UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

DANIELA SCHERER DOS SANTOS

# *Bee clustering*: um Algoritmo para Agrupamento de Dados Inspirado em Inteligência de Enxames

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Profa. Dra. Ana L. C. Bazzan
Orientador

Porto Alegre, abril de 2009

**CIP – CATALOGAÇÃO NA PUBLICAÇÃO**

dos Santos, Daniela Scherer

*Bee clustering*: um Algoritmo para Agrupamento de Dados Inspirado em Inteligência de Enxames / Daniela Scherer dos Santos. – Porto Alegre: PPGC da UFRGS, 2009.

89 p.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2009. Orientador: Ana L. C. Bazzan.

1. Inteligência Artificial, Sistemas Multiagente, Inteligência de Enxames, Agrupamento Distribuído de Dados, Organização de Colônias de Abelhas. I. Bazzan, Ana L. C.. II. Título.

# SUMÁRIO

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| $A^2CA$ | *Adaptive Ant-Clustering Algorithm* |
| ACA | *Ant Clustering Algorithm* |
| ACO | *Ant Colony Optimization* |
| ACOC | *Ant Colony Optimization for Clustering* |
| CSIM | *Clustering Algorithm based on Swarm Intelligence and K-means* |
| I-PACE | *Improved Probabilistic Ant based Clustering for Distributed Databases* |
| MACC | *A Multiagent, Multiobjective Clustering Algorithm* |
| MOCLE | *Multi-objective Clustering Ensemble* |
| SACA | *Standard Ant Clustering Algorithm* |

# LISTA DE SÍMBOLOS

| | |
|---|---|
| $\beta$ | Gravidade da tarefa |
| $\theta$ | Limiar |
| $\Delta$ | Intervalo de tempo |
| $\mu$ | Parâmetro que controla o número de iterações do algoritmo |
| $\mathscr{A}$ | Conjunto de agentes/dados |
| $\lvert\mathscr{A}\rvert$ | Tamanho do conjunto $\mathscr{A}$ |
| $c$ | Centróide de um grupo |
| $C$ | Classe de um objeto |
| $D$ | Número de agentes dançando |
| $e(C,k)$ | Eficiência entre a classe $C$ e o grupo $k$ |
| $F$ | *F-measure* |
| $k$ | Grupo |
| $K$ | Número de grupos |
| $l$ | Número de objetos da classe $C$ dentro do grupo $k$ |
| $m$ | Número de objetos na classe $C$ |
| $n$ | Número de objetos no grupo $k$ |
| $p(C,k)$ | Precisão entre a classe $C$ e o grupo $k$ |
| $Pa$ | Probabilidade de abandonar um agente |
| $Pd$ | Probabilidade de dançar |
| $Pv$ | Probabilidade de visitar um agente |
| $R$ | Índice *Rand* |
| $\mathscr{S} = \{d, v, w\}$ | Conjunto de estados de um agente ($d$ = dançando; $v$ = visitando; $w$ = assistindo uma dança) |
| $S$ | Estímulo |
| $t$ | Tempo |
| $T$ | Tarefa |

| | |
|---|---|
| $U$ | Utilidade de um grupo |
| $var$ | Variância intra-*cluster* |
| $V$ | Agrupamento |
| $\mathscr{X}$ | Conjunto de atributos dos agentes/dados |
| $\|\mathscr{X}\|$ | Número de atributos do agente/dado |
| $Z$ | Estrutura conhecida de um conjunto de dados |

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

Agrupamento de dados é o processo que consiste em dividir um conjunto de dados em grupos de forma que dados semelhantes entre si permaneçam no mesmo grupo enquanto que dados dissimilares sejam alocados em grupos diferentes. Técnicas tradicionais de agrupamento de dados têm sido usualmente desenvolvidas de maneira centralizada dependendo assim de estruturas que devem ser acessadas e modificadas a cada passo do processo de agrupamento. Além disso, os resultados gerados por tais métodos são dependentes de informações que devem ser fornecidas *a priori* como por exemplo número de grupos, tamanho do grupo ou densidade mínima/máxima permitida para o grupo. O presente trabalho visa propor o *bee clustering*, um algoritmo distribuído inspirado principalmente em técnicas de inteligência de enxames como organização de colônias de abelhas e alocação de tarefas em insetos sociais, desenvolvido com o objetivo de resolver o problema de agrupamento de dados sem a necessidade de pistas sobre o resultado desejado ou inicialização de parâmetros complexos. O *bee clustering* é capaz de formar grupos de agentes de maneira distribuída, uma necessidade típica em cenários de sistemas multiagente que exijam capacidade de auto-organização sem controle centralizado. Os resultados obtidos mostram que é possível atingir resultados comparáveis as abordagens centralizadas.

**Palavras-chave:** Inteligência Artificial, Sistemas Multiagente, Inteligência de Enxames, Agrupamento Distribuído de Dados, Organização de Colônias de Abelhas.

**Bee Clustering: a clustering algorithm inspired by swarm intelligence**

# ABSTRACT

Clustering can be defined as a set of techniques that separate a data set into groups of similar objects. Data items within the same group are more similar than objects of different groups. Traditional clustering methods have been usually developed in a centralized fashion. One reason for this is that this form of clustering relies on data structures that must be accessed and modified at each step of the clustering process. Another issue with classical clustering methods is that they need some hints about the target clustering. These hints include for example the number of clusters, the expected cluster size, or the minimum density of clusters. In this work we propose a clustering algorithm that is inspired by swarm intelligence techniques such as the organization of bee colonies and task allocation among social insects. Our proposed algorithm is developed in a decentralized fashion without any initial information about number of classes, number of partitions, and size of partition, and without the need of complex parameters. The bee clustering algorithm is able to form groups of agents in a distributed way, a typical necessity in multiagent scenarios that require self-organization without central control. The performance of our algorithm shows that it is possible to achieve results that are comparable to those from centralized approaches.

**Keywords:** Artificial Intelligence, Multiagent Systems, Swarm Intelligence, Distributed Clustering, Bee Colony Organization.

# 1 INTRODUÇÃO

O processo de agrupar objetos, pessoas, animais, entre outros, é realizado pelos seres humanos em seu dia a dia de acordo com seus próprios esquemas e critérios, geralmente, com o objetivo de facilitar a realização de alguma tarefa futura que exija o manuseio de tais itens agrupados.

Agrupar dados consiste em separá-los em grupos homogêneos de forma que objetos que pertencem ao mesmo grupo sejam mais similares entre si do que objetos que pertencem a grupos diferentes. A necessidade deste agrupamento surge de forma natural em muitos campos de estudos como Biologia, Ciências Sociais, Medicina, entre outros (ANDERBERG, 1973). Como exemplos pode-se citar: um problema que ocorre regularmente na área de ciências sociais onde deve-se agrupar dados provenientes de questionários de uma pesquisa para se obter o perfil sócio-econômico e político de uma população; identificação de espécies animais e vegetais é um exemplo encontrado na disciplina de Biologia; em bioinformática na análise de dados de expressão gênica onde é necessária a identificação de padrões entre milhares de genes na tentativa de elucidar as funções que estes desempenham no organismo.

As técnicas de agrupamento de dados normalmente são dependentes de domínio, ou seja, dificilmente se encontrará uma técnica genérica, aplicável de maneira satisfatória a todos os tipos de dados e em todos os contextos. Por este motivo, existem diversas técnicas e estas são de grande utilidade em diversas áreas.

Recentemente, muitos trabalhos têm sido desenvolvidos destacando a eficiência de abordagens estocásticas baseadas em colônias de insetos sociais (formigas, abelhas, vespas e cupins) para resolver problemas complexos. Estes problemas envolvem, por exemplo, a otimização combinatorial como o problema do caixeiro viajante (DORIGO; GAMBARDELLA, 1997), o problema de atribuição quadrática (GAMBARDELLA; TAILLARD; DORIGO, 1998), o problema de roteamento (CARO; DORIGO, 1997), entre outros.

Inteligência de enxames (*Swarm Intelligence*) é a denominação aplicada a tentativa de desenvolvimento de algoritmos ou instrumentos para solução distribuída de problemas inspirando-se no comportamento coletivo de colônias de insetos sociais e outras sociedades de animais (BONABEAU; THERAULAZ; DORIGO, 1999). Algoritmos inspirados em insetos sociais são caracterizados pela interação entre um grande número de agentes que percebem e modificam seu ambiente localmente.

Agrupamento de dados é um dos problemas no qual a área de inteligência de enxames pode sugerir heurísticas interessantes. Diversos algoritmos para agrupamento de dados inspirados em insetos sociais têm sido propostos com o objetivo de melhorar os resultados obtidos com as técnicas tradicionais.

No presente trabalho apresenta-se o *bee clustering*, um algoritmo para agrupamento de

dados inspirado principalmente em técnicas de inteligência de enxames como organização de colônias de abelhas e alocação de tarefas em insetos sociais. O algoritmo proposto inspira-se no comportamento das abelhas para formar grupos de agentes que representam dados a serem agrupados. Tal comportamento é conhecido como recrutamento. Na natureza, as abelhas viajam para longe da colméia com o objetivo de coletar néctar. Quando este é encontrado, elas retornam para a colméia não só com o néctar coletado mas também com informações sobre a fonte onde o encontraram. De posse dessas informações as abelhas iniciam o processo de recrutamento de outras abelhas para coletarem alimento nesta mesma fonte de néctar. Este recrutamento é realizado através de uma dança, conhecida como dança do recrutamento, um comportamento no qual uma abelha comunica a direção, distância e qualidade de uma fonte de néctar para outras abelhas (CAMAZINE; SNEYD, 1991).

Esta dança do recrutamento é utilizada pelo *bee clustering* para formar grupos de agentes. Para isso, cada agente comporta-se como uma abelha e representa um objeto a ser agrupado. Neste algoritmo as abelhas dançam para recrutar novos indivíduos para participarem de seus grupos. Através deste comportamento, os agentes do algoritmo são capazes de se organizar em grupos de acordo com suas características (ou do objeto que representa) ou habilidades e de maneira distribuída. Isto constitue-se um fator importante em sistemas multiagente onde os agentes não possuem conhecimento global sobre todo o ambiente e precisam executar atividades sem qualquer tipo de controle centralizado.

Durante o agrupamento os agentes precisam tomar decisões sobre participar ou não de um grupo segundo o grau de utilidade deste agente em relação ao seu próprio grupo. Para ajudar os agentes neste processo de tomada de decisão encontrou-se inspiração no trabalho apresentado em Agogino e Tumer (2006), onde os autores desenvolveram um método de *ensemble* de agrupamento baseado em agentes através da computação distribuída de uma utilidade global de um agrupamento. No *bee clustering* utilizou-se o cálculo de uma utilidade que representa a qualidade de um grupo, para que o agente pudesse decidir se troca ou não de grupo.

Baseando-se nestes conceitos, o algoritmo *bee clustering* é capaz de formar grupos de agentes com características, habilidades ou especialidades similares ou complementares.

A seguir descreve-se os principais aspectos que motivaram o desenvolvimento da presente pesquisa, bem como os objetivos atingidos.

## 1.1 Motivação e objetivos

Devido a grande quantidade de dados, os quais precisam ser manipulados e processados por meio de computadores, o problema de agrupamento de dados pode ser visto como um problema de otimização apresentando uma complexidade de ordem exponencial. Métodos de força bruta, como enumerar todos os possíveis grupos e escolher a melhor configuração são simplesmente inviáveis.

O número de maneiras de se combinar $n$ objetos de um conjunto de dados que se deseja agrupar em $k$ grupos é um número *Stirling* de segunda ordem [1]. Este número pode ser gerado pela definição recursiva (ANDERBERG, 1973):

$S(n,k) = 1$, para $k > 0$ e $n = k$

$S(n,0) = 0$, para $n \geq 0$

$S(n,k) = S(n-1,k-1) + kS(n-1,k)$, para $0 < k < n$

[1] trata-se do número de maneiras de se particionar um conjunto com $n$ elementos em $k$ subconjuntos disjuntos e não-vazios.

Com o intuito de ilustrar o crescimento exponencial do número de soluções possíveis para um problema de agrupamento, pode-se exemplificar três objetos *a, b* e *c*. Ao agrupar-se este conjunto em apenas um grupo tem-se $\{abc\}$; em dois grupos obtem-se $\{a,\ bc\}, \{ab,\ c\}$ e $\{ac,\ b\}$; e em três gupos tem-se $\{a,\ b,\ c\}$ (ANDERBERG, 1973). Mas se, por exemplo, possui-se 25 objetos os quais deseja-se dividir em 5 grupos, o número *Stirling* de segunda ordem neste caso é:

$$S_{25}^{(5)} = 2.436.684.974.110.751.$$

Se considerarmos que o número de grupos é desconhecido, o número de possibilidades é a soma dos números *Stirling* de segunda ordem. Então, no caso de 25 objetos:

$$\sum_{j=1}^{j=25} S_{25}^{j} > 4 * 10^{18}$$

Fica claro uma enorme quantidade de grupos que não podem ser computados em um tempo aceitável, mesmo para pequenas bases de dados.

No processo de agrupamento, a busca pela melhor solução no espaço de soluções viáveis é um problema NP-Difícil. Diante de uma grande quantidade de possíveis caminhos para agrupar dados, é natural a existência de um grande número de técnicas para agrupamentos. No entanto, métodos tradicionais têm sido desenvolvidos seguindo uma abordagem centralizada, necessitando que os dados estejam localizados em um único local, o que inviabiliza sua aplicação em contextos como, por exemplo, a Internet, onde os dados encontram-se distribuídos por razões de tamanho, privacidade, dinamicidade, entre outros. Além disso, técnicas centralizadas dependem de estruturas de dados que precisam ser acessadas e modificadas em cada passo da operação de agrupamento, criando assim um ponto único de falha no algoritmo.

Outro inconveniente que pode ser visto em métodos clássicos de agrupamento de dados é que muitos destes algoritmos necessitam de algumas pistas sobre o agrupamento desejado, como por exemplo, o número de grupos que deverão ser gerados, o tamanho esperado para cada grupo ou mesmo a densidade máxima ou mínima dos grupos.

A motivação deste trabalho para o desenvolvimento de um algoritmo distribuído para agrupamento de dados vem também de aplicações relacionadas a sistemas multiagente. Nestes sistemas, agentes que desejam cooperar para realizar uma tarefa qualquer devem ter um meio de encontrar grupos de outros agentes de acordo com algum critério para executar tal tarefa de maneira eficiente. Por exemplo, em um cenário complexo composto por muitos agentes com diferentes especialidades e habilidades, estes agentes devem, de maneira distribuída, formar grupos entre si para realizar suas atividades em times com o objetivo de atingir o melhor resultado possível proveniente desta organização.

Agrupamento de agentes baseando-se em características e habilidades similares, ou mesmo complementares, pode ser visto como um problema de agrupamento de dados e constitui-se o foco deste trabalho.

O principal objetivo do presente trabalho é a apresentação do algoritmo *bee clustering* para resolver o problema de agrupamento de dados de uma maneira descentralizada e sem qualquer informação inicial relacionada ao resultado desejado ou mesmo a necessidade do uso de parâmetros complexos.

## 1.2 Organização dos capítulos

Os capítulos deste trabalho estão dispostos da seguinte maneira: no Capítulo 2 é abordado o problema de agrupamento de dados. São descritos os tipos de atributos e índices de similaridade utilizados no processo, bem como os métodos de agrupamento existentes. Já no Capítulo 3 apresenta-se uma breve introdução sobre inteligência de enxames destacando-se alguns conceitos que são necessários para o entendimento da importância de sua aplicação na solução de problemas distribuídos, descreve-se os principais comportamentos utilizados no desenvolvimento de algoritmos de agrupamento de dados inspirados em insetos sociais e detalha-se o comportamento de interesse do presente trabalho encontrado nas colônias de abelhas.

O Capítulo 4 é dedicado às abordagens para solucionar o problema de agrupamento que estão relacionadas ao algoritmo proposto. Estas abordagens encontram-se divididas em: algoritmos inspirados em inteligência de enxames, algoritmos distribuídos e outros algoritmos relacionados. No final deste capítulo apresenta-se uma análise comparativa destes algoritmos, realizada de acordo com um critério determinado.

No Capítulo 5 descreve-se o algoritmo *bee clustering* proposto neste trabalho. Faz-se uma descrição sobre as decisões que os agentes devem tomar durante o processo de execução do algoritmo e as soluções encontradas para apoiar estas tomadas de decisões. O Capítulo 6 apresenta a validação do método bem como os resultados obtidos após dois tipos distintos de aplicação: uma utilizando-se bases de dados de domínio público, das quais se conhece seu correto agrupamento e outra utilizando-se o ambiente de simulação *RoboCup Rescue* que se traduz parcialmente no problema de agrupar agentes que devem operar de forma colaborativa.

