ALGORITMO DE COLÔNIA ARTIFICIAL DE ABELHAS PARA UM PROBLEMA DE CLUSTERIZAÇÃO CAPACITADO

Luiz Augusto Canito Gallego de Andrade
Programa de Mestrado em Engenharia de Sistemas Logísticos – Escola Politécnica da USP
luizacga@gmail.com

Cláudio Barbieri da Cunha
Programa de Mestrado em Engenharia de Sistemas Logísticos – Escola Politécnica da USP
Av. Professor Almeida Prado, Travessa 2, nº 83 – São Paulo, SP, Brasil
cbcunha@usp.br

RESUMO

Este artigo trata de um algoritmo de Colônia de Abelhas baseado no comportamento de busca de alimentos desses animais, para encontrar boas soluções para um problema de clusterização capacitado que pode ser modelado como um problema de p-medianas capacitado. Este problema pode ser definido como encontrar o melhor agrupamento de pontos, caracterizados pelas suas posições e demandas, em um número pré-definido de conjuntos com uma dada capacidade. Esse problema apresenta muitas aplicações práticas, dentre as quais podem ser citadas: agrupamento de clientes em centros de distribuição, procedimentos estatísticos de agrupamento de dados, etapas iniciais de problemas de roteirização. O problema possui ordem de dificuldade não polinomial, fazendo que a sua solução até a otimalidade seja inviável em instâncias reais, por isso o algoritmo foi elaborado. Os resultados de testes foram comparados com outros resultados da literatura e uma avaliação crítica da aplicação foi feita.

ABSTRACT

In this article we propose an Artificial Bee Colony Algorithm, inspired on beehives' foraging behavior to solve a capacitated clustering problem that can be modeled as a capacitated p-median problem. This problem can be defined as follows: given a set of points, each of them with known coordinates and demand, partition this set into a predefined number *p* of clusters, each one defined by its median point, such as the allocation cost is minimal, and no cluster exceeds its capacity. There are many practical applications of this problem, such as defining client-provider allocation maps, statistical clustering analysis, and routing problems. We also provide the results of computational experiments in which our algorithm is compared to other results found in the literature.

KEYWORDS: Meta-heuristic, Artificial Bee Colony, Capacitated Clustering Problem.

1. Introdução

Neste trabalho considera-se um problema de clusterização (ou agrupamento) capacitado (CCP), também conhecido como o problema "*p*-medianas capacitado" (SCHEUERER; WENDOLSKY, 2006). Basicamente o problema pode ser definido da seguinte forma: dado um conjunto de *n* clientes e suas respectivas demandas, particionar esses pontos em *p* conjuntos ou agrupamentos, chamados *clusters*, de tal modo que a capacidade de cada um dos *p* conjuntos não seja ultrapassada e a soma das distâncias entre os pontos e a mediana de seus respectivos *clusters* seja mínima. Para maiores referências sobre o problema sugere-se consultar, por exemplo, Mulvey e Beck (1984).

O CCP pode ser visto como um problema de localização, uma vez que as medianas dos *clusters* podem ser vistas como instalações prestadoras de serviço e os clientes e a suas respectivas alocações como uma relação de clientes e instalações. Cada *cluster* possui uma determinada capacidade e não se podem alocar clientes a esse determinado *cluster* além da sua capacidade. Dentre os problemas práticos que podem ser modelados como o CCP pode-se citar a alocação de clientes a veículos em problemas de roteirização (KOSKODIS; POWEL, 1992), problemas de localização de armazéns, entre outros.

O problema considerado possui complexidade não polinomial, conforme demonstrado por Garey e Johnson (1979). Isso decorre da sua característica combinatória, o que torna a resolução de instâncias reais de CCP até a solução ótima algo custoso do ponto de vista computacional. Assim sendo, heurísticas para solução de problemas dessa natureza são uma alternativa promissora. Algumas heurísticas já foram empregadas para a solução do CCP. Uma aplicação interessante de um algoritmo de *Variable Neighborhood Search* (VNS) foi documentada em Fleszar e Hindi (2008). O problema também já foi resolvido por um algoritmo de Busca Tabu (TS) por França *et al.* (1999). Em Boccia et al. (2008) os autores desenvoveram um algoritmo baseado em planos de corte de Fenchel que foi implementado em um software comercial de programação inteira mostrando bons resultados na solução do problema.

Uma classe de heurísticas que vêm surgindo com resultados promissores são os chamados algoritmos populacionais baseados na inteligência coletiva de sociedades animais, os chamados algoritmos de *Swarm Optimization*. Alguns algoritmos pertencentes a esse conjunto são: algoritmos de colônias de formigas (SHELOKAR *et al.*, 2004), *Particle Swarm Optimization* (PSO) (JARBOUI *et al.*,2007), entre outros. Um algoritmo desse tipo, que procura imitar o comportamento de colônias de abelhas nas atividades de busca de alimentos é o algoritmo de Colônia Artificial de Abelhas (*Artificial Bee Colony* – ABC) (KARABOGA; AKAY, 2009). O algoritmo ABC foi empregado em problemas de otimização com espaço de busca contínuo em Akay e Karaboga (2010), apresentando bons resultados.

