Metaheurística inspirada no comportamento das formigas aplicada ao problema de agrupamento

Tatiane M. Pacheco[1], Luciana Brugiolo[1], Victor Ströele[1] e Stênio Sã[1]

Programa de Pós-graduação em Ciência da Computação
Departamento da Ciência Da Computação
Universidade Federal de Juiz de Fora (UFJF) – Juiz de Fora, MG – Brasil
{tatiane.martins, lbrugiolo, victor.stroele, ssoares}@ice.ufjf.br,
http://www.ufjf.br/pgcc/

Resumo Métodos aproximativos são muito utilizados na resolução de problemas computacionais complexos, pois são capazes de produzir resultados significativos em um tempo satisfatório. Sendo o problema de agrupamento automático NP-difícil, métodos não-exatos que possuem uma complexidade tratável são desejáveis. Por essa razão, nesse trabalho é apresentada uma metaheurística de inteligência coletiva, inspirada no comportamento das formigas para resolver problemas de agrupamento de dados. O algoritmo implementado foi submetido a testes, recebendo como entrada 68 instâncias da literatura e os resultados foram comparados com outro método de agrupamento baseado em densidade. O índice silhueta foi adotado para mensurar a qualidade dos agrupamentos gerados. Os resultados obtidos pelo algoritmo proposto foram competitivos quando comparados com a literatura, indicando que o algoritmo é capaz de encontrar agrupamentos que representem bem a distribuição original dos dados.
Palavras-chave: Colônia de formigas, agrupamento, metaheurística.

1 Introdução

Segundo [1] *clustering*, ou agrupamento é a identificação de grupos homogêneos de objetos, que consiste em inserir dentro de um mesmo conjunto, elementos que possuem tanto uma alta similaridade intragrupo quanto uma elevada dissimilaridade extra grupo. Tal problema, pode ser subdividido em duas classes distintas, sendo: Problema de Agrupamento e Problema de Agrupamento Automático, no qual no primeiro é necessário conhecer a *priori* o número de grupos, e no segundo, esse número é estimado pelo próprio algoritmo. Ambos os problemas pertencem à classe NP- difícil, quando $k > 3$ [4]. Por conseguinte, existem muitos trabalhos onde são utilizadas metaheurísticas para a resolução de problemas de agrupamento em um tempo viável [11]. Técnicas, inspiradas na natureza, estudam o comportamento de sistemas descentralizados e auto-organizados, visando imitá-los e aplicá-los em problemas computacionais difíceis na busca por soluções satisfatórias [10]. Os comportamentos coletivos que mais inspiraram o desenvolvimento de algoritmos são os baseados em colônias de formigas[8], abelhas[13] e partículas[14].

As metaheurísticas vêm sendo aplicadas em campos diversos, tais como: aprendizado de máquina, mineração de dados, bioinformática, biologia computacional, finanças, otimização estrutural e topológica em eletrônicos, processamento de imagens e sinais, telecomunicações, roteamento, entre outros [20]. Neste artigo, é proposto um algoritmo de otimização baseado em colônias de formigas (*Ant Colony Optimization* - ACO) fundamentado no trabalho de [2] com modificações para melhora de desempenho.

Este trabalho está organizado nas seguintes seções: 1 introdução, 2 trabalhos relacionados, 3 apresenta a ideia central da otimização por colônia de formigas (ACO), 4 definições sobre essa abordagem são introduzidas formando a base para entendimento do algoritmo, 5 resultados computacionais são apresentados e por fim na seção 6 algumas conclusões são discutidas.

2 Trabalhos relacionados

O algoritmo de otimização por colônias de formigas, foi apresentado por [8] no ano de 1996. Em 1997 outra versão dos mesmos autores foi aplicada ao problema do caixeiro viajante [7]. Já em [3] um algoritmo híbrido é proposto para resolver problemas de *project scheduling*, em que combina uma estratégia de busca local com (ACO). Uma combinação de algoritmo genético com ACO é apresentada em [18], tal técnica híbrida foi utilizada em problemas que envolviam informações espaciais.

Uma aplicação no domínio de agrupamento foi apresentada por [6] na qual uma população de formigas movem-se por uma grade 2D, sendo possível carregar objetos de modo a agrupá-los. Outro caso específico, aplicado a esse domínio foi apresentado em [9], onde é proposto um algoritmo que incorpora formigas adaptativas e heterogêneas, realizando o transporte de forma dependente do tempo e um método que retorna um particionamento explícito. Já em [12] as formigas são utilizadas para produzir os centroides iniciais, para que, posteriormente, possam refinar o agrupamento com o algoritmo Fuzzy C. Em [15] foi apresentada uma nova estratégia, aplicada a análise exploratória de dados, que envolviam memória e o uso de vários tipos de formigas artificiais com diferentes velocidades.