Finalmente, o Capítulo 7 apresenta as conclusões do presente trabalho e aponta possíveis direções de pesquisas futuras relacionadas ao algoritmo proposto.

# 2 AGRUPAMENTO DE DADOS

Agrupamento de dados é o nome dado ao conjunto de técnicas que têm por objetivo separar objetos em grupos. Esta separação deve ocorrer de forma que os elementos de um conjunto sejam mutuamente similares e, preferencialmente, muito diferentes dos elementos dos outros conjuntos (BERKHIN, 2002; ANDERBERG, 1973).

As técnicas de agrupamento de dados podem ser vistas como um meio de aprendizado não supervisionado (WITOLD, 2005; JAIN; MURTY; FLYNN, 1999; BERKHIN, 2002) servindo para extrair características escondidas dos dados e desenvolver hipóteses a respeito de sua natureza. Neste último caso, tais métodos podem ser particularmente valiosos em casos onde o resultado final seja inesperado, ou seja, algo em que nunca se tenha pensado (ANDERBERG, 1973).

Jain, Murty e Flynn (1999) apontam alguns passos para realizar a tarefa de agrupamento de dados:

1. Modelo de representação: opcionalmente inclui extração e/ou seleção de características;
2. Definição de um modelo de medida de similaridade/dissimilaridade apropriado ao domínio dos dados;
3. Agrupamento;
4. Abstração dos dados (se necessário);
5. Atribuição de saída (se necessário).

A seqüência dos três primeiros passos encontra-se ilustrada na figura 2.1. A primeira etapa inclui a definição dos atributos a serem levados em consideração no processo de agrupamento dos dados. É neste momento que se realiza o pré-processamento dos dados, como por exemplo normalizações, e se define a forma de representação destes dados, a qual, na maioria dos casos, é realizada por uma matriz de objetos. Os tipos de atributos existentes são descritos a seguir na subseção 2.1.

O próximo passo envolve a definição de um modelo de similaridade/dissimilaridade. Trata-se de uma métrica usada para quantificar a similaridade entre os dados analisados. Este modelo deve ser definido de acordo com o tipo de atributo escolhido na etapa anterior e também será discutido na subseção 2.1.

A etapa seguinte refere-se à aplicação de um algoritmo de agrupamento apropriado para o conjunto de dados. Existem diversos algoritmos que podem ser aplicados nesta etapa. Os algoritmos de agrupamento de interesse para este trabalho encontram-se descritos no Capítulo 4.

Em seguida, tem-se as etapas de abstração dos dados e atribuição de saída. Abstração de dados é um processo de extração da representação compacta e simples de um conjunto de dados. Simplificando-se, trata-se de uma descrição compacta de cada grupo formado. Esta etapa é útil em tomadas de decisão pois descreve de maneira simples e intuitiva os grupos tornando fácil a compreensão humana.

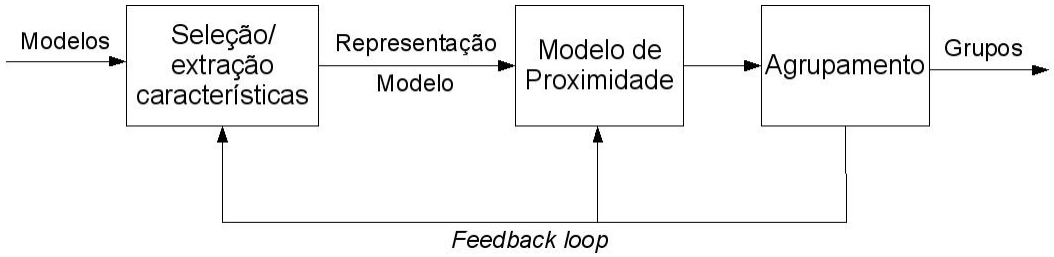


Figura 2.1: Passos para o agrupamento de dados (JAIN; MURTY; FLYNN, 1999)

## 2.1 Tipos de atributos e índices de similaridade

Para agrupar os dados de um conjunto é necessário encontrar algum ou alguns parâmetros que quantifiquem o grau de associatividade entre eles. Esta tarefa é realizada na primeira etapa do processo de agrupamento de dados, onde se define os atributos que serão utilizados para realizar o agrupamento. Para isso, é necessário um profundo conhecimento dos tipos de dados que serão trabalhados.

Definidos os atributos, é necessário também escolher as medidas de similaridades que poderão ser utilizadas para a verificação do grau de semelhança existente entre eles. Esta escolha está diretamente associada ao tipo de atributo com o qual se está trabalhando.

A seguir apresenta-se uma classificação para os tipos de atributos proposta por Anderberg (1973) e alguns possíveis métodos para o cálculo dos índices de similaridade.

### 2.1.1 Atributos contínuos

Os atributos contínuos podem assumir qualquer valor em $\Re$ (WITOLD, 2005). Como exemplos pode-se citar distância, peso, altura, quantidade, temperatura, entre outros. Cada objeto é representado por um vetor de dimensão igual ao número de atributos $n$ a serem considerados e cada elemento do vetor expressa numericamente a medida de cada atributo.

Para atributos contínuos, a métrica mais popular para se calcular a similaridade ou dissimilaridade é a distância Euclidiana (JAIN; MURTY; FLYNN, 1999):

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Outras métricas também podem ser utilizadas para os atributos contínuos. Entre elas pode-se citar:

- Distância *City Block*:

$$d(x,y) = \sum_{i=1}^{n} \mid x_i - x_y \mid$$

- Distância *Minkowski*:

$$d(x,y) = \sqrt[p]{\sum_{i=1}^{n}(x_i - y_i)^p},\ \text{p>0}$$

- Distância *Camberra*:

$$d(x,y) = \sum_{i=1}^{n} \frac{|\, x_i - x_y \,|}{x_i + y_i},\ x_i \text{ e } y_i \text{ são positivos}$$

### 2.1.2 Atributos binários

Os atributos binários podem assumir exatamente dois valores. Como exemplos pode-se citar (sim, não) como respostas em um questionário, (F, M) sexo feminino ou masculino, entre outros. Para exemplificar as métricas utilizadas para o cálculo de similaridade considera-se dois vetores binários x e y que consistem de duas *strings* $[x_k]$ e $[y_k]$ de dados binários. Para compará-las admite-se as seguintes ocorrências:

- a = número de ocorrências quando $x_k = y_k = 1$;
- b = número de ocorrências quando $x_k = 1$ e $y_k = 0$;
- c = número de ocorrências quando $x_k = 0$ e $y_k = 1$;
- d = número de ocorrências quando $x_k = y_k = 0$.

.

Segundo Witold (2005), estes quatro números podem ser organizados em uma matriz $2x2$ também chamada de tabela de contingência:

| | 1 | 0 |
|---|---|---|
| 1 | a | b |
| 0 | c | d |

Com base nestas quatro entradas pode-se encontrar diversas medidas de similaridade de acordo com as seguintes métricas apontadas na literatura (WITOLD, 2005; KOGAN; NICHOLAS; TEBOULLE, 2006; JAIN; DUBES, 1988):

- A métrica mais simples obedece a seguinte fórmula:

$$d(x,y) = \frac{a+d}{a+b+c+d}$$

- Métrica de similaridade de Russel e Rao:

$$d(x,y) = \frac{a}{a+b+c+d}$$

- Coeficiente de Jaccard: envolve o caso onde $x_i = y_i = 1$

$$d(x,y) = \frac{a}{a+b+c}$$

- Coeficiente de Rogers e Tanimoto:

$$d(x,y) = \frac{a+d}{a+d+2(b+c)}$$

### 2.1.3 Atributos discretos

Os atributos discretos assumem um determinado número de estados (JAIN; DUBES, 1988). Por exemplo, para um conjunto de dados sobre as características físicas de um grupo de pessoas, o atributo "cor dos olhos" é um atributo discreto que pode assumir um entre vários estados (castanho escuro, castanho claro, preto, azul, verde).

Quando se trabalha com atributos discretos, a comparação normalmente é realizada verificando-se se os atributos são de mesmo estado ou não. Sendo assim, Anderberg (1973) estendeu o uso das métricas utilizadas em atributos binários para atributos discretos com um novo método de pesagem:

- $n_{a+d}$ é o número de atributos de mesmo estado nos dois objetos;
- $n_{b+c}$ é o número de atributos de estados diferentes nos dois objetos.

A métrica mais simples pode ser calculada de acordo com a equação:

$$d(x,y) = \frac{n_{a+d}}{n_{a+d} + n_{b+c}}$$

Pode-se também utilizar o coeficiente de Rogers e Tanimoto:

$$d(x,y) = \frac{n_{a+d}}{n_{a+d} + 2n_{b+c}}$$

## 2.2 Métodos de agrupamento de dados

Diferentes técnicas para agrupamento de dados são descritas na literatura. Novas abordagens vão surgindo de acordo com a área de aplicação, no entanto, não existem técnicas que possam ser universalmente aplicáveis (JAIN; MURTY; FLYNN, 1999).

Anderberg (1973) relaciona duas categorias de métodos para agrupamento de dados, os hierárquicos e os não hierárquicos, o qual também é conhecido como métodos baseados em particionamento. Estas categorias serão discutidas a seguir.

### 2.2.1 Métodos hierárquicos

Os métodos de agrupamentos hierárquicos produzem uma série de partições aninhadas formando uma hierarquia de grupos, ou seja, uma árvore de grupos denominada dendrograma (BERKHIN, 2002; ANDERBERG, 1973; KOGAN; NICHOLAS; TEBOULLE, 2006). Cada nível da hierarquia representa uma separação do conjunto de dados em um conjunto de grupos. A figura 2.2 mostra um exemplo de dendrograma resultante do agrupamento de cinco itens (A, B, C, D e E). O topo é um grupo contendo todos os cinco itens. Os demais grupos representados pela figura são [A], [B,C,D,E], [B,C] e [D,E].

Kogan, Nicholas e Teboulle (2006) apontam vantagens e desvantagens em se usar métodos de agrupamento hierárquico. Dentre as vantagens pode-se citar:

- flexibilidade com relação a granularidade;
- facilidade de manusear qualquer tipo de similaridade;
- aplicabilidade para qualquer tipo de atributo.

Como desvantagens dos métodos hierárquicos Kogan, Nicholas e Teboulle (2006) relacionam os seguintes:

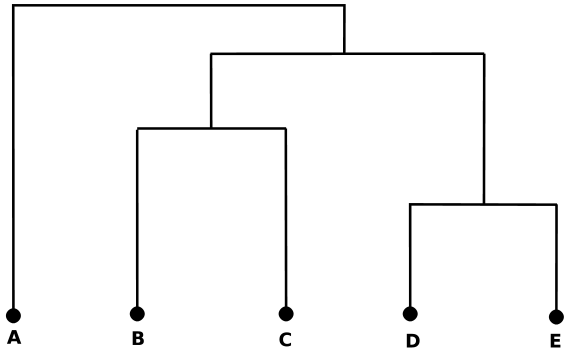


Figura 2.2: Exemplo de dendrograma

- dificuldade em escolher um critério de parada correto;
- a maioria dos algoritmos não são capazes de corrigir uma associação incorreta pois, não revisitam grupos já construídos.

Outra desvantagem que deve ser apontada é o fato da estrutura utilizada para representar os grupos, os dendrogramas, não serem adequadas quando se está trabalhando com uma grande quantidade de dados. Grandes dendrogramas, com vários níveis, são estruturas de difícil compreensão.

Os métodos hierárquicos são geralmete aplicados em áreas biológicas, sociais e nas ciências comportamentais devido a necessidade da construção de taxonomias (JAIN; DUBES, 1988). Estes são divididos em duas categorias - aglomerativos e divisivos (KOGAN; NICHOLAS; TEBOULLE, 2006; ANDERBERG, 1973), as quais serão discutidas a seguir.

#### *2.2.1.1 Aglomerativos*

Os métodos hierárquicos aglomerativos seguem uma abordagem *botton-up* onde cada item a ser agrupado é tratado inicialmente como um grupo de um único elemento que sucessivamente une-se ao grupo mais próximo, ou seja, mais similar, de acordo com uma métrica, formando outros grupos no decorrer das iterações.

Existem três tipos de métricas para se verificar a distância entre dois grupos (WITOLD, 2005; ROMESBURG, 2004):

- ligação simples: a distância entre dois grupos é dada pela distância entre os seus pontos mais próximos. Pode ser chamada também de *neighbour clustering*. Trata-se de um método guloso, que tende a buscar primeiro os objetos mais próximos e deixar os mais distantes em segundo plano;
- ligação média: a distância entre dois grupos é dada pela distância entre seus centróides. Seu problema está no fato de que cada vez que um grupo muda, seu centróide deve ser recalculado, bem como a distância para todos os outros grupos;
- ligação completa: a distância entre dois grupos é a distância entre seus pontos mais distantes.

A seguir pode-se verificar os passos do algoritmo que descreve o funcionamento do método aglomerativo:

1. Fazer um grupo para cada elemento;
2. Encontrar pares de grupos mais similares de acordo com uma medida de similaridade entre grupos;
3. Fundir estes pares de grupos em um grupo maior e recalcular a distância deste grupo para todos os outros elementos;
4. Repetir os passos 2 e 3 até restar um único grupo.

Em cada passo do algoritmo, os dois grupos mais próximos são unidos. O processo é repetido até se ter um único grupo de dados ou até alcançar um certo limiar pré-definido (WITOLD, 2005).

Um importante representante deste método de agrupamento de dados é o conhecido algoritmo *Average link* que será apresentado na Seção 4.3.

#### *2.2.1.2 Divisivos*

Os métodos hierárquicos divisivos, também conhecidos como *top-down*, trabalham de forma oposta aos aglomerativos. O processo inicia com o conjunto inteiro de itens a serem agrupados, sendo considerados como um único grupo, e prossegue dividindo-os em pequenos grupos. Em cada iteração remove-se o elemento que fica mais distante do centróide do grupo. O processo é interrompido quando houver apenas grupos de elementos isolados. Considerando-se a natureza do processo, esta abordagem apresenta-se computacionalmente ineficiente (WITOLD, 2005).

### 2.2.2 Métodos baseados em particionamento

O problema de agrupamento de dados baseado em particionamento pode ser formalmente declarado como segue (JAIN; DUBES, 1988):

- Dados $n$ objetos a serem agrupados em $k$ grupos:
  - parte-se de um agrupamento inicial;
  - calcula-se os pontos médios (centróides) dos grupos;
  - realiza-se uma iteração realocando-se os objetos entre os grupos de acordo com as distâncias entre os objetos e esses pontos médios.

Os métodos baseados em particionamento tentam maximizar as distâncias entre os pontos médios dos grupos e minimizar a distância entre os elementos de um mesmo grupo.

Estes são apropriados em aplicações envolvendo grandes conjuntos de dados, levando vantagens sobre os métodos hierárquicos já que para uma maior quantidade de dados a construção de um dendrograma é computacionalmente proibitiva (JAIN; MURTY; FLYNN, 1999).

Um importante e amplamente usado algoritmo representante deste método é o *K-means*, o qual será descrito na Seção 4.3.

## 2.3 Métodos para avaliar a qualidade de um agrupamento de dados

Validar um agrupamento de dados significa avaliar os resultados obtidos através de uma análise realizada de maneira quantitativa e objetiva (JAIN; DUBES, 1988). Com a

existência de uma grande variedade de técnicas de agrupamento de dados é natural que existam também um certo número de medidas de avaliação quantitativa destas técnicas. Uma completa relação e descrição dos critérios para investigar a qualidade de um agrupamento pode ser encontrado em (JAIN; DUBES, 1988). No presente trabalho apresenta-se somente as medidas de avaliação utilizadas para avaliar o resultado gerado pelo algoritmo proposto, as quais serão descritas nas próximas subseções.

### 2.3.1 Medida de avaliação externa

O primeiro grupo de funções de avaliação analíticas disponíveis para a análise de agrupamento é aquele desenvolvido para os problemas em que o número correto de grupos e a classificação correta para cada dado são conhecidos.

Existem alguns índices que podem ser empregados para a avaliação externa de um agrupamento. Quando a estrutura real dos dados é conhecida, o número de grupos é o primeiro indicador de desempenho do algoritmo, no caso desta informação não ter sido dada *a priori* como o que ocorre em diversas abordagens para agrupamento de dados. Compara-se então a quantidade de grupos gerada pelo algoritmo de agrupamento com o número de grupos real para o conjunto que está sendo analisado.