Algoritmos de clusterização baseados no comportamento das abelhas também já foram desenvolvidos. Um algoritmo para problemas de clusterização não capacitados foi desenvolvido em Zhang *et al.* (2010). Os autores utilizaram uma abordagem similar a Akay e Karaboga (2010), realizaram testes em instâncias de problemas de agrupamento de massas de dados, e obtiveram resultados melhores em relação à outros autores, superando os resultados obtidos com heurísticas de colônia de formigas, algoritmo genético, simulated annealing, e busca tabu.

Em Ozbakir, Baykasoglu e Tapkan (2010) foi desenvolvido um algoritmo de colônia artificial de abelhas para o problema de alocação generalizado (*Generalized Assignment Problem*) um problema similar ao CCP. Os resultados foram comparados com outras heurísticas: busca tabu, algoritmo genético, *path relinking*, colônia de formigas, entre outros. Os autores concluem que o algoritmo desempenha bem em problemas com grande número de pontos e com restrições fortes.

Neste trabalho é proposto desenvolvido um algoritmo de Colônia Artificial de Abelhas para o problema de clusterização capacitado descrito acima. A restrição de capacidade impõe uma dificuldade a mais ao problema resolvido por Zhang *et al.* (2010). O algoritmo foi testado em instâncias de 100, 200, 300 e 400 pontos e os resultados foram comparados com os resultados relatados por Chaves *et al.* (2007).

A estrutura do artigo é a seguinte: a próxima seção se dedica à formalização do modelo matemático que descreve o problema, a terceira seção descreve o algoritmo desenvolvido, a quarta seção corresponde aos testes do algoritmo em instâncias de teste e por último, uma análise comparativa é feita nas conclusões.

2. Formulação do Problema

Seja $I = \{1, 2, 3, ..., n\}$ os índices relativos aos clientes e $J = \{1, 2, 3, ..., p\}$ os índices relativos às medianas. Cada cliente $i \in I$ pode ser caracterizado por sua posição (x_i, y_i) e sua demanda d_i. Cada mediana $j \in J$ pode ser caracterizada pela sua posição (x_j, y_j) e pela sua capacidade b_j (que corresponde à capacidade do cluster). Para cada par de pontos (i,j) define-se c_{ij} como o custo de alocação de i em j. Uma forma de definir o custo de alocação c_{ij} é pela distância euclidiana no plano (x,y) entre os pontos i e j.

Definem-se as seguintes variáveis de decisão:

$$y_j = \begin{cases} 1, \text{se uma mediana está localizada no ponto } j \in J \\ 0, \text{caso contrário} \end{cases}$$

$$x_{ij} = \begin{cases} 1, \text{se o cliente } i \in I \text{ está alocado à mediana } j \in J \\ 0, \text{caso contrário} \end{cases}$$

A formulação clássica do CCP é:

$$\min \sum_{i \in I} \sum_{j \in J} c_{ij} \cdot x_{ij} \tag{1}$$

Sujeito à:

$$\sum_{j \in J} x_{ij} = 1 \quad \forall i \in I \tag{2}$$

$$\sum_{j \in J} y_j = p \tag{3}$$

$$\sum_{i \in I} d_i \cdot x_{ij} \leq b_j \cdot y_j \qquad \forall j \in J \tag{4}$$

$$x_{ij} \in \{0,1\}, y_j \in \{0,1\} \quad \forall i \in I, \ \forall j \in J \tag{5}$$

A função objetivo busca a minimização dos custos de alocação, dado pela soma da distância de cada um dos pontos à sua respectiva mediana. As restrições (2) definem que cada ponto está alocado a uma e somente uma mediana, ou seja, não existe atendimento fracionário das demandas dos pontos. A restrição (3) limita o número de medianas a um valor máximo de p. Já as restrições (4) limitam a capacidade de cada agrupamento de pontos à capacidade da mediana $j \in J$ a qual estão alocados; limita também a alocação de pontos apenas a medianas selecionadas. As restrições (5) definem o domínio das variáveis de decisão x_{ij} e y_j, definidas como variáveis binárias. Para a maioria dos casos práticos, o conjunto de medianas J está contido no conjunto de clientes I, ou seja, $J \subset I$. Assim sendo, soluções para esse problema podem ser definidas em um grafo $s=(S,A)$ $S \subset I$ e $A = \{x_{ij} = 1, i \in I, j \in J\}$ tal que S é o conjunto de medianas selecionadas para representar os *clusters* e A é o mapeamento das alocações.

Essa formulação clássica trata-se de um problema de programação matemática linear inteira. Problemas desse tipo apresentam uma natureza combinatória que dificulta encontrar a solução ótima com eficiência computacional. O problema, conforme demonstrado por Garey e Johnson (1979) é NP-Difícil, o que faz com que métodos heurísticos sejam formas atrativas de busca de soluções boas para o problema de clusterização capacitado.

3. Método de Solução

O método de solução desenvolvido nesse artigo utiliza os conceitos apresentados em Karaboga e Akay (2009) para a construção de um algoritmo de Colônia Artificial de Abelhas (ABC) para o problema de clusterização capacitado. Este procura mimetizar o comportamento de busca de alimento das abelhas.