Todas as abordagens inspiradas no comportamento das formigas citadas nessa seção, são baseadas na versão clássica do ACO, em que cada formiga representa uma possível solução para o problema. Em [2] que inspirou essa proposta, entretanto, a solução é formada pelo conjunto de rotas traçadas pelas formigas. Sendo essa gerada através do caminhamento das formigas no digrafo, onde gradativamente arcos menos visitados vão sendo eliminados, por intermédio dos processos de evaporação e reforço. Assim, a solução final é obtida extraindo os componentes fortemente conexos do conjunto de arcos restantes. Nesse trabalho, apresentamos uma adaptação desse algoritmo inserindo conceitos como: formigas com comportamentos diferentes explorando o espaço de soluções, sorteio de empates, trajeto limitado até o alcance da formiga e modificações na função de probabilidade.

Apesar do foco desse trabalho ser otimização por colônia de formigas, vale lembrar que existem diversas outras técnicas além das baseadas em inteligência coletiva, que são capazes de obter boas soluções, tais como: técnicas hierárquicas, particionais, baseadas em densidade, baseadas em modelos e métodos baseados em *grid*. O algoritmo implementado nessa proposta foi submetido a testes de desempenho e seus resultados foram comparados aos obtidos pelo algoritmo GradeBL, proposto em [16], que mescla agrupamento baseado em densidade com técnica de *grid*.

3 Otimização por colônia de formigas

A ideia central do ACO é imitar o comportamento das formigas para resolver problemas de otimização. Inicialmente, as formigas andam aleatoriamente em busca de alimento e a probabilidade de uma determinada formiga escolher um caminho é proporcional à quantidade de feromônio presente. Como essa substância química evapora com o tempo, os caminhos menos visitados perdem uma grande quantidade de feromônio, levando as formigas aos caminhos com maior concentração da substância. Cada formiga possui também certa probabilidade de se perder da trilha mais atrativa, aspecto importante para o processo de descoberta de novas soluções.

O ACO original é um processo iterativo baseado em duas etapas: construção da solução e a atualização do feromônio. No processo construtivo, são combinadas informações do feromônio com a informação heurística específica relacionada. Na atualização, o feromônio é reduzido de forma gradual, evitando a convergência prematura para uma solução local ótima.

Ao aplicar ACO para realizar agrupamento de dados, são criadas formigas artificiais que caminham por um digrafo, que representam o problema a ser resolvido. Assim, os vértices (nós) são os dados a serem agrupados e os arcos os relacionamentos entre eles, que são ponderados pela razão de aceitação entre dois dados (Aceitar que os dados pertençam ao mesmo *cluster*). As formigas caminham por esse digrafo, fortalecendo os arcos por onde passam, depositando neles seu feromônio. O ponto de partida de cada formiga é definido estocasticamente, ao se deparar com dois ou mais caminhos possíveis, ela tem uma maior probabilidade de selecionar o arco com o maior acúmulo de feromônio depositado. No entanto, também, possui probabilidades de seguir por um caminho que não seja o melhor, para evitar os ótimos locais. O processo de reforço e evaporação faz com que os arcos fracos, abaixo de um limiar, sejam removidos, e no fim, os componentes fortemente conectados são calculados. Ou seja, esse conjunto de componentes representa a solução ideal ou quase ideal para o problema.

4 Abordagem Proposta

Nesta seção é apresentado um método baseado no comportamento das formigas, aplicado ao problema de agrupamento de dados, denominado *Ant-Based Clustering*. Esse método foi parcialmente baseado no trabalho de [2], no qual os

autores apresentam uma proposta de ACO para clusterização, denominada *Ant-Cluster*. Proposta na qual incluem estratégias adaptativas, a fim de acelerar o processo de construção da solução. As definições necessárias para o entendimento da abordagem são apresentadas no decorrer da seção. Abaixo o pseudo-código do algoritmo *Ant-Based Clustering*:

```
1  initParametros(alfa, beta, q, q0, Q, p, limiar, iterMax, numFormigas);
2  initFormigas(qtdGulosas, qtdPonderado, qtdRandom);
3  Enquanto (iterAtual < iterMax) faca
4     Para cada formiga k faca
5        Selecione aleatoriamente no inicial para a formiga k;
6         Enquanto (existe vizinho a visitar)
7           Selecione proximo no de acordo com a funcao de probabilidade;
8         Fim Enquanto
9     Fim faca
10        atualizeFeromonio(arcos);
11     Se (iterAtual % 10 == 0) entao
12       Para cada arco pertencente a D faca
13           Se( feromonio no arco (ij) < limiar )
14           removeArco(i, j);
15       Fim faca
16        atualizaParametros();
17     Fim se
18 iterAtual ++;
19 Fim Enquanto;
20 calculeComponentes(D);
```

