	54533,11	DPSO	50	62088	63130,22	89,64
			JFO	50	55376,8	55619,96	1151,64
			AG	200	60100,7	61500,82	52,73
	50	32101,52	DPSO	50	39011,2	40533,83	118,45
			JFO	50	33390,7	33555,57	3526,84
			AG	200	37503	38987,23	91,53

Tabela 3 – Valores para quatro processadores.

n	**p**	**s**	**alg**	**pop**	**σ**	**m_σ**	**t_m(s)**
324	5	122518	DPSO	48	127749	130134,58	52,86
			JFO	48	126912	127509,48	72,12
			AG	200	127390	128665,91	189,72
	10	79256,36	DPSO	48	82135,6	87357,71	41,21
			JFO	48	80441,9	80813,96	203,17
			AG	200	82325,1	84254,34	138,64
	20	54533,11	DPSO	48	60363,3	63135,31	45,71
			JFO	48	55377,3	55770,06	570,65
			AG	200	58069,1	59563,35	101,65
	50	32101,52	DPSO	48	39437,9	41008	50,68
			JFO	48	33378,3	33549,1	1749,53
			AG	200	35415,8	37190,89	51,85

O tamanho da população é dividida nos processadores. Assim, tendo o DPSO como base com 1 processador, a população utilizada neste trabalho é igual a 50; com 2 processadores, 25 indivíduos em cada processador; e, com 4 processadores, tem-se 12 em cada, tendo uma diferença de dois indivíduos na população total.

8. CONCLUSÃO

Este trabalho apresenta os resultados do desempenho das metaheurísticas Algoritmo Genético (AG), Otimização por Colônia de Partículas Discretas (DPSO) e *Jumping Frog Optimization* (JFO), na busca de soluções para o problema de localização de facilidades denominado problema das p-medinas. No trabalho, utilizou-se um modo de trabalho distribuído em várias máquinas e seu funcionamento se baseia nos estudos relativos aos algoritmos evolutivos paralelos.

Utiliza a instância descrita por Lorena [24] e foram feitas comparações entre os valores conhecidos em [24] com os valores encontrados no presente trabalho. Para tal, utiliza-se uma instância com 324 pontos em que serão localizadas 5, 10, 20 e 50 facilidades.

As soluções da rotina JFO mantêm-se, em todos os casos, superiores ao AG e DPSO. Esse melhor refinamento do JFO está diretamente relacionado com o maior tempo computacional necessário, quando comparado com o tempo computacional despedido pelo AG ou DPSO. Observa-se que à medida que se aumenta o número de processadores e divide-se a população por estes processadores, o tempo gasto na solução em geral diminui. No caso do AG, o tempo gasto não diminui quando se aumenta o número de processadores, devido ao

tempo gasto na comunicação entre máquinas.

A solução é melhorada com o aumento de computadores nos três algoritmos testados.

Conclui-se que com o aumento de máquina em arquitetura distribuída em um modo cooperativo entre os algoritmos pode-se melhorar substancialmente os valores encontrados para problemas de otimização como no caso do PMP aqui testado. Atrelado a este fato, o tempo computacional também é sensivelmente menor que o modo seqüencial, e este pode ser um fator relevante em problemas nos quais o tempo computacional é fator de forte restrição.

9. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Alba, E. & Tomassini, M., Parallelism and Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation, 6, 5, 443- 462, 2002.

[2] Arakaki, R. G. I., Lorena, L. A. N., Uma heurística de localização-alocação (HLA) para problemas de localização de facilidades. Produção, 16, 2, 319-328, 2006.

[3] Ashayeri, J., Heuts, R., Tammel, B., A modified simple heuristic for the p-median problem, with facilities design applications. Robotics and Computer-Integrated Manufacturing, 21, 451–464, 2005.

[4] Avella, P., Sassano, A. e Vasil'ev, I., Computational study of large-scale p-Median problems, Math. Program., A 109, 89-114, 2006.

[5] Belal, M. & El-Ghazawi, T., Parallel Models for Particle Swarm Optimizers, IJICIS, 4, 1, 100-111, 2004.

[6] Campos, G. G. *et al.*, Algoritmos Genéticos e Computação Paralela para Problemas de Roteirização de Veículos com Janelas de Tempo e Entregas Fracionadas, Gestão e Produção, 13, 2, 271-281, 2006

[7] Cantú-Paz, E., A survey of parallel genetic algorithms, Technical report, Computer Science Department and Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 1997

[8] Cantú-Paz, E., Implementing Fast and Flexible Parallel Genetic Algoritms, Handbook of Pratical Genetic Algoritms, Illinois Genetic Algorithms Laboratory, v. 3, University of Illinois at Urbana-Champaign, 1998.

[9] Cantú-Paz, E., & Goldberg, D. E., On the Scalability of Parallel Genetic Algoritms, Evolutionary Computation, 7, 429-449, 1999.

[10] Chaudhry, S.S *et al*, Solving a class of facility location problems using genetic algorithms. Expert Systems, 20, 2, 86-91, 2003.

[11] Chakraborty, R., Dutta A., Island Model Parallel Genetic Algorithm for Optimization of Symmetric FRP Laminated Composites. LNCS 4297, 217 – 228, 2006.

[12] Correa, E. S., *et al*, A genetic algorithm for solving a capacitated p-median problem. Numerical Algorithms, 35, 373–388, 2004.