3.1. O Comportamento de Busca de Alimento das Abelhas:

As abelhas são seres sociais que se organizam e desenvolvem uma inteligência coletiva que aumenta o seu desempenho no meio ambiente em que vivem. Três características dessas colônias são particularmente de interesse no desenvolvimento de algoritmos: auto-organização, adaptação e divisão do trabalho (KARABOGA; AKAY, 2010).

Organizadas em colônias, quando na busca de alimentos, as abelhas possuem três tipos de comportamento: trabalhadoras, exploradoras e oportunistas. As abelhas trabalhadoras são àquelas que efetivamente estão alocadas em alguma fonte de néctar próxima da colmeia; essas abelhas realizam viagens à colônia levando néctar colhido e informações acerca da quantidade de néctar da fonte onde ela está alocada. Essa informação é passada para as outras abelhas num local chamado área de dança, onde as abelhas trabalhadoras realizam movimentos que transmitem a informação sobre a proximidade e quantidade de néctar da fonte a que estão alocadas.

Na área de dança, as abelhas oportunistas assistem à dança das abelhas trabalhadoras e tomam uma decisão sobre qual fonte desejam visitar naquele instante. A decisão das abelhas oportunistas é tomada em função da proximidade e da quantidade de néctar das fontes alocadas às abelhas trabalhadoras. Cada abelha oportunista toma uma decisão e uma vez escolhida uma fonte, elas a visitam e retornam a colmeia com mais néctar, onde aguardam a volta de outras abelhas trabalhadoras para repetirem o processo.

As abelhas exploradoras são àquelas que realizam buscas randômicas nos arredores da colmeia para encontrarem novas fontes de néctar. Quando uma abelha trabalhadora esgota a sua fonte de néctar ela se torna uma abelha exploradora e realiza uma busca randômica ao redor da colmeia e se aloca a uma nova fonte.

Dessa forma, as colônias de abelhas, através da interação entre oportunistas e trabalhadoras desenvolvem uma inteligência coletiva que otimiza a sua busca de alimentos, visto que mais abelhas oportunistas irão para as fontes mais promissoras de néctar. Segue uma ilustração desse comportamento na Figura 1:

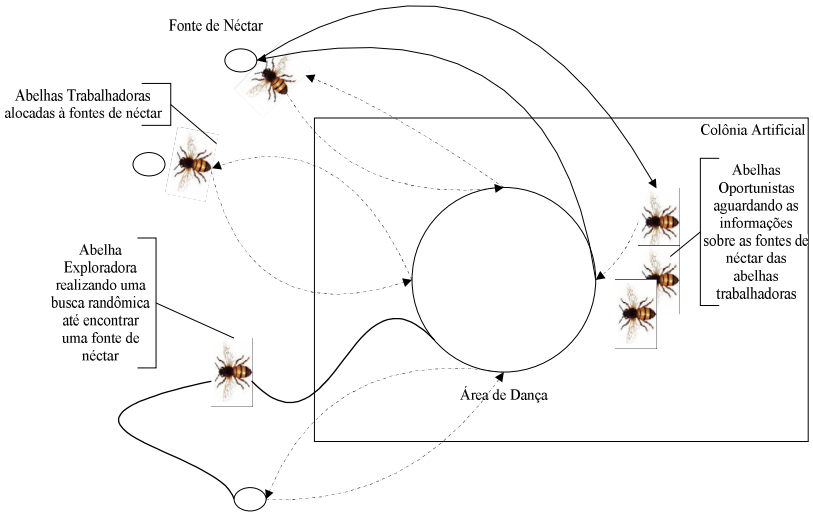

Figura 1 – Ilustração do comportamento de busca de alimentos das abelhas
Fonte: Próprio autor

3.2. Descrição do Algoritmo Baseado em Colônia Artificial de Abelhas

Primeiramente deve ser feita uma distinção entre o conceito de mediana e semente. Mediana é o ponto de um conjunto que minimiza a soma das distâncias entre esse respectivo ponto e os outros pontos do conjunto. Semente é o ponto do conjunto a que os outros pontos estão alocados. Em soluções de elite as medianas e sementes correspondem aos mesmos pontos, todavia, o algoritmo trabalha com o conceito de semente.

No algoritmo desenvolvido a posição de uma fonte de alimentos é representada por uma solução para o problema e a quantidade de néctar dessa fonte representa o valor da função aptidão dessa solução. A posição de uma fonte de alimentos pode ser representada por um vetor $X=\{x_1,x_2,...,x_n\}$, tal que cada valor x_i corresponde à semente a que o ponto i está alocado. No caso de o ponto i ser uma semente, x_i recebe o valor -1.

Nesse caso, a aptidão pode ser considerada igual ao negativo da função objetivo, ou seja:

$$F_{apt} = -F_o = -\sum_{\substack{i \in I \\ i \neq -1}} c_{i,x_i} \tag{6}$$

Em que: F_{apt}: Função Aptidão
F_o: Função Objetivo
$c_{i,xi}$: custo de alocação entre o ponto i e a semente a que ele está alocado x_i.