Tabela 1: Definições

$\tau_{ij}(t)$	Representa o feromônio depositado no arco ij na iteração t.
η_{ij}	Função heurística relacionada ao problema.
α	Efeito da proporção de τ_{ij} no caminho de seleção das formigas.
β	Proporção de η_{ij} no caminho das formigas.
q	Valor sorteado cada vez que a função que calcula a probabilidade for invocada.
$q0$	Valor que delimita as chances de descoberta de novos caminhos.
Q	Constante, para reforço dos arcos por onde as formigas passam.
ρ	Taxa de evaporação do feromônio.
$limiar$	Limiar de corte para eliminação dos arcos ruins.
$iterMax$	Número máximo de iterações.
$numFormigas$	Número de formigas.

O digrafo é formado por um conjunto O de nós e um conjunto A de arcos sendo denotado por $D = (O, A)$. Cada objeto de dado $\in O$ possui um vetor r-dimensional de atributos.

A diferença entre dois objetos de dados O_i e O_j é calculada pela fórmula:

$$diff\,(i,j) = \sqrt{\sum_{k=1}^{r} (o_{ik} - o_{jk})^2}, |i,j = 1,2,3,...n \tag{1}$$

A similaridade entre dois objetos de dados O_i e O_j é obtida pela fórmula:

$$Sim\,(i,j) = 1 - \frac{diff\,(i,j)}{max\,dif} \tag{2}$$

A similaridade média e máxima de O_i, em relação a todos os outros itens de dados, é definida a seguir:

$$mediasim(i) = \frac{1}{n} \sum_{j=1}^{n} Sim\,(i,\ j) \tag{3}$$

$$max\,sim\,(i) = \max_{1 \le j \le n} Sim\,(i\,,j\,) \tag{4}$$

A taxa de aceitação de O_i em relação a O_j é determinada pela equação abaixo e quanto mais similares forem dois nós, maior será. Essa taxa é utilizada como distribuição inicial do feromônio no digrafo.

$$taxaAceitacao\,(i\,,j\,) = Sim\,(i\,,j\,) - \frac{1}{4}\,(mediasim(i\ ,) + max\,sim\,(i)) \tag{5}$$

A função de probabilidade é invocada para guiar a formiga na escolha da trilha que deve seguir. O processo de tomada de decisão da formiga k consiste em definir para qual arco vai mover se. Sendo determinado através de um sorteio, onde um valor q entre [0,1] é definido estocasticamente. Se o q sorteado for menor que a constante $q0$, então a formiga k, move-se para a arco com maior valor na relação feromônio vsinformação heurística, caso contrário, um arco de A com a restrição de pertencer a vizinhança de O_i é sorteado.

Note que no artigo de [2], um sorteio ponderado era realizado onde os arcos com maior feromônio acumulado e uma alta taxa de aceitação possuíam mais chances de serem visitados. Entretanto, diante dos testes computacionais, percebeu-se que a probabilidade de descobertas de novas soluções ficavam prejudicadas, pois tornava o algoritmo excessivamente guloso. Então, a função de probabilidade será definida da seguinte forma:

$$j = \begin{cases} \arg\left\{\max_{\mathrm{u} \in \text{ aos vizinhos } \mathrm{O_i}} \left[\tau_{iu}^{\alpha}\,(t)\,\eta_{iu}^{\beta}\right]\ \right\}, \text{ quando } \mathrm{q} \le \mathrm{q_0} \\ \\ \qquad \text{Caso contrário, seleciona estocasticamente.} \end{cases} \tag{6}$$

A função heurística utilizada, tem por objetivo, refletir o quão bom é para a solução do problema a decisão de seguir por determinado caminho. É calculada no processo de montagem do digrafo e mantém se fixa durante sua execução. Segue:

$$\eta_{ij} = Sim\,(i,j) \tag{7}$$

O processo de atualização do feromônio inclui a evaporação e o reforço ao fim de cada iteração, após todas as formigas caminharem pelo digrafo depositando o feromônio nos arcos por onde passaram, os arcos são atualizados conforme a fórmula a seguir:

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{8}$$

Onde ρ ou coeficiente de evaporação, armazena um valor pertencente ao intervalo [0,1]. O algoritmo possui uma memória, na qual guarda os arcos que foram utilizadas em cada iteração, assim o reforço é aplicado apenas nesses arcos, os demais, são apenas evaporados, a fim de ir eliminando os ruins, provendo assim a solução gradativa do problema. O reforço é definido como segue:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q.Sim\,(i,j)\,,\ Se\ a\ formiga\ k\ passou\ no\ arco\ \ (i,j) \\ \\ 0\ \ caso\ contrário \end{cases} \tag{9}$$