Uma outra medida conhecida é baseada no conhecimento prévio da distribuição dos rótulos das classes $C$ dos dados a serem agrupados. Trata-se do índice *F-measure* o qual combina as idéias de precisão e cobertura para comparar o resultado obtido pelo algoritmo de agrupamento dos dados com a distribuição real de classes destes dados. Para cada grupo $k$ relacionado a uma classe $C$ é calculado:

$$precisão(C,k) = \frac{l}{n}, \quad cobertura(C,k) = \frac{l}{m},$$

onde $l$ é o número de objetos da classe $C$ dentro do grupo $k$, $n$ é o número total de objetos dentro do grupo $k$ e $m$ é o número de objetos na classe $C$.

O índice *F-measure* da classe $C$ e do grupo $k$ é calculado de acordo com a Equação 2.1.

$$F(C,k) = \frac{2 * precisão(C,k) * cobertura(C,k)}{precisão(C,k) + cobertura(C,k)}, \tag{2.1}$$

O índice *F-measure* total para um agrupamento é computado pela Equação 2.2, a qual resulta valores no intervalo [0,1] e deve ser maximizada. $|\mathscr{A}|$ é o tamanho do conjunto de dados.

$$F = \sum \frac{m}{|\mathscr{A}|} * max\{F(C,k)\} \tag{2.2}$$

Outros métodos que geralmente são aplicados na avaliação de um agrupamento são os índices baseados nas estatísticas sobre a concordância, positiva ou negativa, na associação de pares de objetos aos grupos. O índice *Rand* penaliza as associações diferentes de pares de objetos ao comparar o agrupamento gerado com a correta classificação da base de dados.

Dado um agrupamento gerado $V$ cuja estrutura previamente conhecida é $Z$, calcula-se o índice *Rand* ($R$) de acordo com a Equação 2.3, onde as quantidades $a$, $b$, $o$, $z$ são computadas para todos os possíveis pares de dados $x$ e $y$ e seu respectivo grupo $k_Z(x)$, $k_Z(y)$, $k_V(x)$, e $k_V(y)$. $R$ é limitado ao intervalo [0,1] e deve ser maximizado.

$$R = \frac{a+z}{a+b+o+z}, \tag{2.3}$$

com:

$$\begin{aligned} a &= \{x,y|k_Z(x) = k_Z(y) \wedge k_V(x) = k_V(y)\}, \\ b &= \{x,y|k_Z(x) = k_Z(y) \wedge k_V(x) \neq k_V(y)\}, \\ o &= \{x,y|k_Z(x) \neq k_Z(y) \wedge k_V(x) = k_V(y)\}, \\ z &= \{x,y|k_Z(x) \neq k_Z(y) \wedge k_V(x) \neq k_V(y)\}. \end{aligned}$$

O índice $R$ resultará em 1 somente quando o agrupamento gerado for exatamente igual a estrutura real conhecida. Por outro lado, pequenos valores resultantes de $R$ indicam um aumento na falta de concordância entre estas duas estruturas.

### 2.3.2 Medida de avaliação interna

As medidas de avaliação interna medem a qualidade do agrupamento gerado baseando-se apenas no conjunto de dados originais. Estas medidas são normalmente usadas quando a estrutura real dos dados não é conhecida.

Existem alguns índices que podem ser empregados para a avaliação interna de um agrupamento. Entre os mais conhecidos destaca-se a variância intra-*cluster*, o qual será abordado no presente trabalho por se tratar da medida utilizada para avaliar a qualidade do grupo pelo algoritmo proposto.

A variância intra-*cluster* $var(V)$ de um agrupamento $V$ é baseada na minimização do somatório das distâncias entre os elementos e os centróides de seus grupos. $var(V)$ é dada pela Equação 2.4, onde $K$ é o número de grupos, $n$ é o número de elementos dentro do grupo $k$, $c_k$ é o centróide do grupo $k$ e $d(i,c_k)$ é a função de distância empregada entre o elemento $i$ e o centróide $c_k$.

$$var(V) = \sum_{k=1(k\in V)}^{K} \sum_{i=1(i\in k)}^{n} d(i,c_k) \tag{2.4}$$

Quanto menor o valor resultante de $var(V)$, melhor o agrupamento gerado.

## 2.4 Conclusão

Neste capítulo apresentou-se conceitos relacionados a área de agrupamento de dados. Moustrou-se as etapas para se realizar um agrupamento, bem como os tipos de atributos que podem ser manipulados e os índices de similaridade que devem ser utilizados durante o processo de acordo com estes atributos. Finalmente, descreveu-se alguns métodos para validação do agrupamento gerado por um algoritmo.

No próximo capítulo apresenta-se uma breve introdução sobre inteligência de enxames destacando-se alguns conceitos que são necessários para o entendimento da relevância de sua aplicação na solução de problemas distribuídos e consequentemente no desenvolvimento do algoritmo proposto.

# 3 INTELIGÊNCIA DE ENXAMES

Os insetos sociais - formigas, abelhas, vespas e cupins - têm sido objeto de estudos de diversos cientistas principalmente devido ao seu sucesso ecológico. Este sucesso é justificado pela eficiência de seu trabalho em conjunto. Um inseto pode ter apenas algumas centenas de células cerebrais, mas a sua organização é capaz de maravilhas arquitetônicas, de elaborar sistemas de comunicação e de resistir às terríveis ameaças da natureza (RUSSEL; NORVIG, 1995). Sua capacidade coletiva emerge da interação com o ambiente e entre os indivíduos, embora estes tenham uma cognição limitada. O repertório individual dos insetos é limitado e, no entanto, produz, em conjunto, um resultado surpreendente.

Inteligência de enxames (*Swarm Intelligence*) é a denominação aplicada a tentativa de desenvolvimento de algoritmos ou instrumentos para solução distribuída de problemas inspirando-se no comportamento coletivo de colônias de insetos sociais e outras sociedades de animais. Pesquisadores têm muitas razões para achar o estudo de inteligência de enxames atrativo pois, oferece um caminho alternativo para o desenvolvimento de sistemas inteligentes por possuir autonomia, emergência e controle distribuído (BONABEAU; THERAULAZ; DORIGO, 1999).

Como exemplos naturais de inteligência de enxames pode-se citar a construção de ninhos pelo térmites ou sociedades de abelhas, a atividade forrageadora das formigas, a divisão de trabalho e alocação de tarefas nas colônias de formigas, o transporte cooperativo de comida, a seleção de melhor fonte de néctar pelas abelhas, entre outros. O que há de comum nestes comportamentos são algumas características destacadas a seguir:

- flexibilidade: a colônia pode responder à perturbações internas e desafios externos;
- controle distribuído: na colônia não existe um indivíduo centralizador que direciona a realização das tarefas, a organização emerge das interações entre os constituintes do sistema e entre os indivíduos e o ambiente;
- robustez: um colapso individual não provoca a falha do sistema;
- auto-organização: caminhos para as soluções são emergentes e não pré-definidos.

Soluções para diferentes tipos de problemas têm sido encontradas usando heurísticas baseadas em comportamento de formigas e outros insetos. Em Bonabeau, Theraulaz e Dorigo (1999) são apontados alguns algoritmos para solução de problemas inspirados no comportamento de insetos sociais. Um exemplo é *Ant Colony Optimization* (ACO), uma meta-heurística que usa formigas artificiais para encontrar soluções para problemas de otimização combinatorial. O algoritmo inspira-se no comportamento observado em colônias de formigas, que estabelecem coletivamente a menor rota entre uma fonte de

alimento e seu ninho por meio de trilhas de feromônio deixadas ao longo do caminho (DORIGO; DI CARO; GAMBARDELLA, 1999). Dentre as aplicações do algoritmo, pode-se citar o roteamento de veículos, a coloração de grafos e o roteamento em redes de comunicação, entre outros.

Uma outra metáfora aplicada diz respeito a alocação de tarefas, processo que ajusta o número de formigas operárias engajadas em cada tarefa de forma apropriada a situação corrente (GORDON, 1996). Em sociedades de insetos, diferentes atividades são frequentemente realizadas simultaneamente por indivíduos especialistas. Este fenômeno é chamado divisão de trabalho e mostra-se mais eficiente do que se as tarefas fossem realizadas sequencialmente por indivíduos não especializados.

Na área de agrupamento de dados pode-se encontrar diversos trabalhos utilizando inteligência de enxames com o objetivo de melhorar os resultados obtidos com as técnicas tradicionais. Os principais comportamentos inspiradores destes trabalhos são a metáfora de organização de cemitérios e a escolha do menor caminho mediante depósito de um hormônio denominado feromônio. Ambos são comportamentos apresentados na natureza por colônias de formigas reais.

A seguir, descreve-se os principais comportamentos utilizados em algoritmos para agrupamento de dados, organização de cemitérios e escolha do menor caminho, juntamente com outro comportamento, a divisão de trabalho, utilizado pelo algoritmo proposto como inspiração para ajudar os agentes no processo de tomada de decisão sobre recrutar outros agentes para fazer parte de seus grupos. A Seção 3.4 apresenta a dança do recrutamento, um comportamento exibido pelas abelhas de uma colônia, o qual inspirou o processo de formação de grupos utilizado pelo algoritmo neste trabalho.

## 3.1 Organização de cemitérios

Diversas espécies de formigas separam os corpos de formigas mortas, formando um cemitério, ou agrupam suas larvas em diversas pilhas. Este comportamento simples tem incentivado diversos autores (BONABEAU; THERAULAZ; DORIGO, 1999; MONMARCHÉ; SLIMANE; VENTURINI, 1999; VIZINE et al., 2005) a desenvolver novas abordagens para solucionar o problema de agrupamento de dados.

Grande parte dos algoritmos para agrupamento de dados inspirados em insetos sociais partem de uma abordagem apresentada por Bonabeau, Theraulaz e Dorigo (1999). Esta abordagem deu origem ao *Ant Clustering Algorithm* (ACA). Seu mecanismo, inspirado na construção de cemitérios, contém formigas gentes que movem-se aleatoriamente em um *grid* bidimensional e quando encontram-se com um objeto (também distribuído aleatoriamente no mesmo *grid*) têm uma grande probabilidade de coletá-lo se este localiza-se em uma região com pouca concentração de outros objetos. Da mesma forma, a probabilidade de depositar um objeto em um determinado local é alta se a formiga estiver em uma área que contém muitos objetos. A idéia geral é que dados isolados devem ser coletados e posteriormente depositados em uma outra posição onde mais dados daquele tipo estejam presentes.

Formalmente pode-se definir que a probabilidade $p_p$ de um agente (formiga) coletar um objeto é dada pela Equação 3.1, onde $f$ é a fração de objetos percebidos na vizinhança do agente, ou seja, pode ser visto como a visão que cada formiga possui, $k_1$ é um limiar constante.

$$p_p = \left( \frac{k_1}{k_1 + f} \right)^2 \tag{3.1}$$

Quando $f \ll k_1$, $p_p$ é próximo de 1, ou seja, a probabilidade de coletar um objeto é alta quando não há muitos outros objetos na vizinhança. $p_p$ é próximo de 0 quando $f \gg k_1$, ou seja, dificulta a remoção dos objetos de locais onde há uma certa concentração.

Por outro lado, a probabilidade $p_d$ de um agente depositar um objeto que está carregando é dada pela Equação 3.2, onde $k_2$ é outro limiar constante.

$$p_d = \left( \frac{f}{k_2 + f} \right)^2 \tag{3.2}$$

Baseando-se nestas simples regras os agentes conseguem formar grupos de objetos semelhantes.

Para se utilizar este modelo como uma ferramenta de agrupamento de dados é necessário definir $f$ como uma função do problema a ser tratado. No contexto de robótica, por exemplo, $f$ foi definida como o número $N$ de objetos encontrados durante as últimas $T$ unidades de tempo dividido pelo maior número possível de objetos que podem ser encontrados durante $T$.

## 3.2 Divisão de trabalho em insetos sociais

Na natureza, nas colônias de insetos sociais diferentes atividades são frequentemente realizadas simultaneamente por indivíduos especialistas. Este fenômeno é conhecido como divisão de trabalho. A especialização permite aos indivíduos uma maior eficiência na realização das tarefas por conhecê-las e possuir habilidades para realizá-las (BONABEAU; THERAULAZ; DORIGO, 1999).

Para explicar este comportamento de divisão de trabalho exibido pelos insetos sociais, existem alguns modelos teóricos e matemáticos. Em Bonabeau, Theraulaz e Dorigo (1999, Capítulo 3), é descrito um modelo que depende de um limiar de resposta que cada indivíduo possui associado ao estímulo para realizar determinada tarefa. Sendo assim, um indivíduo tende a executar uma tarefa quando o estímulo $S$ para executá-la ultrapassa seu limiar $\theta$ associado. O limiar de resposta $\theta$, expresso em unidades de intensidade de estímulo, é uma variável interna que determina a tendência de um indivíduo, respondendo ao estímulo $S$, realizar a tarefa associada.

A Equação 3.3 mostra a função para o cálculo da probabilidade $(T)$, ou tendência, de um indivíduo atender a resposta a um estímulo.

$$T = \frac{S_i^2}{S_i^2 + \theta_i^2} \tag{3.3}$$

Se o valor do limiar $\theta \gg S$ a probabilidade do indivíduo realizar a tarefa tende a 0. Por outro lado, se o valor do limiar $\theta \ll S$ a probabilidade $T$ tende a 1. No entanto, quando $\theta = S$ a probabilidade é exatamente $\frac{1}{2}$.

Utilizou-se este modelo de estímulo resposta no algoritmo proposto como um mecanismo a ser utilizado pelo agente na tomada de decisão. Tal aplicação será posteriormente detalhada na Seção 5.1.

## 3.3 Modelo de recrutamento em colônias de formigas

Muitas espécies de formigas apresentam um comportamento interessante quando estão procurando por alimento: elas depositam uma substância química, chamada feromônio, ao longo do caminho entre o ninho e a fonte de alimento, formando trilhas. Outras formigas, atraídas por este feromônio, seguem esta trilha e também encontram a fonte. Este processo no qual uma formiga é guiada para uma fonte por meio da trilha de feromônio é denominado recrutamento (BONABEAU; THERAULAZ; DORIGO, 1999).

O *Ant colony optimization* (ACO) (DORIGO; MANIEZZO; COLORNI, 1996) é uma classe de algoritmos baseadas em insetos sociais que tem suas origens no comportamento de recrutamento. Os algoritmos ACO foram inspirados por uma experiência usando colônias de formigas reais na qual, mediante a apresentação de dois possíveis caminhos do ninho até uma fonte de alimento, após determinado tempo, a maioria das formigas selecionava o caminho mais curto para atingir o alimento desejado. A emergência do comportamento de seleção do caminho mais curto ocorre devido a forma de comunicação indireta entre as formigas por meio das trilhas de feromônio (DORIGO; DI CARO, 1999). Essa comunicação indireta via ambiente é denominada stigmergia. Maiores detalhes sobre o tema podem ser encontrados em Camazine et al. (2003).

No ACO, a principal tarefa de uma formiga é encontrar o menor caminho entre um par de nodos em um grafo, no qual o problema é adequadamente mapeado. Desta forma, $G = (N,E)$ é um grafo conectado com $n = |N|$ nodos. De maneira simplificada, o algoritmo ACO pode ser usado para encontrar uma solução para o problema de caminho mais curto definido no grafo $G$, onde a solução é um caminho no grafo conectando um nodo fonte $s$ a um nodo destino $d$. Para cada arco $(i, j)$ do grafo, é associada uma variável $\tau_{ij}$ chamada trilha de feromônio artificial. A quantidade de feromônio na trilha é proporcional a utilidade de usar o arco para construir uma boa solução. Cada formiga aplica, passo-a-passo, uma política de decisão construtiva para construir a solução do problema.

A regra de decisão de uma formiga $k$ localizada em um nodo $i$ usa a trilha de feromônio $\tau_{ij}$ para calcular a probabilidade $p_{ij}^k$ de escolher o nodo $j \in N_i$ como o próximo nodo para visitar. Então:

$$p_{ij}^k = \begin{cases} \tau_{ij} & \text{se } j \in N_i \\ 0 & \text{se } j \notin N_i \end{cases}$$

Enquanto constroem uma solução, formigas depositam uma quantidade constante $\Delta\tau$ de feromônio nos arcos por onde passam. Considerando-se uma formiga que no tempo $t$ move-se de um nodo $i$ para um nodo $j$, a atualização do feromônio $\tau_{ij}$ ocorre como segue:

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau \tag{3.4}$$

Assim, o uso de um arco conectando um nodo $i$ ao nodo $j$ aumenta a probabilidade de outras formigas usarem este mesmo arco no futuro. Com o objetivo de fazer com que as formigas explorem diferentes arcos durante todo o processo de busca, foi adicionado um comportamento de evaporação do feromônio. Desta forma, $\tau \leftarrow (1-\rho)\tau$, onde $\rho \in [0,1]$ em cada iteração do algoritmo.