O algoritmo trabalha com uma população $P=\{1,2,3,...,N\}$ de abelhas, sendo que cada abelha possui uma solução alocada a ela em cada iteração do algoritmo. Assim, cada abelha j é representada por um vetor $X_j=\{x_{1j}, x_{2j},...,x_{nj}\}$, que cada componente x_{ij} corresponde para a abelha j, a semente a que o ponto i está alocado, se o ponto i for definido como semente x_{ij} recebe o valor -1. Cada abelha possui uma lista de sementes independente das sementes das demais Além disso, cada abelha possui um contador interno, $cont_j$, cujo propósito será explicado adiante, um marcador do tipo da abelha e um marcador do valor da função aptidão.

A
Figura 2 ilustra a estrutura de cada solução/abelha.

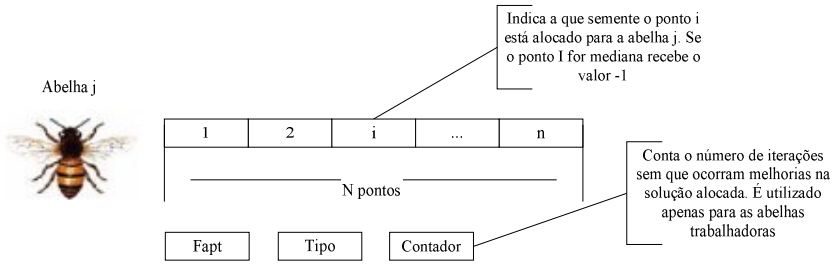

Figura 2 – Representação da estrutura de cada abelha
Fonte: Próprio autor

Como resultado dessa definição, a aptidão da fonte de alimento alocada a abelha j é dada por:

$$F_{apt,j} = F_{o,j} = \sum_{\substack{i \in \exists \\ i \neq 1}} c_{i,x_{ij}} \tag{7}$$

Em que: $F_{apt,j}$: Função aptidão da fonte alocada a abelha j
F_o: Função Objetivo
$c_{i,xij}$: custo de alocação entre o ponto i e a semente a que ele está alocado x_{ij} na solução correspondente à abelha j.

As abelhas por sua vez, apresentam três tipos de comportamento: trabalhadoras, exploradoras, e oportunistas. As abelhas exploradoras realizam buscas randômicas pelo espaço de soluções. Essa busca randômica consiste no sorteio de um ponto semente inicial e alocação das demais sementes segundo uma heurística gulosa. Essa heurística gulosa procura, qual dentre os pontos ainda não definidos como sementes, àquele que maximiza o mínimo custo de alocação entre ele e as sementes já alocadas, e define esse ponto como semente. O processo é repetido até que se tenham p sementes alocadas. Esse procedimento garante que as sementes estarão bem distribuídas pelo espaço de pontos para que se tenha uma alocação de clientes-sementes eficiente. A esse procedimento foi dado o nome de INICIO_SEMENTES, conforme mostrado na Figura 3.

0 – **begin** INICIO_SEMENTES (solução X_j)
1 – gera lista de pontos $L=\{I\}$;
2 – gera lista de sementes $S=\{\varnothing\}$;
3 – gera contador de sementes $cont_s$ e contador de pontos $cont_p$;
4 – sorteia $i \in I$ e aloca primeira mediana: x_{ij}=-1;
5 – tira i da lista de pontos: $L=L-\{i\}$;
6 – coloca i na lista de sementes: $S=S+\{i\}$, $cont_s = 1$;
7 – **while**($cont_s < p$)
8 – Atualiza lista de pontos $L=\{I\}-S$;
9 – **while**($cont_p \leq |I|$)
10 – **select** menor custo de alocação entre um ponto $i \in L$ e uma semente $s \in S$ $c_{is} = \min_{w \in S}\{c_{iw}\}$;
11 – $cont_p = cont_p + 1$;
end while
12 – **select** máxima mínima distância entre um ponto $i' \in L$ e a sua respectiva semente mais próxima $s \in S$: $i' \in L \mid c_{i's} = \max_{w \in L}\{c_{ws}\}$;
13 – faz i' semente: $x_{i'j}$=-1;
14 – atualiza lista de sementes $S=\{k \in I \mid x_{kj} = -1\}$;
end while
15 – **return** $Xj=\{x_{1j}, x_{2j},..., x_{nj}\}$;
16 – **end**

Figura 3 – Procedimento para Início das Sementes (INICIO_SEMENTES)

Em seguida, uma vez alocadas as sementes, as abelhas realizam a alocação dos demais pontos às sementes segundo um outro algoritmo guloso. Esse algoritmo calcula para cada ponto, inclusive para os pontos semente, a diferença entre a segunda mediana mais próxima e a mediana mais próxima. A essa diferença, que representa para cada ponto qual a perda associada à não alocar esse ponto à semente mais próxima, dá-se o nome de arrependimento r. Em seguida os pontos são ordenados de forma decrescente segundo os valores de r. Inicialmente alocam-se as sementes a elas mesmo subtraindo-se a sua demanda de sua capacidade. Para os outros pontos, seguindo o ordenamento feito, tenta-se alocar cada ponto à semente s mais próxima cuja capacidade, b_s, ainda não foi ultrapassada; se houver capacidade disponível para a demanda do ponto, ele é alocado. A esse procedimento foi denominado ALOCA_PONTOS (Figura 4).