A atualização dos parâmetros α e β é feita adaptativamente. Então a quantidade média de feromônio no digrafo é calculado (10). Desse modo em (11) definimos ψ como o peso de distribuição de feromônio que indica sua disposição no gráfico.

$$\tau_{media} = \frac{\sum_{(i,j)\in A}\ \tau\,(i,j)}{|A|} \tag{10}$$

$$\psi = \frac{1}{|A|}\left[\sum_{(i,j)\in A} \left(\tau_{media} - \tau_{ij}\right)^{2}\right]^{\frac{1}{2}} \tag{11}$$

Usando o peso de distribuição de feromônio, o algoritmo atualiza o valor de α e β da seguinte maneira:

$$\alpha = e^{-\psi} \tag{12}$$

$$\beta = \frac{1}{\alpha} \tag{13}$$

No *Ant-Based Clustering* aqui proposto, são inseridas formigas com comportamentos distintos, sendo eles: random, random Ponderado e guloso. As formigas random seguem apenas caminhos estocásticos. As formigas artificiais que foram inicializadas com o comportamento random ponderado, realizam um sorteio estratificado utilizando o acúmulo de feromônio *vs* informação heurística depositado no arco, para o cálculo da probabilidade de seguir por determinado caminho, os que possuem um maior valor acumulado recebem maiores chances de serem selecionados para compor a rota da formiga. Já as formigas gulosas devem seguir sempre pelo arco com maior concentração de feromônio e informação heurística definida pela fórmula 6. O objetivo dessas alterações é aumentar a diversidade das soluções encontradas, ao mesmo tempo, em que o números

de formigas é reduzido, diminuindo o custo computacional necessário para a construção da solução.

Outra modificação adotada, trata da possibilidade de acontecerem empates no processo de escolha do próximo nó a visitar, uma vez que um ou mais arcos podem possuir o mesmo valor determinado pela função de probabilidade em 6. Sendo assim, todos devem possuir chances iguais de serem selecionados, então um sorteio simples com tais tuplas é realizado nesse caso. Em *Ant-Cluster* original, uma formiga deve visitar todos os nós pertencentes ao digrafo, porém em *Ant-Based Clustering* a formiga k percorre até onde alcançar, visitando os nós que pertencem a sua vizinhança.

5 Resultados Computacionais

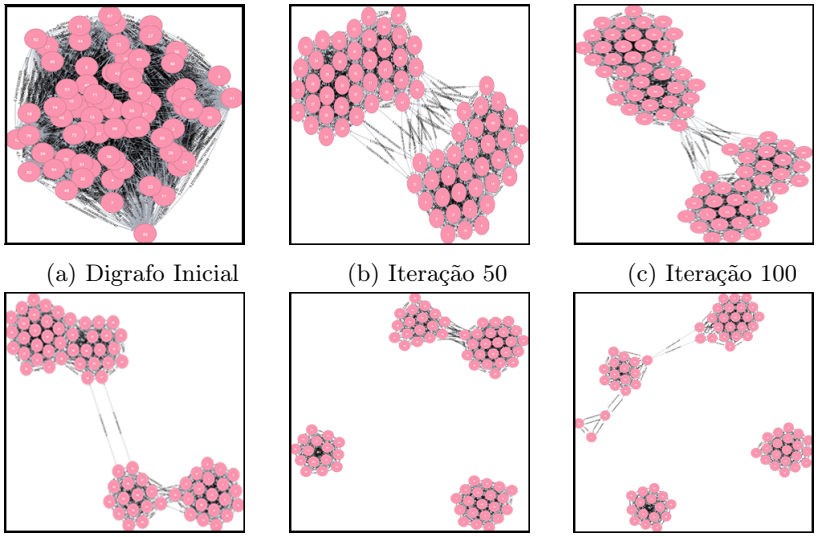
O trabalho apresentado aqui, foi implementado em linguagem Java e executado em um computador dotado de 8GB de memória RAM, processador i5 de 2,53 GHZ, sistema operacional Windows 7 Professional para arquiteturas de 64 *bits*. Foram utilizadas 68 instâncias da literatura para execução dos testes e validação do algoritmo. Tais instâncias foram divididas em 3 grupos, classificados como: comportadas, não comportadas e bases adicionais [16]. O grupo denominado "comportadas", é composto por instâncias onde os *clusters* são bem definidos e separados, já no grupo não comportadas, as instâncias possuem muitos pontos entre os *clusters* [5]. O conjunto de bases adicionais contém 21 instâncias construídas por [5] e por [19] com formatos e densidades variadas. O algoritmo também foi submetido a 6 instâncias bem conhecidas, sendo elas: Iris, Ruspini , Maronna, 200DATA, Vowel, Gauss9, Glass e Soybean(*small*).