## 3.4 Modelo de recrutamento em colônias de abelhas

O principal comportamento de interesse do presente trabalho encontra-se nas colônias de abelhas e chama-se dança do recrutamento. As abelhas de uma colônia selecionam

coletivamente a melhor fonte de néctar disponível usando regras comportamentais muito simples. Durante o processo de procura por alimento as abelhas viajam até 10 km longe de sua colméia pra coletar néctar. Após encontrá-lo, cada abelha retorna para a colméia com informações sobre a fonte do néctar coletado e pode apresentar três possíveis comportamentos:

- compartilhar informações sobre a fonte do néctar coletado por meio da dança do recrutamento, um comportamento no qual uma abelha comunica para suas companheiras a direção, distância e qualidade da fonte encontrada tentando recrutá-las para também forragearem naquela fonte;

- continuar coletando néctar na mesma fonte sem tentar recrutar outras abelhas;

- abandonar a fonte de néctar encontrada e posicionar-se em uma área dentro da colméia, denominada pista de dança, a fim de observar companheiras que estão dançando para suas fontes de néctar e ser atraída para forragear em uma outra fonte.

Em Seeley, Camazine e Sneyd (1991) os autores apresentaram um experimento mostrando como o processo de recrutamento acontece. Duas fontes de néctar são apresentadas para uma colônia de abelhas: fonte A e fonte B (a qual é melhor). Foi verificado que, passados alguns minutos, toda a colônia concentra sua atividade de coleta na fonte B, aquela que possui néctar de melhor qualidade. O experimento mostrou que a decisão de cada abelha é baseada em informações limitadas sobre a fonte de néctar que foi visitada. No entanto, embora simples, este comportamento faz com que a colônia selecione a fonte de melhor qualidade.

Baseando-se nestas informações, Camazine e Sneyd (1991) desenvolveram um modelo matemático que demonstra como as propriedades do sistema emergem de interações entre os agentes. Este modelo é baseado em agentes com as seguintes probabilidades:

- $PX_A$ e $PX_B$: probabilidade de abandonar as fontes de néctar A e B respectivamente;

- $PD_A$ e $PD_B$: probabilidade de executar a dança para recrutar companheiras para as fonte de néctar A e B respectivamente, onde $PD = 1 - PX$;

- $PF_A$ e $PF_B$: probabilidade de atender ao convite de uma abelha dançarina pra forragear nas fonte A e B respectivamente. $PF_A$ e $PF_B$ são calculadas de acordo com as Equações 3.5 e 3.6 , onde $d_A$ e $d_B$ indicam a proporção de tempo que as abelhas dançam para as fontes A e B respectivamente, $D_A$ e $D_B$ representam o número de abelhas forrageando nas fontes A e B respectivamente.

$$PF_A = \frac{D_A d_A}{D_A d_A + D_B d_B} \tag{3.5}$$

$$PF_B = \frac{D_B d_B}{D_A d_A + D_B d_B} \tag{3.6}$$

O modelo matemático proposto em Camazine e Sneyd (1991) reflete o comportamento observado em colônias de abelhas na natureza. Após determinados ciclos de tempo os agentes do modelo agrupam-se nas fontes de maior qualidade.

## 3.5 Conclusão

No presente capítulo apresentou-se conceitos relacionados a inteligência de enxames, descreveu-se os dois principais comportamentos utilizados em pesquisas relacionadas a agrupamento de dados inspirados em insetos sociais e detalhou-se o comportamento de interesse deste trabalho, a dança de recrutamento das abelhas, o qual inspirou o desenvolvimento do algoritmo proposto.

A capacidade de agrupamento encontrada nas colônias de abelhas durante o processo de coleta de alimentos mostra como um comportamento sofisticado emerge da interação entre seres bastante simples. Os mecanismos utilizados pelas abelhas na atividade de recrutamento serão aplicados na solução do problema de agrupamento de dados no algoritmo proposto no Capítulo 5.

# 4 TRABALHOS RELACIONADOS

O presente capítulo reúne alguns algoritmos para agrupamento de dados, relacionados ao algoritmo proposto, que foram investigados para o desenvolvimento do presente trabalho. A apresentação destes trabalhos encontra-se dividida em três categorias:

- Algoritmos inspirados por inteligência de enxames;
- Algoritmos distribuídos;
- Algoritmos para *ensemble* de agrupamento.

No final deste capítulo encontra-se uma comparação entre todos os algoritmos apresentados destacando-se suas principais características.

## 4.1 Algoritmos inspirados em inteligência de enxames

Nesta seção apresenta-se alguns algoritmos relacionados ao *bee clustering* que também são inspirados por inteligência de enxames. Tais algoritmos encontram-se subdivididos em três categorias de acordo com a inspiração biológica utilizada: algoritmos inspirados em ACO, algoritmos inspirados em organizações de cemitérios e algoritmos inspirados em outros comportamentos relacionados com inteligência de enxames.

### 4.1.1 Algoritmos inspirados em ACO

A primeira classe de algoritmos para agrupamento de dados a ser apresentada é aquela que utiliza como inspiração biológica o comportamento de recrutamento exibido por algumas colônias de formigas. Tal comportamento encontra-se descrito na Seção 3.3. A seguir, então, apresenta-se alguns exemplos de algoritmos inspirados em ACO para resolver o problema de agrupamento de dados.

#### *4.1.1.1 Algoritmo Shelokar*

Um importante representante dos algoritmos inspirados em ACO utilizados para agrupamentos é o algoritmo proposto por Shelokar, Jayaraman e Kulkarni (2004). Como os autores não deram um nome ao seu algoritmo, no presente trabalho ele será referenciado como algoritmo *Shelokar*.

O algoritmo *Shelokar* depende de trilhas de feromônio para guiar formigas de forma que elas selecionem um grupo para cada objeto. Uma busca local é necessária para melhorar a solução a cada iteração antes da trilha de feromônio ser alterada. O valor da trilha de feromônio é dado por $\tau_{ij}$ no nodo $(i, j)$. Onde $i$ é o objeto e $j$ é o grupo. No algoritmo,

formigas visitam os dados um de cada vez, sequencialmente, e selecionam um grupo para o objeto considerando unicamente a concentração de feromônio.

O algoritmo considera $R$ agentes (formigas) para construir soluções. Um agente começa com um vetor de soluções vazio $S$ de tamanho $N$ (número de objetos para agrupar). O valor atribuído aos elementos de $S$ representa o número do grupo para o qual o objeto é associado.

No início do algoritmo a matriz de feromônio de tamanho $N \times K$ ($K$ = número de grupos), é inicializada com pequenos valores de $\tau_0$. Para gerar uma solução $S$, a formiga seleciona o número do grupo da seguinte maneira:

- usando a probabilidade $q_0$: o grupo com concentração máxima de feromônio é escolhido. $q_0$ é um número definido *a priori*, $0 < q_0 < 1$. Os resultados apresentados pelo autor foram simulados com $q_0 = 0,98$;

- usando uma distribuição estocástica, $p_{ij}$, com probabilidade $(1 - q_0)$. Onde,

$$p_{ij} = \frac{\tau_{ij}}{\sum_{k=1}^{K} \tau_{ik}} j = 1, ..., K \tag{4.1}$$

  onde $p_{ij}$ é a probabilidade do objeto $i$ participar do grupo $j$.

A qualidade da solução construída pela formiga é medida de acordo com a seguinte função objetivo:

$$MinF(w,m) = \sum_{j=1}^{K} \sum_{i=1}^{N} \sum_{v=1}^{n} w_{ij} \parallel x_{iv} - m_{jv} \parallel^2 \tag{4.2}$$

tal que

$$\sum_{j=1}^{K} w_{ij} = 1, i = 1,..., N \tag{4.3}$$

$$\sum_{i=1}^{N} w_{ij} = 1, j = 1,..., K \tag{4.4}$$

onde $x_{iv}$ é o valor do $v^{th}$ atributo do $i^{th}$ objeto, $m$ uma matriz de centros de tamanho $K \times n$, $n$ é o número de atributos, $m_{jv}$ a média dos valores dos $v^{th}$ atributos de todos os objetos no grupo $j$, $w$ uma matriz de pesos de tamanho $N \times K$, $w_{ij}$ um peso associado do objeto $x_i$ com o grupo $j$ o qual pode ser atribuído da seguinte maneira:

$$w_{ij} = \begin{cases} 1 & \text{se o objeto } i \text{ está contido no grupo } j \\ 0 & \text{senão} \end{cases} \tag{4.5}$$

onde $i = 1,..., N$, $j = 1,..., K$

$m_{jv}$ pode ser calculado de acordo com a equação abaixo:

$$m_{jv} = \frac{\sum_{t=1}^{N} w_{ij} x_{iv}}{\sum_{t=1}^{N} w_{ij}}, j = 1,..., K, v = 1,..., n \tag{4.6}$$

O algoritmo *Shelokar* ainda realiza uma busca local em $L$ soluções geradas pelas formigas. $L$ representa as 20% melhores soluções do total de soluções. Antes da busca local,

as formigas são organizadas em ordem crescente de acordo com o valor resultante na equação 4.2.

Após a busca local, a matriz de feromônio é alterada considerando as $L$ melhores soluções entre os $R$ membros de acordo com a seguinte equação:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{l=1}^{L} \Delta\tau_{ij}^{l}, i = 1, ..., N, j = 1,..., K \tag{4.7}$$

onde $\rho$ é a persistência da trilha que respeita o intervalo $[0,1]$ e $(1-\rho)$ é a taxa de evaporação do feromônio. Valores de $\rho$ mais altos, indicam que as informações coletadas nas iterações passadas são esquecidas mais rapidamente. A quantidade $\Delta\tau_{ij}^{l}$ é igual a $1/F_l$ se o grupo $j$ é atribuído ao $i^{th}$ elemento da solução construída pela formiga $l$, senão é 0. A solução ótima é aquela que minimiza o valor da função objetivo, equação 4.2. O valor da ***melhor solução na memória*** será alterado para o valor da solução obtida como a ***melhor solução da iteração*** se esta obtiver com a função objetivo um valor mais baixo que a ***melhor solução na memória***.

Desta forma, o algoritmo *Shelokar* executa repetidamente os três passos a seguir até um valor máximo de iterações:

1. geração de novas $R$ soluções pelas formigas usando a informação da trilha de feromônio modificada originada das iterações anteriores;

2. realização de uma busca local;

3. alteração da matriz de feromônios.

O algoritmo foi testado em base de dados reais e artificiais e comparado com abordagens inspiradas em algoritmos genéticos e busca tabu. Segundo os autores, o algoritmo *Shelokar* mostrou-se uma abordagem viável e uma eficiente heurística para o grupamento de dados, já que resultou em soluções de boa qualidade em aceitável tempo de processamento.

Como pode ser visto, o algoritmo apresenta exatamente a mesma limitação imposta pelo *k-means*, necessidade de definição *a priori* do número de grupos em que o conjunto de dados será particionado, além de depender de informações disponíveis em várias estruturas de dados.

#### *4.1.1.2 ACOC*

Outro exemplo de algoritmo para agrupamento de dados inspirado no modelo do ACO é o *Ant Colony Optimization for Clustering* (ACOC) (KAO; CHENG, 2006). Dado um conjunto de dados contendo $m$ objetos com $n$ atributos e um número de grupos pré-determinado $(g)$ o ACOC encontra um agrupamento ótimo. O algoritmo possui como modelo de espaço de solução um grafo que representa a matriz de objetos que devem ser agrupados. O número de linhas é igual a $m$ e o número de colunas é igual a $g$. Se um nodo é denotado por $N(i, j)$ significa que o objeto $i$ será atribuído ao grupo $j$. No grafo, cada formiga move-se de um nodo para outro, depositando feromônio nos nodos e construindo uma solução. Os nodos com mais feromônio serão mais atrativos para as outras formigas. Em cada passo, uma formiga seleciona aleatoriamente um objeto não agrupado e o adiciona a um novo nodo considerando a intensidade do feromônio e a distância euclidiana entre o objeto a ser agrupado e cada centro de grupo armazenado na matriz de centros $(C^k)$ carregada pela formiga. Cada formiga possui, além da matriz de centros, uma lista que

representa sua memória ($tb^k$), para evitar que um objeto seja agrupado mais de uma vez por uma formiga. Quando esta lista está cheia, significa que a formiga completou a construção da solução. A formiga também mantém uma matriz de feromônio para armazenar os valores.

O algoritmo ACOC pode ser resumido nos seguintes passos:

1. Inicialização da matriz de feromônios: os elementos da matriz de feromônio (PM) são inicializados arbitrariamente com valores baixos ($\tau_0$);

2. Inicialização de todas as variáveis internas das formigas: para cada formiga, resseta a lista de memória ($tb^k$), a matriz de centros ($C^k$), de tamanho $g \times n$, e a matriz de pesos ($W^k$), de tamanho $m \times g$, onde $k = 1 \sim R$. $R$ é o total de formigas, $R \leq m$;

3. Seleção de um objeto: cada formiga seleciona aleatoriamente um objeto $i$ que não está em sua lista de memória ($tb^k$);

4. Seleção de um grupo $j$ para o objeto $i$: para determinar $j$, duas estratégias podem ser aplicadas:

   (a) Exploração: permite que formigas movam-se de maneira gulosa para um nodo cujo produto do nível de feromônio e o valor da heurística sejam os mais altos.

   $$j = \begin{cases} \arg\max_{u \in N_i}\{[\tau(i,u)][n^k(i,u)]^\beta\} & \text{se } q \leq q_o \\ S & \text{senão} \end{cases} \tag{4.8}$$

   onde $N_i$ é o conjunto de $g$ nodos nos quais o objeto $i$ pode ser colocado, $q$ é um valor probabilístico gerado aleatoriamente, $q_0$ é um valor probabilístico definido *a priori*, $\beta$ é um parâmetro especificando o relativo peso de $n^k_{ij}$, $\beta > 0$. $n^k_{ij} = 1/d^k(i,j)$ e é um valor heurístico de $N(i,j)$ para uma formiga $k$. O valor de $S$ é escolhido de acordo com a Equação 4.9.

   $$P^k(i,j) = \frac{[\tau(i,u)][n^k(i,u)]^\beta}{\sum_{j=1}^{g}[\tau(i,u)][n^k(i,u)]^\beta} \tag{4.9}$$

   A distância entre o objeto $i$ e o centro $j$ de uma formiga $k$, $d^k(i,j)$, é definido pela distância euclidiana:

   $$d^k(i,j) = \sqrt{\sum_{v=1}^{n}(x_{iv} - c^k_{jv})^2} \tag{4.10}$$

   onde $c^k_{jv}$ refere-se ao atributo $v$ do centro $j$ da formiga $k$, $x_{iv}$ é o valor do $v^{th}$ atributo do $i^{th}$ objeto.

   (b) Sondagem: distribui probabilidades para nodos candidatos, e deixa a formiga escolher um deles de maneira estocástica de acordo com a equação 4.9. O nodo mais promissor é aquele que tiver a mais alta probabilidade.

   Para escolher entre as duas possíveis estratégias, as formigas utilizam a equação 4.8.

5. Alteração da informação da formiga: alterar a lista de memória ($tb^k$), matriz de pesos ($w^k$) usando a equação 4.11 e a matriz de centros ($C^k$) usando a equação 4.12.

$$w_{ij} = \begin{cases} 1 & \text{se o objeto } i_{1,\ldots,m} \text{ é agrupado no grupo } j_{1,\ldots,g} \\ 0 & \text{senão} \end{cases} \tag{4.11}$$

$$C_j = \frac{\sum_{i=1}^{m} w_{ij} X_i}{\sum_{i=1}^{m} w_{ij}} \tag{4.12}$$

onde $j = 1, \ldots, g$, $X$ é a matriz de objetos de tamanho $m \times n$, $w_{ij}$ é o peso associado de $x_i$ com $c_j$, $x_i$ é o vetor do $i^{th}$ objeto e $x_i \in R^n$, $c_j$ é o vetor do $j^{th}$ centro de grupo e $c_j \in R^n$.