[13] Correa, E. S, Freitas, A. A. e Johnson, C. G., A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics data set, *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation(GECCO)*, 35-42, 2006.

[14] Crainic, T. G., *et al.*, Cooperative Parallel Variable Neighborhood Search for the p-Median, Journal of Heuristics, 10, 293–314, 2004.

[15] Dubreuil, M., Gagné, C., Parizeau, M., Analysis of a Master-Slave Architecture for Distributed Evolutionary Computations. *IEEE Transactions on Systems, Man, and Cybernetics*, 36, 1, 229-235, 2006.

[16] Galvão, R. D., A dual-bounded algorithm for the p-median problem. Operations Research, 28, 1112-1121, 1980.

[17] García, F. J. M. & Pérez, J. A. M., Optimización por Enjambre para la p-Mediana Continua y Discreta. *5º Congreso Espanhol de Metaheurísticas, Algoritmos Evolutivos e Bioinspirado,* MAEB, 2007.

[18] Garey, M. R. & Jonhson, D. S., Computers and intractability: a guide to the theory of NP-completeness. W.H. Freeman and Co., San Francisco, CA, 1979.

[19] Gillet, B. & Johnson, J., Multi-terminal vehicle-dispatch algorithm. Omega, 4, 711-718.

[20] Grefenstette, J. J., Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, v. smc-16, 1, 122-128, 1986.

[21] Hoffman, L. T, Gómez, A. T., Utilização da Pesquisa Tabu na geração de um Sistema de informação Geográfica aplicado ao problema de localização de torres de rádio transmisão

[22] Kennedy, J. & Eberhart, R. C., Particle Swarm Optimization, In Proceedings of the IEEE International Conference on Neural Networks, Piscataway, NJ, 1942-1948, 1995.

[23] Kennedy, J. & Eberhart, R. C., A discrete binary version of the particle swarm algorithm: In Proceedings of the Conference on Systems, Man, and Cybernetics, Piscataway, NJ, USA, 4104-4109, 1997.

[24] Koh, B., *et al,* Parallel Asynchronous Particle Swarm Optimization. *Int J Numer Methods Eng*., 23; 67(4), 578–595, 2006.

[25] Lorena, L. A. N., *et al*., Integração de Modelos de Localização a Sistemas de Informações Geográficas, *Operations Research*, 8, 2, 180-195, 2001.

[26] Mayer, M. K., Plebani, L. J, A parallel algorithm manager for networked workstations. *Annals of Operations Research*, 71, 229 – 258, 1997.

[27] Mladenovic, N., *et al*., The p-Median Problem: A Survey of Metaheuristic Approaches, Europen Journal of Operation Research, 179, 927-939, 2007.

[28] Mostaghim, S., Branke, J., Schmeck, H., Multi-Objective Particle Swarm Optimization on Computer Grids. *Conference on Genetic and Evolutionary Computation(GECCO)*, 869-875, 2007.

[29] Reese, J., Methods for Solving the p-Median Problem: An Annotated Bibliography, Department of Mathematics, Trinity University, 2005.

[30] Resende, M. G. C., Werneck, R. F., On the implementation of a swap-based local search procedure for the p-median problem. AT&T Labs Research Technical Report, 2002.

[31] Resende, M. G. C., Werneck, R. F., A Hybrid Heuristic for the p-Median Problem. AT&T Labs Research Technical Report, 2003.

[32] Rosing, K. E. & Hodgson, J., Heuristic concentration for the p-median: an example demostrating how and why it works. *Computers & Operations Research*, 29, 1317-1330, 2002.

[33] Sambatti, S.B., *et al.*, Some Parallel Strategies for an Epidemic Genetic Algorithm Applied to an Inverse Heat Conduction Problem. *In Proceedings of International Conference on Computational and Experimental Engineering and Sciences*, 2004.

[34] Schutte, J. F., *et al*, Parallel global optimization with the particle swarm algorithm. International Journal for Numerical Methods in Engineering, 61, 2296–2315, 2004

[35] Shi,Y., Eberhart , R. C., A modified particle swarm optimizer, In Proceeding of the IEEE Congress on Evolutionary Computation (CEC), Piscataway, NJ, pp. 69-73, 1998.

[36] Shi,Y., Particle Swarm Optimization. *IEEE Neural Networks Society*, 8-13, 2004.

[37] Simoneto, E. O., Borenstein, D., Gestão Operacional da Coleta Seletiva de Resíduos Sólidos Urbanos – Abordagem Utilizando um Sistema de Apoio à Decisão. Gestão e Produção, 13, 3, 449-461, 2006.

[38] Smiderle, A., *et al*, Técnicas da Pesquisa Operacional Aplicadas a um Problema de Cobertura de Arcos, TEMA, Sociedade Brasileira de Matemática Aplicada e Computacional, 5, 2, 347-356, 2004.

[39] Srinivas M. e Patnaik, L. M., Adaptive probabilities of crossover and mutation in genetic algorithm. *IEEE Transactions on Systems, Man and Cybernetics*, 24, 4, 656-667, 1994.

[40] Yang, et al, A Quantum Particle Swarm Optmization. Proceeding of the 2004 IEEE Congress on Evolutionary Computation, 1, 320-324, 2004.