Caso para um determinado ponto, não seja possível alocá-lo em nenhuma semente, a solução é abandonada (comando *exit*) e repete-se o procedimento INICIO_SEMENTES para trocar a lista de sementes e tentar uma nova alocação.

Esse procedimento realizado pelas abelhas exploradoras de busca randômica é equivalente aos processos de diversificação de outros algoritmos e heurísticas para fazer com que a busca se desprenda de eventuais ótimos locais.

```
0 – begin ALOCA_PONTOS(solução Xj)
1 –     gera contador de pontos cont_p e marcador de alocação marcador;
2 –     gera lista de sementes: S={k ∈ I | x_kj = -1};
3 –     calcula para cada ponto o valor do arrependimento: r_i = c_ij2-c_ij1;
4 –     sort L={I} decrescente em r_i: L = {i1,i2,i3,...,in} ⊂ I;
5 –     for all i∈L
6 –         gera lista com as medianas para o ponto i: S_i = {k ∈ I | x_kj = -1};
7 –         sort S_i crescente de acordo com c_ij: S_i={s1,s2,...sp} | s_j ∈ S_i é a j-ésima
                semente mais próxima do ponto i;
8 –     end for all
9 –     for all (s∈S)
10 –        b_s = b_s-d_s;
11 –    end for all
12 –    cont_p = 0;
13 –    while (cont_p < | I |)
14 –        if(i_contp ∈ L | x_i,j ≠ -1)
15 –            marcador = 0;
16 –            conts = 0;
17 –            k=1;
18 –            while (marcador ≠ 1 e k < p)
19 –                if(b_sk-d_contp ≥ 0)
20 –                    aloca o ponto i = contp à k-ésima semente
                            mais próxima de i: x_contp,sk=1;
21 –                    b_sk = b_sk-d_contp;
22 –                    marcador=1;
23 –                end if
24 –                k=k+1;
25 –            end while
26 –            if(marcador = 0)
27 –                exit;
28 –            end if
29 –        end if
30 –        cont_p=cont_p+1;
31 –    end while
32 –    return Xj={x1j, x2j,..., xnj};
33 – end
```

Figura 4: Procedimento para Alocação dos Pontos (ALOCA_PONTOS)

As abelhas trabalhadoras por sua vez, já possuem uma solução guardada em suas memória, e o que elas fazem são buscas locais nessa solução. O procedimento de busca ocorre da seguinte forma: sorteia-se uma semente, e sorteia-se também, um ponto alocado a essa semente; esses pontos são trocados, ou seja, o ponto sorteado, alocado à semente sorteada se transforma em semente e efetua-se o procedimento de alocação de pontos, ALOCA_PONTOS. Caso a solução não seja viável, reinicia-se a busca local. Esse procedimento foi denominado BUSCA_LOCAL, mostrado na Figura 5. Trata-se de uma estrutura de vizinhança muito simples, do tipo *swap*. Visto que o número de buscas locais será muito elevado, não se fez necessária estrutura de vizinhança mais elaborada para que o algoritmo apresentasse desempenho satisfatório.

Se a busca local permitir uma melhoria da solução alocada a abelha, essa nova solução passa a ser a da memória da respectiva abelha e o contador interno de cada abelha $cont_j$,que é um

contador de buscas sem melhoria, é zerado, caso contrário mantém-se a solução atual e adiciona-se uma unidade ao contador $cont_j$. Segue uma descrição mais detalhada dessa busca:

```
0 – begin BUSCA_LOCAL (solução X_j)
1 –       gera solução auxiliar X'_j;
2 –       faz X'_j=X_j;
3 –       sorteia i tal que x'_ij=-1;
4 –       sorteia i' tal que x'_i'j=i;
5 –       troca x'_ij e x'_i'j: x'_i'j=-1 e x'_ij=i;
6 –       ALOCA_PONTOS (solução X'_j);
7 –       if (solução X'_j é inviável)
8 –                 cont_j=cont_j+1;
9 –                 exit;
10 –      end if
11 –      if (solução X'_j é viável)
12 –                if (F_apt(X'_j)> F_apt(X_j))
13 –                          faz X_j=X'_j;
14 –                          cont_j=0;
15 –                end if
16 –                else
17 –                          cont_j=cont_j+1;
18 –                end else
19 –      end if
10 –end
```

Figura 5: Procedimento de Busca Local

A Figura 6 ilustra o procedimento de busca local.