Em geral, as técnicas de agrupamento exigem alguma métrica de avaliação dos grupos gerados, no intuito de validar a qualidade da solução encontrada, sendo assim a métrica utilizada aqui foi o índice silhueta. A definição e uma ótima explicação sobre como calcular a silhueta pode ser vista em [17]. Tal métrica gera valores pertencentes ao intervalo [-1,1] para cada objeto de dado, os que apresentam silhueta com valores próximos a 1 estão bem classificados e valores menores que 1 indicam que o objeto está mal classificado e talvez possa pertencer a outro grupo. A silhueta da solução é obtida através da média das somas das silhuetas de todos os objetos.

O teste de validação inicial foi executado no dataset Ruspini e na Figura 1 é possível ver todo o processo executado pelo ACO. Para o exemplo em questão, obtém-se como resposta k = 4 e o cálculo da silhueta gera o valor de **0,738**. Note que conforme a Figura 1f, existem arcos restantes ligando dois componentes distintos, entretanto, tais arcos possuem apenas uma direção e portanto não atendem o conceito de componentes fortemente conexos[1]. Portanto, mesmo existindo alguns arcos o resultado do agrupamento está correto.

Com fins de validação dos agrupamentos gerados, as silhuetas obtidas foram comparadas com o algoritmo denominado GradeBL, proposto por [16]. O Gra-

[1] Os Componentes Fortemente Conectados de um grafo são os conjuntos de vértices mutuamente alcançáveis [21]



(a) Digrafo Inicial (b) Iteração 50 (c) Iteração 100

(d) Iteração 150 (e) Iteração 200 (f) Iteração final

Figura 1: Teste de validação Ruspini dataset

deBL é uma junção de um método de agrupamento baseado em densidade com a técnica de *grid*, em que os autores acrescentam dois procedimentos: identificação de ruídos e deslocamento de grade para identificação de grupos. Testes adicionais foram realizados entre o *Ant-Based Clustering* e o *Ant-Clustering* com o objetivo especifico de avaliar se as modificações inseridas foram eficazes para a redução do número de formigas utilizadas na construção da solução mantendo a qualidade e diminuindo o tempo computacional. O que fica evidente nas Tabelas 2 e 3 onde foram obtidos resultados parecidos com relação a taxa de erros de ambos os algoritmos porém o tempo obtido pelo *Ant-Based Clustering* destaca-se muito em relação ao *Ant-Clustering* obtendo 11,33 e 40,24 segundos respectivamente para o *dataset* Glass. Já para o *dataset* Soybean a diferença foi mais modesta sendo 2,28 para o *Ant-Based Clustering* e 9,33 segundos gastos pelo *Ant-Clustering*.

Glass		
Parâmetros	Ant-Based Cluster	Ant-Cluster
Iterações	230	500
Nº de Clusters	6	6
Max Erros	13	11
Min Erros	9	6
Média de Erros	10,43	7,82
Táxa de Erros	4,87%	3,65%
Custo de Tempo	11,33	40,24

Tabela 2: Glass *dataset*

Soybean (small)		
Parâmetros	Ant-Based Cluster	Ant-Cluster
Iterações	230	500
Nº de Clusters	4	4
Max Erros	2	2
Min Erros	0	0
Média de Erros	0,66	0,78
Táxa de Erros	1,40%	1,66%
Custo de Tempo	2.28	9.33

Tabela 3: Soybean(*small*) *dataset*

Comportadas						
Instâncias	Best	GradeBL	k	ACO	k	GAP
Ruspini	**0,778**	**0,778**	**4**	0,738	**4**	0,040
Iris	**0,686**	-		**0,686**	2	0,000
Maronna	**0,575**	**0,575**	**2**	0,562	**2**	0,013
200Data	**0,847**	**0,847**	**4**	0,823	**3**	0,024
Vowel	**0,369**	**0,369**	**2**	**0,369**	**2**	0,000
Gauss9	**0,257**	0,250	**2**	**0,257**	**2**	0,000
100p3c	**0,808**	**0,808**	**3**	0,786	**3**	0,022
100p7c	**0,842**	**0,842**	7	0,834	7	0,008
100p10c	**0,834**	**0,834**	10	**0,834**	10	0,000
200p4c	**0,790**	**0,790**	4	0,773	4	0,017
300p3c	**0,780**	**0,780**	3	0,766	3	0,014
400p3c	**0,813**	**0,813**	3	0,799	3	0,014
500p3c	**0,827**	**0,827**	3	0,825	3	0,002
600p15c	**0,806**	**0,806**	15	0,781	15	0,025
700p4c	**0,807**	**0,807**	4	0,797	4	0,010
800p23c	**0,793**	**0,793**	23	0,724	20	0,069
900p5c	**0,724**	**0,724**	5	0,716	5	0,008
900p12c	**0,845**	**0,845**	12	0,818	11	0,027
1000p6c	**0,753**	**0,753**	6	0,709	5	0,044
1000p14c	**0,844**	**0,844**	14	0,808	15	0,036
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx						