6. Verificação da lista de memória de cada formiga: checar se a lista de memória de cada formiga está cheia. Se está cheia, vai para o passo 7, senão volta para o passo 3;

7. Cálculo da função objetivo: calcular o valor da função objetivo de cada formiga, $J^k$, de acordo com a seguinte equação:

$$MinimizeJ(W,C) = \sum_{i=1}^{m} \sum_{j=1}^{g} w_{ij} \|X_i - C_j\| \tag{4.13}$$

onde

$$\|X_i - C_j\| = \sqrt{\sum_{v=1}^{n} (x_{iv} - c_{jv})^2} \tag{4.14}$$

Após, classifica-se as $R$ formigas em ordem crescente de $J^k$. A melhor solução é chamada de ***melhor solução da iteração*** e é comparada com a ***melhor solução encontrada*** proveniente de outras iterações. A melhor das duas será a nova ***melhor solução encontrada***. O resultado da comparação é armazenado para posteriormente ser comparado com outras melhores soluções encontradas nas iterações.

8. Alteração da trilha de feromônio: alterar a matriz de feromônio (PM). A regra de alteração global é aplicada e somente as formigas elitistas podem adicionar feromônio no final de cada iteração.

9. Verificação da condição de parada: se o número de iterações exceder o número máximo permitido, então o algoritmo pára e retorna a melhor solução encontrada, senão, volta para o passo 2.

O algoritmo ACOC representa uma extensão do algoritmo proposto por Shelokar, Jayaraman e Kulkarni (2004), explicado anteriormente, onde a diferença significante existente entre os dois é que o ACOC não limita-se unicamente ao agrupamento baseado na quantidade de feromônio, ele também leva em consideração a distância euclidiana entre o objeto e o centro de uma formiga. Esta característica faz com que o ACOC se destaque na qualidade das soluções produzidas.

O testes foram realizados pelos autores com bases de dados reais e artificiais. Seus resultados, comparados com aqueles obtidos com o algoritmo *Shelokar* e *K-means*, mostraram que embora seu desempenho seja bem inferior ao *K-means*, as soluções geradas foram melhores. Já com relação ao algoritmo *Shelokar*, o ACOC mostrou-se melhor em desempenho e nas soluções construídas.

Uma deficiência verificada no algoritmo ACOC é com relação ao emprego, para cada formiga, de uma memória para evitar que um objeto seja agrupado mais de uma vez. Desta forma as formigas deixam de ter um conhecimento puramente local, como ocorre com formigas reais, para apresentarem um conhecimento do sistema em um nível mais global. Em Johnson (2003) o fato da *ignorância* ser uma característica e não um defeito é apontado como um dos princípios para se obter sucesso global por meio de conhecimento local. Johnson (2003) menciona ainda que é melhor se construir sistemas descentralizados com elementos simples densamente interconectados e deixar que o comportamente mais sofisticado ocorra aos poucos.

### 4.1.2 Algoritmos inspirados em organizações de cemitérios

Outra classe de algoritmos a ser apresentada no presente trabalho é aquela que utiliza como inspiração biológica o comportamento de organização de cemitérios, também exibido por algumas colônias de formigas. A maioria dos algoritmos para agrupamento de dados que são inspirados em inteligência de enxames costumam utilizar este tipo de comportamento natural como inspiração. Tal comportamento encontra-se descrito na Seção 3.1. A seguir, encontram-se detalhados alguns algoritmos inspirados neste comportamento.

#### *4.1.2.1 SACA*

Lumer e Faieta (1994) desenvolveram o algoritmo *Standard Ant Clustering Algorithm* (SACA) inspirado no algoritmo ACA, apresentado na Seção 3.1. Os dados que deveriam ser agrupados foram tomados em um espaço Euclidiano de dimensão $L$, $\Re^L$. Como ambiente os autores utilizaram uma grade bi-dimensional com vizinhança unitária.

A função $f$ é calculada pela seguinte equação:

$$f(X_i) = \begin{cases} \frac{1}{S^2} \sum_{x_j \in Viz_{(sxs)}} \left[ \frac{1-d(X_i,X_j)}{\alpha} \right] & \text{se } f > 0 \\ 0 & \text{senão} \end{cases} \tag{4.15}$$

onde $f(X_i)$ é a medida da similaridade média do item $X_i$ em relação a outro item $X_j$ na vizinhança de $X_j$, $\alpha \in [0,1]$ é um fator que define a escala de dissimilaridade, $d(X_i,X_j) \in [0,1]$ é a distância Euclidiana entre os dados $X_i$ e $X_j$ em seus espaços originais, e $S^2$ é o tamanho da vizinhança local (normalmente $S^2 \in [9,25]$).

No SACA, as probabilidades de um agente coletar e depositar um objeto são dadas pelas equações 4.16 e 4.17 respectivamente.

$$p_p(X_i) = \left( \frac{k_1}{k_1 + f(X_i)} \right)^2 \tag{4.16}$$

$$p_d(X_i) = \begin{cases} 2f(X_i) & \text{se } f(X_i) < k_2 \\ 0 & \text{se } f(X_i) \geq k_2 s \end{cases} \tag{4.17}$$

Da mesma forma que no algoritmo ACA, $k_1$ e $k_2$ são limiares constantes.

Uma das maiores dificuldades em aplicar o *SACA* para resolver problemas complexos vem do fato de que ele gera um número de grupos que é bem maior que a quantidade necessária. Além disso, o algoritmo geralmente não estabiliza em uma solução, ou seja, fica constantemente construindo e desconstruindo grupos durante suas iterações.

A seguir, apresenta-se o algoritmo $A^2CA$ desenvolvido com o intuito de superar os problemas anteriormente mencionados.

#### *4.1.2.2 $A^2CA$*

Em Vizine et al. (2005) é proposto o *Adaptive Ant-Clustering Algorithm* ($A^2CA$), um algoritmo para agrupamento de dados com o objetivo de melhorar os resultados já obtidos com o algoritmo SACA.

O $A^2CA$ , além de ter sido inspirado no comportamento de agrupamento de corpos pelas formigas, também buscou inspiração no comportamento dos cupins que enquanto estão construindo seu ninho, depositam feromônio nas porções de terra empilhadas, servindo como sinal de reforço para outros cupins empilharem mais terra na mesma região. Outra inspiração biológica leva em consideração o fato de que as formigas não se limitam a perceber somente suas vizinhas imediatas, elas têm um poder de alcance um pouco maior que varia no tempo e de formiga para formiga.

Da mesma forma que no SACA, a idéia geral do algoritmo consiste em um *grid* com objetos alocados aleatoriamente e formigas caminhando aleatoriamente para tentar agrupar os objetos que são mais similares respeitando aqueles dois comportamentos, citados anteriormente, de coletar e depositar um determinado objeto em uma posição do *grid*.

Para superar alguns problemas encontrados no SACA, o $A^2CA$ propõe três modificações:

1. um decaimento de $k_1$ (limiar que no SACA era constante):
   - a cada ciclo (10000 passos) $k_1$ sofre um decaimento geométrico: $k_1 \longleftarrow 0.98 \times k_1$ $(k_{1min} = 0.001)$
2. um campo de visão progressiva:
   - o algoritmo SACA define um campo de visão fixo para a formiga, o que pode algumas vezes causar comportamentos inapropriados por não permitir a distinção entre grupos de diferentes tamanhos. Uma área de percepção pequena, implica em uma pequena percepção dos grupos em um nível global. Por outro lado, uma grande área de percepção pode ser ineficiente nas iterações iniciais. Sendo assim, o $A^2CA$ propõe um campo de visão progressivo, ou seja, quando a formiga percebe um grande grupo ela incrementa o seu campo de visão $s_i^2$ em $n_s$ unidades até alcançar seu valor máximo. Portanto, $s_i^2$ é um parâmetro específico para cada formiga que será dinamicamente e independentemente alterado enquanto o algoritmo é executado. Assim, se $f(X_i) > \theta$ e $s^2 \leq s_{max}^2$ então $s^2 \longleftarrow s^2 + n_s$. Os autores sugerem $s_{max}^2 = 7 \times 7$ e $\theta = 0.6$.
3. uma heurística inspirada no processo de *feedback* positivo [1] por meio de feromônio depositado pelos cupins quando estão realizando a tarefa de construção de ninhos:
   - o algoritmo propõe a adição de uma variável $\phi(i)$ associada com cada posição $i$ do *grid*, tal que a quantidade de feromônio naquela exata posição torna-se uma função de presença ou ausência de um objeto em $i$. A formiga artificial adicionará feromônio no objeto que está transportando e este será transferido para a posição do *grid* quando o objeto for depositado. Durante cada iteração, o feromônio $\phi(i)$ de cada célula do *grid* evapora de acordo com uma taxa fixa. A probabilidade de uma formiga pegar um objeto do *grid* é inversamente

[1] são simples regras comportamentais que executadas pelos agentes, e mediante a obtenção de bons resultados, fazem com que os demais agentes também passem a atuar segundo estas regras. Maiores detalhes podem ser encontrados em Camazine et al. (2003).

proporcional à quantidade de feromônio da posição do objeto e à densidade de objetos ao redor de $i$. A abordagem proposta aumenta a probabilidade de desconstrução de grupos pequenos e de alocação de objetos em grupos mais densos. Desta forma, as probabilidades de coletar e depositar um objeto em uma determinada posição $i$ do *grid*, são dadas pelas equações 4.18 e 4.19 respectivamente.

$$p_p(i) = \frac{1}{f(i)\phi(i)} \left( \frac{k_1}{k_1 + f(i)} \right)^2 \tag{4.18}$$

$$p_d(i) = f(i)\phi(i) \left( \frac{f(i)}{k_2 + f(i)} \right)^2 \tag{4.19}$$

onde, $f(i)$ é a função dependente da densidade de objetos.

A taxa na qual o feromônio evapora em cada posição do *grid* é calculada de acordo com: $\phi(i) \longleftarrow \phi(i) * 0.99$.

Para a validação do algoritmo os autores também utilizaram bases de dados reais e artificiais, comparando os seus resultados com o algoritmo SACA. Segundo os autores, o $A^2CA$ conseguiu superar o SACA demonstrando robustez para classificar os dados de uma base em um número correto de grupos, baixa variação nos resultados, e estabilização após um número fixo de iterações automaticamente definidas pelo algoritmo.

Os autores do $A^2CA$ preocuparam-se em proporcionar a apresentação dos resultados obtidos de maneira visual em um *grid* bidimensional, o que, segundo eles, pode ajudar o usuário a lidar com a sobrecarga de informações.

A característica acima mencionada pode ser apontada como vantagem somente se o algoritmo estiver tratando uma base de dados relativamente pequena. Caso contrário a representação dos grupos em um *grid* bidimensional ficará inviável.

#### *4.1.2.3 AntClass*

Monmarché, Slimane e Venturini (1999) desenvolveram o algoritmo denominado *AntClass* com o objetivo de agrupar dados numéricos sem a necessidade de se definir *a priori* quantos grupos serão necessários, como ocorre com o *K-means*.

No *AntClass* as formigas estão aptas a criar, construir e destruir pilhas de objetos. Uma pilha consiste de uma coleção de pelo menos dois objetos.

O algoritmo base (Algoritmo 1) inspira-se no comportamento de organização de cemitérios, ou seja, o de coletar e depositar um objeto em determinada posição.

**Algorithm 1**: Algoritmo base *AntClass*

```
1 Inicialize aleatoriamente as formigas no grid ;
2 repeat
3     foreach formiga ant_i do
4         Move ant_i ;
5         if ant_i não está carregando nenhum objeto then
6             Procedure 1 ;
7         else
8             Procedure 2 ;
9 until critério de parada;
```

No *AntClass* as formigas não movimentam-se de forma totalmente aleatória como ocorre em outros algoritmos. O *Move*, mostrado no algoritmo 1, indica que a formiga seleciona uma entre oito possíveis direções e então com uma probabilidade $P_{direction}$ ela continua na direção selecionada ou gera aleatoriamente uma nova direção.

As formigas do algoritmo *AntClass* podem visualizar tanto objetos quanto pilhas de objetos a serem agrupados. Quando uma formiga visualiza uma pilha $H$ para agrupar, a qual possui mais que dois objetos, ela coleta o objeto mais dissimilar, $O_{dissim}$, desde que sua dissimilaridade respeite a seguinte restrição:

$$\frac{D(O_{dissim}(H), O_{center}(H))}{D_{mean}(H)} > T_{remove}$$

onde $O_{center}(H)$ e $D_{mean}(H)$ são calculados segundo as Equações 4.20 e 4.21 respectivamente.

$$O_{center}(H) = \frac{1}{n_H} \sum_{O_i \in H} O_i \tag{4.20}$$

$$D_{mean}(H) = \frac{1}{n_H} \sum_{O_i \in H} D(O_i, O_{center}(H)) \tag{4.21}$$

$n_H$ é a quantidade de objetos na pilha, $D$ é a distância Euclidiana e $T_{remove} \in [0.1, 0.2]$ é o valor mínimo de dissimilaridade necessário para remover um objeto de uma pilha.

**Procedure 1** `ColetarObjeto`

```
1  Rotular as 8 c posições vizinhas a ant_i como inexploradas;
2  repeat
3      foreach c_i do
4          if c_i está ocupado com um objeto O then
5              Pegue O com probabilidade P_load;
6          else if c_i está ocupado com uma pilha H com dois objetos then
7              Remova um dos dois objetos com probabilidade P_destroy;
8          else if c_i está ocupado com uma pilha H com mais que dois objetos then
9              if ( D(O_dissim(H),O_center(H)) / D_mean(H) > T_remove ) then
10                 Remova o objeto mais dissimilar O_dissim(H) ;
11     Rotular c_i como explorado;
12 until todas as posições tenham sido exploradas ou um objeto tenha sido coletado;
```

Por outro lado, quando uma formiga está carregando um objeto, ela verifica as oito posições ao redor de sua posição corrente e considera três possíveis casos:

- se a posição está vazia: então a formiga possui uma probabilidade $P_{drop}$ de depositar o objeto;
- se a posição contém somente um objeto $O'$: a formiga depositará o objeto $O$ em $O'$ e criará uma pilha de dois objetos contanto que a seguinte restrição seja satisfeita:

$$\frac{D(O, O')}{D_{max}} < T_{create}$$

onde $T_{create}$ é o máximo valor de dissimilaridade permitido para criar uma pilha de dois objetos, e $D_{max}$ é calculado de acordo com a Equação 4.22. seguinte equação:

$$D_{max}(H) = max_{O_i,O_j \in H} D(O_i, O_j) \tag{4.22}$$

- se a posição contém uma pilha $H$: neste caso a formiga adiciona o objeto $O$ em $H$ contanto que:

$$D(O, O_{center}(H)) < D(O_{dissim}(H), O_{center}(H))$$

**Procedure 2** `DepositarObjeto`

```
1  Rotular as 8 c posições vizinhas a ant_i como inexploradas;
2  repeat
3      foreach c_i = 1,...,8 do
4          if c_i está vazio then
5              Deposite O com probabilidade P_drop;
6          else if c_i está ocupado com um objeto O' then
7              if (D(O,O')/D_max) < T_create then
8                  Deposite O em O' para criar uma pilha;
9          else if c_i está ocupado com uma pilha H then
10             if D(O, O_center(H)) < D(O_dissim(H), O_center(H)) then
11                 Deposite O em H ;
12     Rotular c_i como explorado;
13 until todas as posições tenham sido exploradas ou um objeto tenha sido
   depositado;
```

Uma outra característica importante do *AntClass* está relacionada a memória local de cada formiga. Desta forma, quando uma formiga encontra uma pilha $H$ qualquer, ela armazena em sua memória a sua localização, $O_{center}(H)$, $D(O_{Dissim}(H), O_{center}(H))$ e quando está carregando um objeto, ela busca em sua memória por uma pilha no qual ela poderia depositar aquele objeto. Se ela encontra, então a memória desta pilha é ativada e a formiga vai até a sua posição. Se a formiga não depositar o objeto em outras pilhas que encontrar no caminho, então ela depositará o objeto na pilha alvo, desde que ela ainda esteja válida, ou seja, não tenha sido destruída ou muito modificada por outra formiga. A memória da formiga possui quatro posições. Sendo assim, quando uma nova pilha é encontrada e a sua memória está cheia, as pilhas mais velhas são substituídas pela nova fazendo com que a formiga esqueça as pilhas antigas. Esta memorização foi adicionada a formiga com o intuito de aumentar e acelerar a atribuição de objetos as pilhas.

O desempenho do *AntClass* mostrou-se ineficiente com as característica apresentadas anteriormente, já que sua convergência além de ser longa era difícil de ser atingida. Sendo assim, os autores combinaram o algoritmo *K-means* a sua abordagem com o intuito de corrigir os erros de agrupamento cometidos pelas formigas e também agrupar rapidamente e eficientemente os objetos não agrupados por elas.