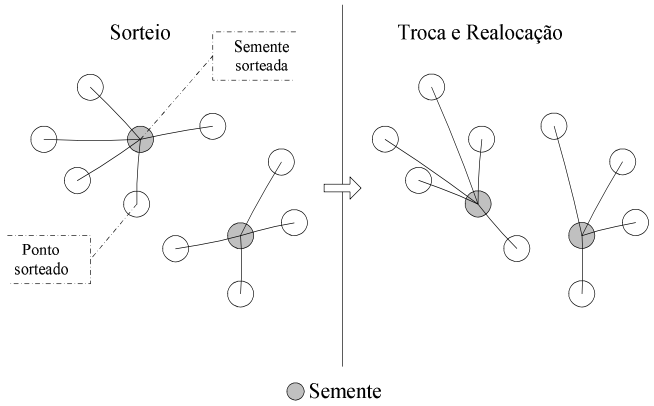

Figura 6 – Ilustração do procedimento de busca
Fonte: Próprio autor

As abelhas trabalhadoras realizam uma busca local em cada iteração do algoritmo, até que seja realizado um número *limite* de buscas sem melhoria da solução na memória da abelha. Caso esse número tenha sido atingido, a abelha trabalhadora se transforma numa abelha exploradora e na próxima iteração do algoritmo ela realizará os procedimentos de busca randômica e alocação de pontos.

Em cada iteração, as abelhas oportunistas precisam tomar uma decisão e escolher uma solução alocada à outra abelha não oportunista, para então realizar uma busca local nessa solução

e eventualmente melhorá-la. Essa decisão é baseada num sorteio tal que a probabilidade de uma determinada solução, alocada a uma abelha trabalhadora, ser escolhida é igual a:

$$P(X_j) = p_j = \frac{F_{apt,j}}{\sum_{k=1}^{N} F_{apt,k}} \quad (8)$$

As variáveis da equação (8) já foram definidas. Cada abelha oportunista realiza uma busca local na solução sorteada e se houver melhoria, essa nova solução fica alocada na memória da abelha, caso contrário, se não houver melhoria, ou se a busca retornar uma solução inviável, a abelha fica com a solução anterior.

O procedimento relativo às abelhas oportunista foi denominado OPORTUNISMO e é mostrado na Figura 7.

```
0 – begin OPORTUNISMO (solução Xj)
1 –     gera solução auxiliar X'j;
2 –     for each (Xj de abelhas não oportunistas)
3 –         pj = Faptj/∑iFapti;
4 –     end for each
5 –     calcula probabilidades acumuladas Pi=∑k≤ipk;
6 –     sorteia número aleatório entre 0 e 1;
7 –     procura abelha i correspondente
8 –     faz X'j=Xi;
9 –     X'j = BUSCA_LOCAL(solução X'j);
10 –    if (solução X'j é inviável)
11 –        exit;
12 –    end if
13 –    if (solução X'j é viável)
14 –        if (Fapt(X'j)> Fapt(Xj))
15 –            faz Xj=X'j;
16 –        end if
17 –    end if
18 –end
```

Figura 7: Procedimento relativo à abelha oportunista (OPORTUNISMO)

A atuação das abelhas oportunistas corresponde aos processos de intensificação da busca em locais promissores do espaço de busca, visto que soluções com maior aptidão têm maiores chances de serem selecionadas e consequentemente de serem mais exploradas por buscas locais, compondo o processo de inteligência coletiva das abelhas.

O processo de interação entre as abelhas define o algoritmo desenvolvido, o ALGORITMO_ABC (Figura 8). Ele consiste na geração da população de abelhas e realização de buscas locais e randômicas conforme as iterações da heurística. O critério de parada adotado foi o de número máximo de iterações *MCN*.

4. Testes Computacionais

Os experimentos computacionais foram efetuados utilizando um computador com processador Intel Dual Core, com velocidade de 1,66GHz e 1Gb de memória RAM, sistema operacional Microsoft Windows XP Professional. O algoritmo foi implementado em linguagem C++ utilizando o pacote . e o *Microsoft Visual C++ 2008 Express Edition*. Os parâmetros utilizados foram: população de 20 abelhas, o número total de ciclos *MCN* de 5000 e um *limite* igual a 100, ou seja, se alguma abelha trabalhadora realizar 100 iterações sem melhoria da

solução corrente alocada à memória dessa abelha, ela se transforma em uma abelha exploradora, realiza uma busca randômica e continua a explorar essa nova solução.

```
0 – begin ALGORITMO_ABC
1 –     gera estrutura de melhor solução para guardar a cada iteração X_b;
2 –     gera população (gera P soluções X_j)
3 –     define as abelhas (abelhas trabalhadoras X_t ∈ T, abelhas oportunistas X_o ∈ O,
        abelhas exploradoras X_e ∈ E={∅}) | (T∩O∩E⊂P)
4 –     for each (X_j ∈ T)
5 –         INICIO_SEMENTES(X_j);
6 –         ALOCA_PONTOS(X_j);
7 –     end for each
8 –     gera contador de iterações cont_it = 0;
9 –     while (cont_it <MCN)
10 –        for each (X_j ∈ T)
11 –            BUSCA_LOCAL (X_j);
12 –            if (cont_j>limite)
13 –                T=T-{X_j}
14 –                E=E+{X_j}
15 –            end if
16 –        end for each
17 –        for each (X_j ∈ O)
18 –            OPORTUNISMO (X_j);
19–         end for each
20 –        for  each (X_j ∈ E)
21 –            marcador=0;
22 –            while (marcador ≠ 1)
23 –                INICIO_SEMENTES (X_j)
24 –                ALOCA_PONTOS (X_j)
25 –                if (X_j é viável)
26 –                    marcador = 1;
27 –                end if
28 –            end while
29 –        end for each
30 –        select X_b | F_apt,b = max_{k∈P}(F_apt,k)
31 –    end while
32 –return X_b e F_apt,b;
33 –end
```