Tabela 4: Instâncias comportadas

Adicionais						
Instâncias	Best	GradeBL	k	ACO	k	Gap
30p	**0,821**	0,368	4	**0,821**	12	0,000
97p	**0,754**	0,648	4	**0,754**	7	0,000
181p	**0,751**	**0,751**	6	0,700	4	0,051
300p4c	**0,767**	**0,767**	4	0,750	4	0,017
350p5c	**0,771**	**0,771**	5	0,759	5	0,012
450p4c	**0,777**	**0,777**	4	0,766	4	0,011
500p3c	**0,826**	**0,826**	3	0,825	3	0,001
600p3c	**0,759**	**0,759**	3	**0,759**	3	0,000
900p5c	**0,726**	**0,726**	5	0,716	5	0,010
1000p6c	**0,759**	**0,759**	6	0,709	5	0,050
2000p11c	**0,726**	**0,726**	11	0,713	11	0,013
2face	**0,674**	**0,674**	2	0,667	2	0,007
3dens	**0,808**	**0,808**	2	0,762	2	0,046
Convdensity	**0,882**	**0,882**	3	0,854	3	0,028
Convexo	**0,668**	**0,668**	3	**0,668**	3	0,000
Face	**0,097**	**0,097**	2	0,079	2	0,018
Moreshapes	**0,742**	**0,742**	6	0,728	7	0,014
Numbers	**0,572**	**0,572**	9	0,520	10	0,052
Numbers2	**0,612**	**0,612**	10	0,600	10	0,012
Outliers	**0,637**	**0,637**	2	0,634	7	0,003
Outliers_ags	**0,754**	0,665	4	**0,754**	7	0,000

Tabela 5: Instâncias adicionais

Nas Tabelas 5, 4 e 6 são apresentados os resultados comparativos entre o GradeBL e o ACO. Sendo considerados para fins de comparação as melhores silhuetas obtidas por ambos os algoritmos. Nas tabelas citadas, a coluna *best* indica a melhor silhueta para a instância, a coluna *Gap* é dada pela diferença entre a coluna *best* e a silhueta obtida pelo ACO, a maior silhueta encontra-se em negrito e o k indica o número de grupos encontrado por cada algoritmo. Na Tabela 2, onde são apresentados os resultados para as bases comportadas, fica claro que o ACO tem um desempenho satisfatório, obtendo a melhor silhueta para as instâncias 100p10c, Iris, Vowel e Gauss9. Para as demais obteve gap médio de 0,019, gap mediano igual a 0,014, gap mínimo e máximo 0 e 0,069 respectivamente, com desvio padrão de 0,017. Vale notar que o ACO encontrou agrupamentos com silhuetas maiores que 0,709 para 16 das 20 instâncias da tabela 2, e em 6 das instâncias comportadas as silhuetas encontradas foram superiores a 0,807.

O desempenho do ACO nas instâncias adicionais também foi bem satisfatório e o algoritmo obteve a melhor silhueta em 5 das 21 instâncias, com gap médio, mediano, mínimo e máximo de 0,017, 0,012, 0 e 0,052 consecutivamente, com desvio padrão de 0,018. Em 66,66% dos casos as silhuetas encontradas foram superiores a 0,7. Através da imagem 2, por inspeção visual, podemos ver o agrupamento realizado pelo ACO na base face. A base é sintética e os pontos sob o espaço R2, têm o intuito de representar um rosto sorrindo. O ACO foi capaz de agrupar os elementos da face de uma forma bem representativa obtendo um k=6. No qual o olho esquerdo, olho direito, boca, nariz e o contorno do rosto com as orelhas se encontram em grupos distintos, contudo, apesar do ótimo agrupamento realizado pelo algoritmo, que pode ser verificado na imagem, a silhueta encontrada possui valor negativo. Tal comportamento no calculo da silhueta, deve-se ao fato de o índice se comportar bem em agrupamentos compactos e claramente separados e por usar proximidades médias funciona melhor em *clusters* esféricos [17].