A combinação do *K-means* com o algoritmo realizado pelas formigas mostrou-se válida pelo fato de corrigir diversas classificações realizadas de maneira incorreta. No entanto, deixou a desejar no sentido de que o número de grupos gerados foi sempre superestimado. Isto fez com que os autores introduzissem uma outra abordagem inspirada em agrupamentos hierárquicos.

Desta forma, após ser executado o algoritmo baseado no comportamento das formigas e o algoritmo *K-means*, uma versão modificada do algoritmo das formigas é executada para completar o processo. A modificação proposta foi simplesmente que ao invés de lidar com coleta e depósito de objetos, agora as formigas trabalham na coleta e depósito de pilhas inteiras diminuindo assim o número de grupos gerados.

A adição deste terceiro passo ao algoritmo *AntClass* fez com que pequenos erros de classificação voltassem a acontecer. Para solucionar este problema os autores introduziram um outro passo adicional ao algoritmo, rodando novamente o *K-means*. Sendo assim, o algoritmo *AntClass* constitue-se de quatro passos:

1. o algoritmo baseado no comportamento de formigas;
2. algoritmo *K-means* usando o agrupamento inicial fornecido pelas formigas;
3. o algoritmo das formigas modificado, onde elas carregam pilhas inteiras de objetos ao invés de um único objeto;
4. e novamente o *K-means*.

O AntClass foi aplicado pelos autores em bases de dados numéricos reais e artificiais e comparado com *K-means* e ISODATA (BALL; HALL, 1965), uma versão melhorada do *K-means*. Como resultado, o AntClass se mostrou superior aos demais ao realizar um agrupamento de melhor qualidade.

O algoritmo AntClass não requer qualquer informação inicial sobre o particionamento, o que constitue uma importante vantagem. Entretanto é dependente de uma série de parâmetros gerados em um intervalo pré-determinado pelos autores. Isto poderia ser um problema se todos os agentes do modelo assumissem os mesmos valores. No entanto, a abordagem de agentes heterogêneos aplicada ao algoritmo parece ter superado este inconveniente.

#### *4.1.2.4 CSIM*

Um outro algoritmo, denominado *Clustering Algorithm based on Swarm Intelligence and K-means* (CSIM) e inspirado na organização de cemitérios, é encontrado em Bin et al. (2002). O CSIM foi desenvolvido com o propósito de agrupar documentos. Ele trabalha de maneira semelhante ao algoritmo ACA apresentado em (BONABEAU; THERAULAZ; DORIGO, 1999). Sendo assim, o processo de agrupamento ocorre baseando-se nas probabilidades de coletar e depositar objetos em determinadas posições formando grupos de objetos similares.

Para cada documento $o_i$ em uma coleção de documentos $k$, $V$ é o conjunto de palavras únicas encontrado em $k$ e $m = |V|$. O vetor $O_i = (w_{i1}, w_{i2}, w_{i3}, ..., w_{im})$ representa o documento $o_i$, onde $w_{ij}$ é a freqüência da palavra $w_j$ no documento $o_i$. O CSIM é detalhado no Algoritmo 4.

Para medir a similaridade entre um par de documentos, o CSIM usa a seguinte equação:

$$sim(o_i, o_j) = \frac{w_{i1} * w_{j1} + ... + w_{im} * w_{jm}}{|o_i| * |o_j|} \tag{4.23}$$

**Algorithm 4**: *Algoritmo base CSIM*

```
1  antNumber é o número total de formigas;
2  n é o número de iterações;
3  α é um coeficiente de similaridade;
4  p_r é uma probabilidade gerada aleatoriamente;
5  k é um limiar constante;
6  Inicializar o coeficiente α, antNumber, n, k e demais parâmetros necessários;
7  Atribuir um par de coordenadas (x, y) para cada objeto;
8  Inicializar as antNumber formigas e atribuir a cada uma um objeto;
9  repeat
10     foreach j = 1, ..., antNumber do
11         Computar a similaridade do objeto dentro de uma região local com raio r de
           acordo com a equação 4.26;
12         if formiga está descarregada then
13             Calcular a probabilidade P_p de coletar o objeto de acordo com a
               equação 4.24;
14             Comparar P_p com a probabilidade P_r;
15             if (P_p < P_r) then
16                 a formiga não coletará este objeto;
17                 outro objeto é aleatoriamente atribuído a formiga;
18             else
19                 a formiga coleta o objeto e troca seu estado para carregada;
20                 um novo par de coordenadas é passado como posição pra formiga;
21             else if formiga está carregada then
22                 Calcular a probabilidade P_d de depositar o objeto de acordo com a
                   equação 4.25;
23                 if (P_p > P_r) then
24                     a formiga depositará o objeto;
25                     outro objeto é aleatoriamente atribuído a formiga;
26                     um novo par de coordenadas é passado como posição pra
                       formiga;
27                 else
28                     um novo par de coordenadas é passado como posição pra
                       formiga;
29                     a formiga continua com o mesmo objeto;
30 until n;
```

onde $(w_{i1},...,w_{im})$ indica o documento $o_i$ e $(w_{j1},...,w_{jm})$ indica o documento $o_j$.

As probabilidades de coletar e depositar um objeto são calculadas de acordo com as equações 4.24 e 4.25 respectivamente.

$$P_p = \begin{cases} 1 & f(o_i) \leq 0 \\ 1 - k * f(o_i) & 0 < f(o_i) \leq 1 \div k \\ 0 & f(o_i) > 1 \div k \end{cases} \tag{4.24}$$

$$P_d = \begin{cases} 1 & f(o_i) \geq 1 \div k \\ k * f(o_i) & 0 < f(o_i) \leq 1 \div k \\ 0 & f(o_i) \leq 0 \end{cases} \tag{4.25}$$

onde $k$ é um limiar constante e $f(o_i)$ é calculado de acordo com a seguinte equação:

$$f(o_i) = \sum_{o_j \in Neigh(r)} \left[ 1 - \frac{10 * (1 - sim(d_i, d_j))}{\alpha} \right] \tag{4.26}$$

$\alpha \in [1,10]$ é um coeficiente de similaridade, $d_i$ e $d_j$ são calculados pelas equações 4.27 e 4.28 respectivamente.

$$|d_i| = \sqrt{\sum_{k=1}^{m} w_{ik}^2} \tag{4.27}$$

$$|d_j| = \sqrt{\sum_{k=1}^{m} w_{jk}^2} \tag{4.28}$$

Da mesma maneira que o *AntClass*, o CSIM emprega o algoritmo *K-means* para melhorar os resultados obtidos. Desta forma, é composto de duas fases:

1. um conjunto inicial de grupos é formado pelo algoritmo base (Algoritmo 4);
2. em seguida o método *K-means* é empregado otimizando os resultados.

O CSIM foi testado em três bases de documentos reais e comparado com o *K-means* e CSI (BIN; ZHONGZHI, 2001). Os resultados mostraram um melhor desempenho do CSIM em relação aos demais algoritmos no domínio de agrupamento de documentos.

#### *4.1.2.5 Algoritmo baseado em múltiplas colônias de formigas*

Em Yang e Kamel (2006) é proposto um algoritmo que, além da organização de cemitérios, imita o comportamento cooperativo de colônias de formigas, onde diversas colônias trabalham paralelamente e independentemente cooperando mutuamente através da troca de informações para encontrar uma solução otimizada. O algoritmo consiste de duas partes: uma onde diversas colônias de formigas, independentes, usando o algoritmo básico ACA (BONABEAU; THERAULAZ; DORIGO, 1999), agrupam um conjunto de objetos e outra onde uma formiga rainha junta os agrupamentos das diferentes colônias.

Os algoritmos 5, 6 e 7 descrevem todo o procedimento envolvido neste método de agrupamento de dados.

O algoritmo consiste de múltiplas colônias de formigas trabalhando paralelamente para produzir o melhor agrupamento possível de um conjunto de dados. As colônias são heterogêneas com quatro tipos diferentes de possíveis determinação dos parâmetros $\beta$ (coeficiente de similaridade), $v$ (velocidade da formiga) $\mu$ (coeficiente de dissimilaridade):

**Algorithm 5**: *Agrupamento inspirado em múltiplas colônias*

```
N é o número de colônias de formigas;
antNumber é o número de formigas em cada colônia;
Inicializar os coeficientes N, M, antNumber e demais parâmetros necessários;
repeat
    Agrupamento baseado em formigas();
until N;
```

1. constante: todas as formigas possuem os mesmos valores para os parâmetros;
2. aleatório: os valores são distribuídos aleatoriamente entre [1, valor máximo];
3. aleatoriamente dividido: metade das formigas assume um determinado valor enquanto que a outra metade assume outro;
4. aleatoriamente diminuído: os parâmetros iniciam com valores altos que vão decaindo gradativamente de maneira aleatória.

A probabilidade de uma formiga *descarregada*, movendo-se aleatoriamente, coletar um objeto é calculada pela equação 4.29 enquanto que a probabilidade de uma formiga depositar o objeto que está carregando é calculada pela equação 4.30, onde $f(o_i)$, denota a similaridade entre o objeto $i$ e os demais objetos localizados na vizinhança da formiga, e pode ser calculada pela equação 4.31 ou 4.32. $Z$ é a matriz de similaridade $nxn$, $\mu$ é um coeficiente de dissimilaridade e $\beta$ é um coeficiente de similaridade. Se $\beta$ receber um valor muito alto, resultará em uma tendência maior dos objetos ficarem no mesmo grupo. Por outro lado, quando $\beta$ recebe um valor baixo, a similaridade diminuirá e pode ocasionar um resultado extremo de muitos grupos separados.

$$P_p(o_i) = 1 - sigmoid(f(o_i)) \tag{4.29}$$

$$P_d(o_i) = sigmoid(f(o_i)) \tag{4.30}$$

onde

$$sigmoid(x) = \frac{1}{1+e^{-cx}}$$

$c_x$ é uma variável que pode acelerar a convergência do algoritmo quando é incrementada. Durante o procedimento de agrupamento, podem haver alguns objetos muito dissimilares de todos os outros elementos. As formigas tendem a ficar bloqueadas tentando posicionar estes objetos diminuindo a velocidade da convergência do algoritmo. Quando isto ocorre, o parâmetro $c_x$ pode ser alterado para um valor maior com o objetivo de ajudar as formigas a depositarem estes objetos dissimilares nos próximos estágios do algoritmo.

$$f(o_i) = \sum_{o_i \in Neigh_{sxs}(r)} \left[ 1 - \frac{d(o_i, o_j)}{\beta} \right] \tag{4.31}$$

$$f(o_i) = \sum_{o_i \in Neigh_{sxs}(r)} \left[ \frac{Z[i,j]}{\mu} \right] \tag{4.32}$$

**Algorithm 6**: *Agrupamento baseado em formigas()*

```
1  Mm é o número máximo de iterações;
2  Atribuir aleatoriamente um objeto para cada formiga que está descarregada;
3  repeat
4      foreach j = 1,...,antNumber do
5          if Agrupamento inspirado em múltiplas colônias (Algoritmo 5) then
6              Computar a similaridade do objeto dentro de uma região local de
               acordo com a equação 4.31;
7          else if Agrupamento agregado (Algoritmo7) then
8              Computar a similaridade do objeto de acordo com a equação 4.32;
9          if formiga está descarregada then
10             Calcular a probabilidade P_p de coletar o objeto de acordo com a
               equação 4.29;
11             Comparar P_p com a probabilidade P_r;
12             if (P_p < P_r) then
13                 a formiga não coletará este objeto;
14                 outro objeto é aleatoriamente atribuído a formiga;
15             else
16                 a formiga coleta o objeto e troca seu estado para carregada;
17                 move o objeto para uma nova posição;
18         else if formiga está carregada then
19             Calcular a probabilidade P_d de depositar o objeto de acordo com a
               equação 4.30;
20             if (P_p > P_r) then
21                 a formiga depositará o objeto;
22                 a formiga troca seu estado para descarregada;
23                 outro objeto é aleatoriamente atribuído a formiga;
24             else
25                 a formiga continua com o mesmo objeto movendo-o para uma nova
                   posição;
26 until Mn;
```

**Algorithm 7**: *Agrupamento agregado()*

```
1  M é o número máximo de iterações;
2  repeat
3      Coletar o resultado do agrupamento gerado em (Algorithm 5);
4      Computar a matriz de adjacência H de acordo com a equação 4.34;
5      Calcular a matriz de similaridades Z de acordo com a equação 4.33 e enviá-la
       de volta para cada colônia de formiga;
6      Selecionar um agrupamento como grupo de dados atual para a colônia de
       formigas de acordo com uma das estratégias: KP, CE, ou LO;
7      foreach i = 1,...,N do
8          Agrupamento baseado em formigas();
9  until M;
```

A tarefa da formiga rainha é agregar os agrupamentos gerados pelas diversas colônias e calcular uma nova matriz de similaridade de acordo com a equação 4.33. $O = (o_1, o_2, ..., o_n)$ representa um conjunto de objetos, e um agrupamento destes $n$ objetos em $k$ grupos pode ser representado com um vetor $\lambda \in N^n$. $r$ são os grupos de agrupamentos, que a formiga rainha precisa analisar, com o $q^{th}$ agrupamento $\lambda^{(q)}$ tendo $k^{(q)}$ grupos. A nova matriz de similaridades é enviada para cada colônia, a qual continuará a re-agrupar, só que usando esta nova informação e um novo conjunto de dados. Este novo conjunto de dados é escolhido de acordo com as seguintes estratégias:

- KP (*keeping predecessor*): Cada colônia continua com seu último agrupamento resultante ;
- CE (*circular exchange*): Cada colônia escolhe o último agrupamento resultante da sua colônia vizinha pela direita (ou esquerda);
- LO (*lowest outliers*): Cada colônia escolhe o conjunto de dados resultante do último agrupamento, com o número mais baixo de objetos muito dissimilares dos demais.

Este procedimento é repetido até que os agrupamentos resultantes não tenham mais modificações.

$$Z = \frac{1}{r} H H^T \tag{4.33}$$

onde a matriz $H^T$ é a transposição da matriz $H$, $Z$ é a matriz esparsa *nxn*.

$$H = (H^{(1)} ... H^{(r)}) \tag{4.34}$$

Para validar o algoritmo, os autores utilizaram bases de dados reais e artificiais. Os resultados obtidos mostraram que o algoritmo obteve um desempenho superior ao *K-means* e ao algoritmo *Shelokar*. No entanto, o algoritmo apresenta uma limitação com relação a necessidade de se determinar *a priori* o número de colônias e também quanto a determinação de seus parâmetros.

### 4.1.3 Algoritmos inspirados por outros comportamentos baseados em inteligência de enxames

Outros comportamentos, além dos citados anteriormente nas seções 3.1 e 3.3, são utilizados como inspiração para o desenvolvimento de soluções para o problema de agrupamento de dados.

A seguir serão descritos alguns destes algoritmos e suas respectivas inspirações biológicas.

#### *4.1.3.1 AntClust*

Em Labroche, Monmarché e Venturini (2002) é proposto o algoritmo *AntClust*, um novo método para resolver o problema de agrupamento de dados baseado na modelagem do sistema de reconhecimento químico de formigas. Este sistema permite que formigas percebam as diferenças entre companheiras e intrusas, e então criem grupos homogêneos de indivíduos que compartilham o mesmo odor. O comportamento ocorre em quatro distintos níveis de análise para cada formiga:

1. A existência de um odor químico, parcialmente construído pela formiga, dependente da espécie e do ambiente;

2. Um mecanismo de recepção química, o qual permite a percepção do odor químico de outras formigas;

3. Um modelo de referência que indica o tipo de odor que uma companheira de ninho deveria possuir. Este modelo é aprendido e continuamente alterado;

4. Um conjunto de regras de decisões que dispara comportamentos de acordo com a similaridade entre os modelos de referência e o odor percebido em uma formiga.

De maneira simplificada, a cada iteração do algoritmo *AntClust* duas formigas, selecionadas aleatoriamente, encontram-se comparando seu odor e limiar de aceitação, os quais são modificados de acordo com as regras de decisão. Os grupos são formados de acordo com a similaridade entre os odores das formigas, o qual determina se uma formiga faz parte de um ou outro ninho. No final da execução, a reunião de formigas em um número de ninhos onde companheiras são mais similares do que cada formiga de outra colônia, obtém um particionamento do conjunto de objetos.

Para uma formiga $i$, os seguintes parâmetros são definidos:

- O rótulo $Label_i$ é determinado pelo ninho para o qual a formiga $i$ pertence e é simplesmente codificado por um número. No início as formigas não são influenciadas por qualquer ninho, então $Label_i$ é 0.