Figura 8: Algoritmo principal de controle das abelhas

O protótipo foi testado em seis instancias de testes: sjc1.dat (100 pontos, 10 medianas), sjc2.dat (200 pontos, 15 medianas), sjc3a.dat (300 pontos, 25 medianas), sjc3b.dat (300 pontos, 30 medianas), sjc4a.dat (402 pontos, 30 medianas) e sjc4b.dat (402 pontos, 40 medianas). Essas instâncias foram propostas por Lorena e Senne (2004) e encontram-se disponíveis em http://www.lac.inpe.br/~lorena/instancias.html. Visto que o algoritmo utiliza números aleatórios, para cada instância foram feitos 5 testes. Os resultados quanto aos valores mínimo e médio da função objetivo e quanto aos tempos médios de processamento são apresentadas a seguir na Tabela 1.

Na coluna de referência, a tabela também os valores obtidos com os encontrados em Boccia *et al.* (2008). No seu trabalho, os autores desenvolveram um algoritmo de planos de corte para o problema e o implementaram utilizando um software comercial de programação inteira em um computador com um processador de 1,6GHz e memória RAM de 728Gb.

Tabela 1 - Resultados dos testes computacionais

Instância	Solução Inicial		Solução Final		Tempo de Process. até a Melhor Solução (s)	Tempo Total de Process. (s)	Pior Resultado	Referência		
	Mín.	Média	Mín.	Média				Mín. de Referência	Desvio	Tempo de Process. de Referência (s)
sjc1.dat	20792	22270	17437	17450	55	103	17469,4	17289	0,85%	38
sjc2.dat	38644	40284	33271	33315	137	275	33352,2	33271	0,00%	128
sjc3a.dat	53806	54935	45623	45794	356	615	45959,9	45335	0,64%	459
sjc3b.dat	47502	48002	40842	40992	529	712	41054,5	40636	0,51%	72
sjc4a.dat	73402	74644	62653	62872	324	978	63043,1	61926	1,17%	1210
sjc4b.dat	61071	61967	52896	52967	1004	1279	53050,3	52458	0,84%	670

Fonte: Próprio autor

Os resultados mostram que o desempenho do algoritmo de colônia de abelhas apresentado é da mesma ordem de grandeza que resultados de outros autores. Os tempos de processamento apresentados são os tempos totais médios entre as 5 vezes que cada instância foi testada sendo que os tempos totais correspondem ao tempo total de processamento até a conclusão de 5000 ciclos, e não até a solução encontrada, o que significa que as melhores soluções foram determinadas em tempos menores apresentados na coluna de tempo de processamento até a melhor solução. De fato, o algoritmo mostra uma convergência bastante rápida. Comparado com os tempos de processamento apresentados em Lorena e Senne (2004), Chaves *et al.* (2007), Scheuerer e Wendolsky (2006), e Boccia *et al* (2008) o algoritmo produz resultados satisfatórios em tempos de processamento comparáveis.

Outro ponto relevante quanto à qualidade das soluções é o aspecto visual dos mapas de alocação determinados pelo algoritmo; a Figura 9 apresenta uma representação pictórica das soluções encontradas para as instâncias sjc2.dat e para a instância sjc4b.dat. O diâmetro dos círculos representa a demanda dos respectivos pontos.

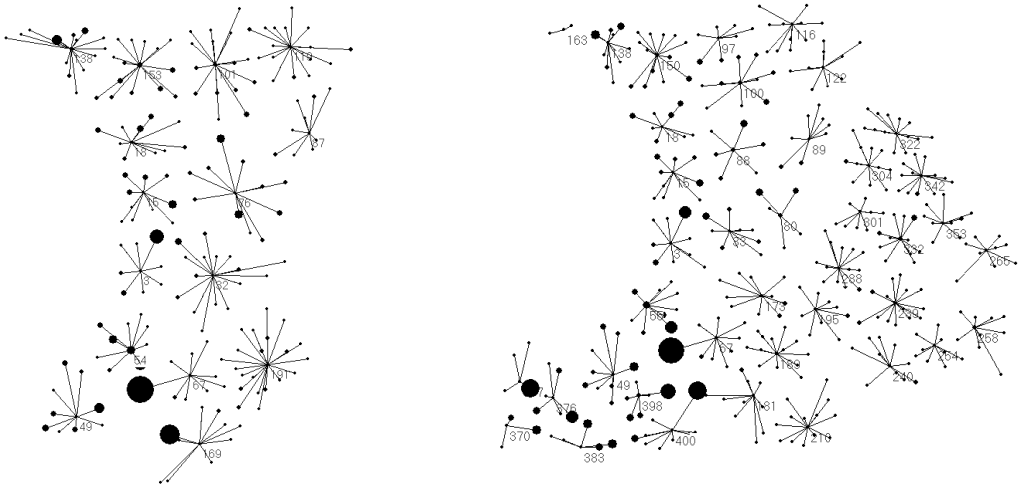

Figura 3 – Representação gráfica da solução encontrada para as instâncias sjc2 e sjc4b.dat
Fonte: Próprio autor