Com relação as bases não comportadas, o ACO não obteve um desempenho satisfatório do ponto de vista dos valores de silhuetas encontrados, pois, em apenas 3

Não comportadas						
Instâncias	Best	GradeBL	k	ACO	k	GAP
100p3c1	**0,133**	0,043	2	**0,133**	4	0,000
100p5c1	**0,769**	**0,769**	5	0,729	17	0,040
100p7c1	**0,378**	**0,378**	8	0,326	23	0,052
200p2c1	**0,848**	**0,848**	2	0,749	2	0,099
200p3c1	**0,768**	**0,768**	3	0,648	2	0,120
200p7c1	**0,424**	**0,424**	2	0,310	8	0,114
200p8c1	**0,410**	**0,410**	6	0,306	8	0,104
200p12c1	**0,446**	**0,446**	14	0,321	12	0,125
300p2c1	**0,838**	**0,838**	2	0,758	2	0,080
300p3c1	**0,750**	**0,750**	3	0,690	2	0,060
300p6c1	**0,677**	**0,677**	6	0,577	11	0,100
300p10c1	**0,549**	**0,549**	4	0,514	3	0,035
300p13c1	**0,453**	**0,453**	4	0,449	5	0,004
400p4c1	**0,382**	0,131	2	**0,382**	2	0,000
400p17c1	**0,218**	**0,218**	2	0,193	24	0,025
500p6c1	**0,667**	**0,667**	7	0,557	20	0,110
500p19c1	**0,349**	0,212	2	**0,349**	20	0,000
600p3c1	**0,745**	**0,745**	3	0,661	3	0,084
800p10c1	**0,092**	0,066	2	**0,092**	30	0,000
800p18c1	**0,781**	**0,781**	18	0,628	16	0,153
1000p5c1	**0,712**	**0,712**	5	0,586	11	0,126
1000p27c1	**0,313**	0,169	2	**0,313**	35	0,000
1100p6c1	**0,740**	**0,740**	6	0,618	12	0,122
1500p6c1	**0,701**	**0,701**	6	0,630	10	0,071
2000p9c1	**0,630**	**0,630**	9	0,572	15	0,058

Tabela 6: Instâncias não comportadas

instâncias esse valor foi superior à 0,7. Contudo o seu desempenho em relação ao GradeBL não foi discrepante, uma vez que tal algoritmo obteve silhuetas superiores á esse valor em apenas 8 das 25 instâncias os *gaps* médio (0,067), mediano (0,071), mínimo (0,000) e máximo (0,153) continuaram equilibrados e sem valores demasiadamente divergentes e o desvio padrão foi de apenas 0,048. Além disso o ACO também foi capaz de obter a melhor silhueta em 5 dos casos.

Um fato interessante com relação ao comportamento do ACO nas bases da Tabela 6 pode ser visto na Figura 3, onde fica claro novamente que apesar dos valores de silhuetas não serem tão bons, o ACO é capaz de encontrar agrupamentos que representam bem a distribuição original dos dados. Como dito anteriormente as instâncias não comportadas possuem muitos pontos entre os *clusters* e o ACO trata esses pontos como *outliers*, identificando-os e considerando-os como grupos *singletons*, mas mantém os pontos que estão em *clusters* bem definidos no mesmo componente fortemente conexo.

6 Conclusões e Trabalhos Futuros

O algoritmo *Ant-Based Clustering* apresentado nesse trabalho, obteve bons resultados no que diz respeito as silhuetas obtidas para as instâncias comportadas e adicionais, obtendo agrupamentos que representam significativamente a estrutura dos dados originais, vide as silhuetas das Tabelas 4 e 5. Contudo para as instâncias não comportadas, faz-se necessário, uma etapa adicional, com a finalidade de refinar o agrupamento gerado, haja visto que as silhuetas encontradas não foram tão satisfatórias. Vale salientar, que o algoritmo utilizado nessa comparação, o GradeBL, passou por uma etapa adicional de remoção de ruídos, fato este que colaborou para a melhora das silhuetas