- O modelo é metade definido pelo odor genético $Genetic_i$ e metade por um limiar de aceitação $Template_i$ que é aprendido durante uma fase de inicialização. O limiar $Template_i$ resultante é uma função de todas as similaridades observadas durante este período de inicialização. $Template_i$ é dinâmico e alterado após cada encontro realizado pela formiga $i$.

- $M_i$ que reflete se a formiga $i$ é bem sucedida durante seus encontros com todas as formigas encontradas. Se uma jovem formiga não realizou nenhum encontro, $M_i = 0$ no tempo $t = 0$. $M_i$ é incrementado quando a formiga $i$ encontra uma outra formiga com o mesmo rótulo e diminuído quando o rótulo é diferente.

- $M_i^+$ que mede quão bem aceita é a formiga $i$ em seu ninho. Seu valor aumenta quando a formiga $i$ encontra outra com o mesmo rótulo e quando ambas aceitam-se entre si, e diminui quando não há aceitação entre as formigas.

- Uma idade $A_i$ que no início é igual a 0 e é usada quando o limiar de aceitação é alterado.

- $Max(Sim(i,\cdot))$ e $Sim(i,\cdot)$ similaridades observadas durante os encontros de $i$ com outras formigas.

O algoritmo simula encontros entre as formigas selecionando aleatoriamente duas formigas a cada iteração. Rótulos e limiares de aceitação são alterados de acordo com algumas regras comportamentais descritas a seguir:

1. Criação do ninho: Se $Label_i = Label\,j = 0$ e $Acceptance(i,j)$, então criar um novo $Label_{NEW}$ e $Label_j \longleftarrow Label_{NEW}$. Se $Acceptance$ é falso, então a regra 6 é aplicada.

   $Acceptance(i,j) \Leftrightarrow (Sim(i,j) > Template_i) \wedge (Sim(i,j) > Template_j)$

2. Atribuir uma formiga sem *Label* para um ninho: Se $(Label_i = 0 \wedge Label_j \neq 0)$ e $Acceptance(i,j)$, então $Label_i \longleftarrow Label_j$. O caso $(Label_j = 0 \wedge Label_i \neq 0)$ é trabalhado da mesma maneira.

3. Encontro *positivo* entre duas companheiras: Se $(Label_i = Label_j) \wedge (Label_i \neq 0) \wedge (Label_j \neq 0)$ e $Acceptance(i,j)$ então incrementar $M_i, M_j, M_i^+$, e $M_j^+$

   Para incrementar: $x \longleftarrow (1-\alpha) * x + \alpha$

   Para decrementar: $x \longleftarrow (1-\alpha) * x$

   onde $\alpha = 0.2$

4. Encontro *negativo* entre duas companheiras: Se $(Label_i = Label_j) \wedge (Label_i \neq 0) \wedge (Label_j \neq 0)$ e $Acceptance(i,j) = false$ então incrementar $M_i, M_j$, e decrementar $M_i^+$, e $M_j^+$. A formiga $x(x = i, x = j)$ que possui a integração errada no ninho $(x|M_x^+ = Min_{k \in [i,j]} M_k)$ perde seu rótulo e não possui mais ninho $(Label_x \longleftarrow 0, M_x \longleftarrow 0$ e $M_x^+ \longleftarrow 0)$

5. Encontro entre duas formigas de diferentes ninhos: Se $(Label_i \neq Label_j)$ e $Acceptance(i,j)$, então decrementar $M_i$ e $M_j$. A formiga $x$ com o mais baixo $M_x$ muda seu ninho, passando a pertencer ao ninho da formiga que encontrou.

6. Regra padrão: Se nenhuma outra regra é aplicada então nada acontece.

O AntClust foi comparado com o *K-means* através da aplicação em bases de dados reais e artificiais. Os resultados apontados pelo autor mostram que o AntClust obteve melhores resultados, podendo tratar tanto pequenas como grandes bases de dados. Porém mostrou-se ineficiente quando um importante número de grupos é esperado como resultado. Os autores acreditam que este comportamento ocorre porque existe apenas uma regra que pode criar novos ninhos.

#### *4.1.3.2 AntTree*

Uma interessante abordagem utilizada para resolver o problema de agrupamento foi apresentado por Azzag et al. (2003). Trata-se do algoritmo denominado *AntTree*.

O *AntTree* foi inspirado em um comportamento observado nas colônias de formigas onde estas estão aptas a construir estruturas por meio do auto-agrupamento dos seus corpos. As formigas podem, por exemplo, construir cadeias com seus corpos para conectar folhas, para atravessar de um espaço a outro ou para construir o ninho. Estas estruturas se separam após um determinado tempo, normalmente quando o objetivo que motivou sua construção foi atingido.

Deste comportamento os autores extraíram algumas propriedades para desenvolver o algoritmo *AntTree*:

- as formigas constróem este tipo de estruturas vivas começando de um suporte fixo (uma folha ou um galho);
- as formigas movem esta estrutura enquanto ela está sendo construída;
- a formiga pode alcançar qualquer posição da estrutura;
- a maioria das formigas que constituem a estrutura podem ser bloqueadas sem a menor possibilidade de ser retiradas. Um exemplo são as formigas posicionadas no meio da cadeia;

- algumas formigas (um número normalmente menor que a quantidade de formigas bloqueadas) que estão conectadas a estrutura podem se desconectar no momento que quiserem. Um exemplo são as formigas que estão alocadas no final da cadeia;

- pode-se observar um fenômeno tanto de crescimento da estrutura quanto de diminuição.

O principal princípio do *AntTree* é que cada formiga representa um nodo de uma árvore e conseqüentemente um dado a ser agrupado. O nodo raiz $a_0$ representa o suporte a partir do qual a árvore será construída. Formigas irão se fixar gradativamente neste nodo inicial e sucessivamente nas formigas fixadas a este nodo. Este comportamento se repete até que todas as formigas estejam fixadas a estrutura. Todos estes movimentos acontecem mediante um cálculo de similaridade $Sim(i, j)$ (equação 4.35). Então, para cada formiga $a_i$, $i \in [1, N]$ são definidos os seguintes conceitos:

- a conexão de saída de $a_i$ é aquela que $a_i$ pode manter em direção a outra formiga;

- as conexões de entrada de $a_i$ são aquelas que uma outra formiga pode manter em direção a $a_i$, como exemplo pode-se citar as pernas da formiga;

- o dado $d_i$ é representado pela formiga $a_i$;

- um limiar de similaridade $T_{Sim}(a_i)$ e um limiar de dissimilaridade $T_{Dissim}(a_i)$ será localmente alterado por $a_i$.

$$Sim(i, j) = 1 - \sqrt{\frac{1}{M} \sum_{k=1}^{M} (v_{ik} - v_{jk})^2} \tag{4.35}$$

onde $M$ significa o número de atributos numéricos usados para descrever cada dado, $v_{ik}$ é o valor do $k^{th}$ atributo do $i^{th}$ dado.

Durante o agrupamento da estrutura, cada formiga $a_i$ assumirá um dos dois comportamentos abaixo:

- movendo-se na árvore: $a_i$ movendo-se sobre o suporte $a_0$ ou sobre uma outra formiga indicada por $a_{pos}$, mas $a_i$ ainda não está conectada na estrutura. $a_i$ encontra-se completamente livre para mover-se no suporte ou em direção a outras formigas dentro de sua vizinhança. Se $a_{pos}$ indica a formiga onde $a_i$ está localizada, então $a_i$ se movimentará aleatoriamente para qualquer imediata posição vizinha de $a_{pos}$ na árvore;

- conectada a árvore: $a_i$ não pode mais ficar muito tempo desprendida da estrutura. Cada formiga tem somente uma conexão de saída em direção a outra formiga e não pode ter mais que $L_{max}$ conexões de entrada.

Detalhes sobre o *AntTree* encontram-se no Algoritmo 8. Em cada passo do algoritmo, uma formiga $a_i$ é selecionada, dentre uma lista de formigas, para se conectar na estrutura ou mover-se de acordo com a sua similaridade entre seus vizinhos. Para cada formiga $a_i$, dois casos precisam ser considerados. O primeiro caso é quando a formiga $a_i$ está conectada no suporte (Procedimento 9), se $a_i$ é suficientemente similar a $a^+$ de acordo com seu limiar de similaridade (onde $a^+$ é a formiga mais similar a $a_i$ entre todas as formigas já conectadas no suporte), $a_i$ é movida em direção a $a^+$ para ser agrupada na

mesma sub-árvore, ou seja, no mesmo grupo. Se $a_i$ não for suficientemente similar a $a^+$, verifica-se então se é suficientemente dissimilar a $a^+$. Se for, $a_i$ é conectada ao suporte. Isto significa que o algoritmo cria uma sub-árvore, onde formigas são o mais dissimilares possível das formigas de outras sub-árvores conectadas ao suporte. Finalmente, se $a_i$ não for suficientemente similar ou dissimilar a $a^+$, o algoritmo altera o seu limiar de acordo com as equações 4.36 e 4.37 para permitir que $a_i$ seja mais tolerante e aumente sua probabilidade de ser conectada na próxima vez que for considerada.

$$T_{Sim}(a_i) \longleftarrow T_{Sim}(a_i) * 0.9 \tag{4.36}$$

$$T_{Dissim}(a_i) \longleftarrow T_{Dissim}(a_i) + 0.01 \tag{4.37}$$

**Algorithm 8**: *Main*

```
1 todas as formigas são colocadas no suporte e seus limiares de similaridade e
  dissimilaridade respectivamente inicializados em 1 e 0;
2 while existe uma formiga a_i não conectada do
3     if a_i está localizada no suporte then
4         Procedure 1;
5     else
6         Procedure 2;
```

**Procedure 1** `Support case`

```
1 if a formiga ainda não está conectada ao suporte a_0 then
2     Conectar a_i em a_0;
3 else if Sim(a_i, a^+) ≥ T_Sim(a_i) then
4     mover a_i em direção a a^+;
5 else if Sim(a_i, a^+) < T_Dissim(a_i) then
6     conectar a_i ao suporte a_0 (ou, se não houver conexão disponível em a_0,
      diminuir T_Sim(a_i) e mover a_i em direção a a^+);
7 else
8     diminuir T_Sim(a_i) e incrementar T_Dissim(a_i);
```

O segundo caso é quando a formiga $_i$ está em cima de outra formiga ($a_{pos}$) (Procedure 10). Se houver uma conexão de entrada livre para $a_{pos}$ e se $a_i$ é suficientemente similar a $a_{pos}$ e suficientemente dissimilar as formigas conectadas em $a_{pos}$, então $a_i$ será conectada a $a_{pos}$. Neste caso, $a_i$ significa a raiz de uma subárvore, ou um subgrupo, abaixo de $a_{pos}$. Senão, $a_i$ é aleatoriamente movida em direção a um nodo vizinho de $a_{pos}$ e seu limiar é alterado de acordo com as equações 4.36 e 4.37. Desta forma $a_i$ se moverá no grafo para encontrar uma melhor localização para ser conectada.

Para avaliar e comparar os resultados obtidos com o *AntTree*, os autores utilizaram bases de dados representados por atributos numéricos (neste caso, um dado é um vetor de números reais).

**Procedure 2** `Ant case`

```
1 a_pos é a formiga na qual a_i está locada;
2 a_k é uma vizinha de a_pos selecionada aleatoriamente;
3 if Sim(a_i, a_pos) ≥ T_Sim(a_i) then
4     a^+ é a formiga vizinha de a_pos a qual é mais similar para a_i;
5     if Sim(a_i, a^+) < T_Dissim(a_i) then
6         conectar a_i a a_pos (ou, se não houver conexão disponível em a_pos, mover a_i
          em direção a a_k);
7     else
8         diminuir T_Dissim(a_i), incrementar T_Sim(a_i) e mover a_i em direção a a_k;
```

## 4.2 Algoritmos distribuídos

Dentre os algoritmos distribuídos investigados para a realização deste trabalho descreve-se o *Improved Probabilistic Ant based Clustering* (I-PACE) (CHANDRASEKAR; SRINIVASAN, 2007), por se tratar de um algoritmo para agrupamento distribuído de dados inspirado em inteligência de enxames. O I-PACE foi desenvolvido para superar algumas desvantagens encontradas pelos autores no algoritmo *Probabilistic Ant based Clustering* (PACE) (CHANDRASEKAR; VIJAYKUMAR; SRINIVASAN, 2006).

O I-PACE utiliza como inspiração biológica o comportamento de organização de cemitérios, descrito na Seção 3.1, juntamente com um comportamento de reconhecimento de formigas da mesma colônia por meio de um odor químico. Assim, são formados grupos de formigas que carregam dados relacionados.

Considerando-se uma base de dados distribuída entre $n$ localizações, o I-PACE realiza o agrupamento através da formação de várias zonas nestas localizações de acordo com três fases: fase de consulta a base de dados, fase de atribuição de probabilidades e a fase de agrupamento.

Na fase de consulta a base de dados, um conjunto de palavras-chave é extraído de uma consulta a base de dados realizada pelo usuário. A seguir, o algoritmo realiza a computação do número de ocorrências de cada palavra-chave nas $n$ localizações e parte para a realização da fase de atribuição das probabilidades. Nesta fase, o algoritmo realiza a computação das probabilidades relacionadas as ocorrências destas palavras-chave em cada localização. Logo após, na fase de agrupamento, cada uma das $n$ localizações é dividia em diversas regiões denominadas zonas. Os agentes são distribuídos entre estas várias zonas e estão livres para navegar dentro de sua própria zona para buscar dados a serem agrupados. Enquanto exploram suas zonas, os agentes comportam-se segundo o algoritmo ACA descrito na Seção 3.1, o qual é inspirado pelo comportamento de organização de cemitérios exibido por algumas espécies de formigas. Desta forma, ao encontrar um dado, o agente possui uma probabilidade de coletá-lo e ao carregar um dado, o agente tem uma probabilidade de depositá-lo. Estas probabilidades consideram a densidade da região onde o agente está navegando.

Em seguida o algoritmo aplica uma função de vizinhança que é utilizada para calcular a similaridade existente entre os grupos. Logo após, realiza o processo de aglomeração intra-zona, onde ocorre a junção dos grupos similares dentro de cada zona, seguida pela fase de aglomeração inter-zona, onde ocorre a junção dos grupos entre as diversas zonas de uma localização. Finalmente, o algoritmo realiza o processo de aglomeração sobre todos os grupos de dados de todas as localizações para assim obter um único agrupamento.

## 4.3 Outros algoritmos relacionados

Durante o desenvolvimento do algoritmo proposto encontrou-se inspiração também em algoritmos desenvolvidos para solucionar o problema de *ensemble* de agrupamento. O problema de *ensemble* de agrupamento de dados é um subconjunto do problema de agrupamento de dados. Ele consiste em combinar o resultado de vários algoritmos de agrupamento obtidos do mesmo conjunto de dados, em um resultado unificado. Este resultado é obtido sem a utilização dos dados originais que foram usados para gerar o conjunto de soluções obtidas dos vários algoritmos de agrupamento (STREHL; GHOSH, 2002).

A seguir descreve-se os algoritmos para *ensemble* relacionados ao algoritmo proposto.

### 4.3.1 Algoritmo para *ensemble* distribuído

O trabalho apresentado em Agogino e Tumer (2006) propõe um método de *ensemble* distribuído de agrupamento de dados baseado em agentes. Dado um conjunto de agrupamentos resultantes da aplicação de algoritmos para agrupamento de dados, o método deve encontrar um agrupamento final (unificado) que melhor caracteriza estes resultados sem precisar utilizar os dados originais. Para isso, pequenos subconjuntos de dados destes agrupamentos são atribuídos a cada agente que deverá votar para qual grupo seus dados devem pertencer no agrupamento final. Através de aprendizado por reforço, os agentes aprendem a votar de maneira que atinjam um resultado melhor. No final do processo os dados irão pertencer aquele grupo que receber a maior quantidade de votos dos agentes.