5. Conclusões

O objetivo deste artigo é apresentar um algoritmo de colônia artificial de abelhas para a solução de um problema de clusterização capacitado, e com base em testes computacionais, identificar os pontos fortes e fracos da estratégia baseada em colônia artificial de abelhas. O algoritmo apresenta algumas vantagens interessantes quanto ao seu desenvolvimento. Primeiramente é um algoritmo que necessita de poucos parâmetros, ao contrário de outras abordagens como por exemplo as encontradas em França *et al.* (1999), Fleszar e Hindi (2008), e Diaz e Fernandez (2006); de fato são necessários três parâmetros, o número máximo de ciclos, o tamanho da população e o número máximo de buscas locais sem melhoria para cada abelha trabalhadora. Outro ponto interessante do algoritmo é que não é necessária uma estrutura de vizinhança muito complexa, o que pode ser muito vantajoso em questões de implementação.

De uma maneira geral, os resultados se mostraram muito próximos dos melhores resultados encontrados na literatura, o que comprova a eficácia do algoritmo. Os resultados deste trabalho apontam os algoritmos da classe *Swarm Optimization* como um campo promissor no desenvolvimento de algoritmos e heurísticas.

Uma possibilidade de avanço no trabalho apresentado é quanto à aplicação de estruturas de vizinhança mais complexa que asseguraria uma busca de vizinhança mais minuciosa. O efeito dessa abordagem pode eventualmente se mostrar vantajoso do ponto de vista de esforço computacional, resultando numa necessidade de um número menor de ciclos. Além disso, a combinação do algoritmo clássico ABC, apresentado por Karaboga (2005), com outras heurísticas e meta-heurísticas consagradas poderia ser implementado para comparação dos resultados. Outra possibilidade de continuidade da pesquisa é a aplicação do algoritmo ABC em outros problemas diferentes do CCP para a avaliação de sua eficácia e eficiência.

Referências Bibliográficas

Akay B., Karaboga D. (2010). A modified Artificial Bee Colony algorithm for real-parameter optimization, Information Science, (no prelo).

Boccia M., Sforza A., Sterle C., Vasilyev I. (2008) A Cut and Branch Approach for the p-Median Problem Based on Fenchel Cutting Planes. *Journal of Mathematical Modeling and Algorithms* 7:43-58.

Chaves A.A., Correa F.A., Lorena L.A.N. (2007). Clustering search heuristic for the capacitated p-median problem. *Advances in Software Computing Series* 44: 136–43.

Diaz, J.A., Fernandez, E. (2006). Hybrid scatter search and path relinking for the capacitated p-median problem. *European Journal of Operational Research* 169, 570–585.

França, P.A., Sosa, M.S.,Pureza, V. (1999). An adaptative tabu search algorithm for the Capacitated clustering problem. *International Transactions in Operational Research* 6, 665–678.

Fleszar K., Hindi D.S., (2008). An effective VNS for the capacitated p-median problem. *European Journal of Operational Research* 191(3), 612-622.

Garey, M.R., Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

Jarboui, B., Cheikh, M., Siarry, P., Rebai, A. (2007). Combinatorial particle swarm optimization (CPSO) for partitional clustering problem. *Applied Mathematics and Computation* 192, 337–345.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical report – TR06*, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga, D., Akay, B. (2009) A comparative Study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation* 214, 108–132.

Karaboga, D., Ozturk, C. (2011). A novel clustering approach: Artificial Bee Colony (ABC) algorithm. *Applied Soft Computing* 11, 652–657.

Koskosidis, Y.A., Powell, W.B. (1992). Clustering algorithms for consolidation of customer orders into vehicle shipments. *Transportation Research B* 26, 365-379.

Lorena, L.A.N., Senne, E.L.F. (2004). A column generation approach to capacitated p-median problem. *Computers and Operations Research* 31, 863–876.
Mulvey, J.M., Beck, M.P. (1984). Solving capacitated clustering problems. *European Journal of Operational Research* 18(13), 339-348.
Ozbakir, L., Baykasoglu, A., Tapkan, P. (2010). Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation* 215, 3782–3795.
Scheuerer, S., Wendolsky, R. (2006). A scatter search heuristic for the capacitated clustering problem. *European Journal of Operational Research* 169(2), 533–547.
Shelokar, P. S., Jayaraman, V. K. Kulkarni, B. D. (2004) An ant colony approach for clustering. *Analytica Chimia Acta* 509, 187–195.
Zhang, C., Ouyang, D., Ning, J. (2010). An artificial bee colony approach for clustering. *Expert Systems with Applications* 37, 4761–4767.