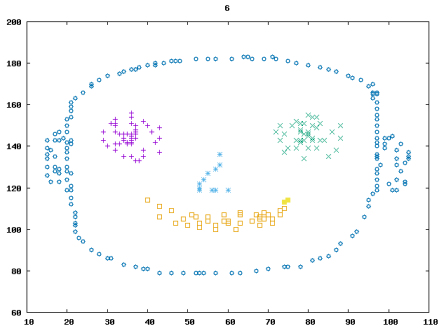


Figura 2: Face

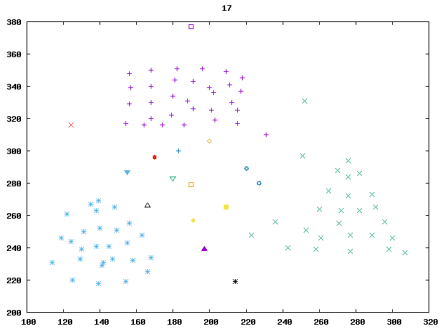


Figura 3: Base 100p3c1

encontradas, enquanto que o ACO foi implementado apenas utilizando a ideia central de clusterização por colônia de formigas, não sendo utilizada qualquer etapa de limpeza adicional, assim o desempenho do ACO mostrou-se promissor e há a expectativa de obtenção de melhores resultados com a implementação do refinamento citado.

Outro ponto importante é a questão do índice silhueta, conforme foi mencionado acima, tal métrica comporta se melhor em dados bem separados. Fica claro na Figura 2, que um ótimo particionamento de dados foi obtido, porém o valor de silhueta foi negativo, indicando que possivelmente a utilização de outros índices para analise de agrupamentos, isolada ou concomitantemente a silhueta, possam avaliar melhor o desempenho do ACO. Um estudo dos possíveis índices a serem adotados há de ser realizado, buscando selecionar os que forem mais adequados aos tipos de instâncias utilizadas nos testes de agrupamento.

o *Ant-Based Clustering* foi capaz de gerar agrupamentos expressivos para todas as instâncias e seu desempenho geral foi dentro do esperado, mostrando indícios que tende a melhorar com a implementação de alguns aprimoramentos. A intenção dos autores para trabalhos futuros é implementar uma busca local multi-objetivo responsável por realizar um refinamento nos componentes fortemente conexos definidos pelo ACO bem como utilizar um calibrador de parâmetros para definir a melhor faixa de valores para suas variáveis, que até o momento são obtidos empiricamente.

Referências

1. Arabie, P., Hubert, L.J., De Soete, G.: Clustering and classification. World Scientific (1996)
2. Chen, L., Tu, L., Chen, H.J.: Data clustering by ant colony on a digraph. In: Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on. vol. 3, pp. 1686–1692. IEEE (2005)
3. Chen, W., Shi, Y.j., Teng, H.f., Lan, X.p., Hu, L.c.: An efficient hybrid algorithm for resource-constrained project scheduling. Information Sciences 180(6), 1031–1039 (2010)
4. Cowgill, M.C., Harvey, R.J., Watson, L.T.: A genetic algorithm approach to cluster analysis. Computers & Mathematics with Applications 37(7), 99–108 (1999)
5. Cruz, M.D.: O problema de clusterização automática. Ph.D. thesis, Universidade Federal do Rio de Janeiro (2010)

6. Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Chrétien, L.: The dynamics of collective sorting robot-like ants and ant-like robots. In: Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats. pp. 356–363 (1991)
7. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation 1(1), 53–66 (1997)
8. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26(1), 29–41 (1996)
9. Handl, J., Knowles, J., Dorigo, M.: Ant-based clustering and topographic mapping. Artificial life 12(1), 35–62 (2006)
10. Handl, J., Meyer, B.: Ant-based and swarm-based clustering. Swarm Intelligence 1(2), 95–113 (2007)
11. Hruschka, E.R., Campello, R.J., Freitas, A.A., et al.: A survey of evolutionary algorithms for clustering. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 39(2), 133–155 (2009)
12. Kanade, P.M., Hall, L.O.: Fuzzy ants as a clustering concept
13. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer engineering department (2005)
14. Kennedy, J.: Stereotyping: Improving particle swarm performance with cluster analysis. In: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on. vol. 2, pp. 1507–1512. IEEE (2000)
15. RAMOS, V., MUGE, F., PINA, P.: Self-organized data and image retrieval as a consequence of inter-dynamic synergistic relationships in artificial ant colonies (2002)
16. Rodrigues, W.C., Fadel, A.C., Semaan, G.S., de Moura Brito, J.A.: Um novo método baseado em grade e densidade com tratamento de ruídos para a identificação do número ideal de grupos
17. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics 20, 53–65 (1987)
18. Schockaert, S., Smart, P.D., Twaroch, F.A.: Generating approximate region boundaries from heterogeneous spatial information: An evolutionary approach. Information Sciences 181(2), 257–283 (2011)
19. Soares, S., Ochi, L.S.: Um algoritmo evolutivo com reconexão de caminhos para o problema de clusterização automática
20. Talbi, E.G.: Metaheuristics: from design to implementation, vol. 74. John Wiley & Sons (2009)
21. Ziviani, N., et al.: Projeto de algoritmos: com implementações em Pascal e C, vol. 2. Thomson (2004)