O agrupamento final é avaliado por meio da maximização de uma utilidade global, a qual é computada de maneira distribuída. No entanto, maximizar uma utilidade global diretamente não se caracteriza um problema trivial em um cenário multiagente já que as ações de outros agentes também influenciam no resultado desta utilidade. Assim, os autores propuseram a maximização de uma utilidade específica do agente denominada utilidade privada. Desta forma para que a maximização desta utilidade privada resultasse na maximização da utilidade global ela deveria ser fatorada. Uma utilidade privada é fatorada quando qualquer ação de um agente para aumentar sua utilidade privada conduza a um aumento da utilidade global. A Equação 4.38 mostra como se calcula a chamada "utilidade da diferença" utilizada pelos autores como inspiração para auxiliar os agentes a maximizarem a utilidade global do agrupamento final. Na equação $G$ é a utilidade global, $G(z)$ é a utilidade considerando as ações de todos os agentes e $G(z-i)$ é a utilidade considerando todos os agentes exceto o agente $i$.

$$D_i(z) = G(z) - G(z-i) \tag{4.38}$$

A utilidade global de um agrupamento final pode ser computada de maneira distribuída através do compartilhamento dos votos de todos os agentes via *broadcasting*. Assim, conhecendo os votos de todos os agentes, cada agente pode computar o agrupamento final e também sua utilidade privada, a qual poderá ser também compartilhada por *broadcasting* com todos os demais agentes e todos poderão finalmente computar a utilidade global do agrupamento.

A utilidade da diferença foi utilizada no algoritmo *bee clustering* para ajudar os agentes no processo de tomada de decisão sobre a troca de grupo. Porém, como os agentes do algoritmo proposto trabalham de maneira distribuída, a utilidade a ser maximizada é uma utilidade local, a qual será detalhada na Seção 5.1.

### 4.3.2 MOCLE

Outro trabalho da área de *ensemble* de agrupamento relacionado ao algoritmo proposto é o algoritmo *Multi-Objective Clustering Ensemble* (MOCLE) (FACELI; CARVALHO; SOUTO, 2006). O MOCLE preocupa-se em realizar a análise de agrupamento final, a partir de um conjunto de agrupamentos, baseando-se na combinação multi-objetivo [2] de algoritmos de agrupamento.

Dado um conjunto com $n$ objetos $X = \{x_1, x_2, ..., x_n\}$, o MOCLE primeiro constrói um conjunto de partições iniciais, $\Pi = \{\pi^1, \pi^2, ..., \pi^r\}$, onde $r$ é o número de partições iniciais e $\pi^i = \{c_1^i, c_2^i, ..., c_{K^i}^i\}$ é uma partição de $X$ em $K^i$ grupos tal que $\bigcup_{j=1}^{K^i} c_j^i = X$. O conjunto de partições $\Pi$ é construído a partir dos resultados de diferentes algoritmos de agrupamento de dados que otimizem diferentes critérios, com diferentes valores para seus parâmetros. Como não se conhece *a priori* o nível de refinamento de cada possível estrutura dos dados utilizados, os diferentes valores de parâmetros usados para os algoritmos fornecem partições com diferentes níveis de refinamento, ou seja, diferentes número de grupos, grupos com diferentes densidades, entre outros. Desta forma, os autores esperam que a população inicial $\Pi$ contenha bons particionamentos de acordo com as diversas possibilidades de critérios de agrupamentos, capturando diferentes estruturas nas bases de dados.

O conjunto de partições $\Pi$ é utilizado pelo MOCLE como população inicial para um algoritmo genético multi-objetivo o qual seleciona as melhores partições de acordo com um critério empregado e combina estas partições. A seleção das melhores partições é realizada segundo dois objetivos complementares: a variância intra-*cluster* e a conectividade. A variância indica quão próximos os dados se encontram em um grupo ou agrupamento, enquanto que a conectividade reflete a frequência com que dados vizinhos são colocados no mesmo grupo. Estes dois objetivos equilibram suas tendências de aumentar ou diminuir com o número de grupos, o que é importante para a exploração do espaço de soluções evitando a convergência para soluções triviais.

Desta forma o algoritmo MOCLE é capaz de explorar a diversidade de possíveis soluções por meio de uma população inicial e gerar um conjunto de soluções alternativas ao invés de uma solução simples.

A justificativa para se apresentar o algoritmo MOCLE como um dos trabalhos relacionados ao algoritmo proposto está ligada a dois motivos principais. O primeiro está relacionada à uma das motivações que deu origem ao MOCLE - a aplicação da análise de agrupamento para descoberta de sub-classes, pois pretende-se investigar, como trabalho futuro, a eficácia do algoritmo *bee clustering* na descoberta de estruturas alternativas em bases de dados de bioinformática, mais especificamente na análise das funções dos genes e na descoberta de subtipos de doenças.

A segunda justificativa está relacionada a proposta do algoritmo MOCLE em realizar o *ensemble* de acordo com múltiplos objetivos, pois outra direção futura de pesquisa relacionada a aplicação do algoritmo *bee clustering* também está ligada a otimização de múltiplos objetivos. Em Santos, Oliveira e Bazzan (2009) apresentou-se o algoritmo *Multi Ant Colony Clustering* (MACC), desenvolvido para resolver o problema de agrupamento de dados considerando a otimização de múltiplos objetivos. O MACC, que foi inspirado em colônias de formigas, apresentou resultados promissores o que nos leva a acreditar que a utilização de inteligência de enxames para o problema de agrupamento multi-objetivo

[2] Abordagens multi-objetivo consistem na otimização direta de vários objetivos (critérios de agrupamento) simultaneamente (HANDL; KONWLES, 2005).

seja viável.

### 4.3.3 K-means

Ele busca minimizar a distância dos elementos a um conjunto de $k$ centros de forma iterativa. A distância entre um ponto $p_i$ e um conjunto de grupos, dada por d($p_i$,$\chi$), é definida como sendo a distância do ponto ao centro mais próximo dele. A função a ser minimizada é dada por:

$$d(p_i,\chi) = \frac{1}{n}\sum_{i=1}^{n} d(p_i,\chi)^2$$

O algoritmo depende de um parâmetro ($k$=número de grupos) definido pelo usuário e exige portanto algum conhecimento prévio. Isto costuma ser um problema, tendo em vista que normalmente não se sabe quantos grupos existem *a priori*.

O algoritmo *K-means* pode ser descrito como segue:

1. Escolher aleatoriamente $k$ distintos valores iniciais para centros $z_1, z_2, ..., z_k$ dos grupos de $n$ pontos $x_1, x_2, ..., x_n$;

2. Associar o ponto $x_i$, $i = 1,2,...,n$ ao grupo $C_j$, $j \in (1,2,...,k)$ se
   $\|x_i - z_j\| < \|x_i - z_p\|$, $p = 1,2,...,k$ e $j \neq p$ mais próximo;

3. Recalcular novos centros $z_1^*, z_2^*, ..., z_k^*$ de cada grupo de acordo com a seguinte equação:
   $$z_i^* = \frac{1}{n_i}\sum_{x_j \in C_i} x_j \tag{4.39}$$
   onde $i = 1,2,...,k$ e $n_i$ é o número de elementos pertencentes ao grupo $C_i$;

4. Repetir os passos 2 e 3 até que nenhum elemento mude de grupo.

As principais vantagens deste algoritmo é a sua simplicidade e o fato de ser extremamente veloz, convergindo em poucas iterações para uma configuração estável, na qual nenhum elemento está designado para um grupo cujo centro não seja o mais próximo.

Um eventual problema é que pode ocorrer uma indevida separação dos grupos no caso de se definir uma má inicialização dos centros, já que esta é feita de forma arbitrária no início da execução. Outro ponto importante que pode afetar a qualidade dos resultados é a necessidade de se escolher *a priori* o número de grupos desejados (JAIN; MURTY; FLYNN, 1999). Uma pequena quantidade de grupos pode causar a junção de dois grupos naturais, enquanto que um número muito grande pode fazer com que um grupo natural seja quebrado em dois. É devido a estes fatores que diversos métodos são desenvolvidos para resolver o problema de agrupamento de dados com o intuito de superar tais desvantagens apontadas pelo algoritmo *K-means*.

Um destes métodos é o algoritmo ISODATA, tão conhecido quanto o algoritmo *K-means*. O ISODATA realiza o agrupamento de dados iterativamente da mesma forma que o *k-means*, no entanto ele apresenta complementarmente a opção de aumentar o número de grupos segundo um limite estipulado.

### 4.3.4 Average Link

Um importante representante do método de agrupamento de dados hierárquico aglomerativo é o conhecido algoritmo *Average link*, o qual contróe uma matriz de distâncias entre os grupos a partir do conjunto de dados que deve ser agrupado (JAIN; DUBES, 1988).

No *Average link* cada dado é inicialmente tratado como um grupo, e em seguida, a cada iteração, o par de grupos mais similares é unificado em um único grupo e a matriz de distâncias é atualizada. A similaridade entre os dois grupos é calculada de acordo com a métrica de ligação média tratada na subseção 2.2.1. A condição de parada do algoritmo pode ser, por exemplo, quando o correto número de grupos for encontrado ou quando quando formar um único grupo.

Os principais problemas apresentados pelo algoritmo *Average link* são inerentes ao método hierárquico e estão relacionados aos seguintes aspectos abaixo relacionados:

- alta complexidade computacional $O(n^2)$;
- o algoritmo não é capaz de corrigir uma associação incorreta pois após um dado ser alocado em um determinado grupo, ele não será considerado novamente;
- definição *a priori* do número de grupos para uma correta convergência do algoritmo no caso de se optar pela determinação do número de grupos como critério de parada do algoritmo.

## 4.4 Estudo comparativo

Para se fazer uma análise comparativa dos algoritmos para agrupamento de dados apresentados neste trabalho, é importante se considerar o domínio para o qual o algoritmo foi proposto. Portanto, definiu-se alguns critérios de comparação dependentes de domínio e alguns não dependentes de domínio.

A seguir destaca-se os critérios dependentes de domínio utilizados para comparação dos algoritmos citados nas seções anteriores:

- Domínio de aplicação: Os algoritmos *Shelokar* (SHELOKAR; JAYARAMAN; KULKARNI, 2004), ACOC (KAO; CHENG, 2006), SACA (LUMER; FAIETA, 1994), AntClust (LABROCHE; MONMARCHÉ; VENTURINI, 2002), AntTree (AZZAG et al., 2003) e I-PACE (CHANDRASEKAR; SRINIVASAN, 2007) não especificaram explicitamente qual a sua aplicação, porém seus resultados foram obtidos mediante experimentos em bases de dados numéricos. O $A^2CA$ (VIZINE et al., 2005) obteve seus resultados através de base de dados numéricos e binários e também utilizou uma base de dados na área de bioinformática. O AntClass (MONMARCHÉ; SLIMANE; VENTURINI, 1999) foi desenvolvido com o propósito de agrupar dados numéricos. Já o CSIM (BIN et al., 2002) objetiva o agrupamento de documentos. O algoritmo proposto por Yang e Kamel (2006) também não especifica o domínio de aplicação, mas foi testado em bases de dados numéricos e uma base textual de documentos. Os algoritmos MOCLE (FACELI; CARVALHO; SOUTO, 2006) e *ensemble* distribuído (AGOGINO; TUMER, 2006) dispensam tal classificação pois não atuam diretamente nas bases de dados e sim nos agrupamentos resultantes de outros algoritmos.

- Representação gráfica dos resultados: Dentre os algoritmos apresentados nas seções anteriores, o $A^2CA$ (VIZINE et al., 2005), o AntTree (AZZAG et al., 2003) e o MOCLE (FACELI; CARVALHO; SOUTO, 2006) apresentam uma forma de representação gráfica do resultado obtido para o usuário. O $A^2CA$ usa um *grid* bidimensional para representar os dados enquanto que o AntTree utiliza uma estrutura de árvore. Ambos realizaram seus testes em bases de dados onde o número de grupos no qual os dados serão particionados é pequeno (entre 1 e 10), o que viabiliza o tipo de representação gráfica utilizada. Porém, para a representação do agrupamento de uma base de dados onde o número de grupos é bem maior, a representação gráfica fica comprometida. Já o algoritmo MOCLE (FACELI; CARVALHO; SOUTO, 2006) utiliza um esquema de cores para colorir um agrupamento gerado associando cores a cada um de seus grupos e consequentemente aos dados pertencentes a estes grupos.

- Quantidade de parâmetros que devem ser inicializados pelo usuário: É bastante difícil encontrar parâmetros ótimos para os algoritmos, especialmente porque estes são, em geral, dependentes do conjunto de dados que se está manipulando. Se os parâmetros não forem ótimos, então os resultados têm grandes chances de serem incorretos. Os algoritmos *K-means*, algoritmo *Shelokar* (SHELOKAR; JAYARAMAN; KULKARNI, 2004) e o ACOC (KAO; CHENG, 2006) necessitam da definição do parâmetro $k$ (número de grupos). O algoritmo *Shelokar* e ACOC necessitam, ainda, da inicialização do parâmetro $q_0$ (um valor probabilístico definido *a priori*). O ACOC também possui um parâmetro $\beta$ (parâmetro que especifica o peso relativo de $n_{ij}^k$), o qual deve ser maior que 0. O SACA (LUMER; FAIETA, 1994) necessita da inicialização dos parâmetros $\alpha \in [0,1]$ (um fator que define a escala de dissimilaridade), $k_1$ e $k_2$ (limiares constantes). O $A^2CA$ (VIZINE et al., 2005) possui os seguintes parâmetros $s_{max}^2$ (maior campo de visão possível da formiga) e $\theta$ (constante). O CSIM (BIN et al., 2002) apresenta um limiar constante $k$, $\alpha \in [1,10]$ (coeficiente de similaridade), *antNumber* (número de formigas) e $n$ (número de iterações). O algoritmo proposto por (YANG; KAMEL, 2006) depende da inicialização da variável $c_x$ (usada para acelerar a convergência do algoritmo), dos parâmetros $\beta$ (coeficiente de similaridade), $\mu$ (coeficiente de dissimilaridade), $N$ (número de colônias), $M$, $M_m$ (número máximo de iterações) e *antNumber* (número de formigas em cada colônia). O algoritmo AntClass (MONMARCHÉ; SLIMANE; VENTURINI, 1999) possui uma grande quantidade de parâmetros, porém estes são inicializados automaticamente. O valor de cada parâmetro é atribuído aleatoriamente dentro de um intervalo pré-definido pelos autores. O algoritmo I-PACE (CHANDRASEKAR; SRINIVASAN, 2007) depende da formulação de uma consulta pelo usuário para extrair palavras-chave as quais serão utilizadas na realização do agrupamento.

Abaixo, apresenta-se os critérios definidos que são independentes de domínio:

- Definição do número de grupos *a priori*: Trata-se de uma importante característica a ser observada nos algoritmos para agrupamento de dados. Geralmente não se tem conhecimento de quantos grupos serão necessários para que se obtenha o melhor agrupamento. Um número definido incorretamente comprometerá diretamente o resultado do algoritmo. Os algoritmos *K-means*, algoritmo *Shelokar* (SHELOKAR; JAYARAMAN; KULKARNI, 2004) e o ACOC (KAO; CHENG, 2006) são dependentes de uma definição *a priori* do número de grupos nos quais os dados

serão particionados. O algoritmo proposto por Yang e Kamel (2006) necessita da definição *a priori* do número de colônias que relizarão o processo de agrupamento.

- Utilização de elementos centralizadores: O algoritmo proposto por (YANG; KAMEL, 2006) possui uma matriz de feromônio $Z$ e uma matriz $H$. Além disso, apresenta um agente, denominado formiga rainha, que agrega os agrupamentos gerados pelas diversas colônias do modelo, e calcula uma nova matriz de similaridade. É possível perceber que o resultado gerado pelo algoritmo é totalmente dependente do trabalho deste agente. O algoritmo *Shelokar* (SHELOKAR; JAYARAMAN; KULKARNI, 2004) possui uma matriz de feromônio $NxK$ ($N$ =número de objetos para agrupar e $K$ =número de grupos), uma matriz de centros $m$ de tamanho $Kxn$ ($n$ =número de atributos), uma matriz de pesos $w$ de tamanho $NxK$. O ACOC (KAO; CHENG, 2006) possui uma matriz $mxg$ ($m$ =número de objetos para agrupar e $g$ =número de grupos), uma matriz de centros $C^k$ de tamanho $mxn$ ($n$ =número de atributos), uma matriz de pesos $W^k$ de tamanho $mxg$, uma matriz de objetos $X$ de tamanho $mxn$, uma lista $tb^k$ que representa a memória da formiga e uma matriz de feromônio.$PM$.

- Existência de agentes com ampla visão do ambiente: No algoritmo ACOC (KAO; CHENG, 2006), cada agente possui uma memória que representa a lista de objetos agrupados para evitar que a formiga agrupe um objeto mais de uma vez. Já no AntClass (MONMARCHÉ; SLIMANE; VENTURINI, 1999), os agentes formigas não possuem uma visão tão ampla do sistema. Sua memória limita-se as quatro últimas pilhas de objetos encontradas. Mas é uma informação que interfere na convergência do algoritmo.

- Algoritmos Híbridos: Os algoritmos CSIM (BIN et al., 2002) e AntClass (MONMARCHÉ; SLIMANE; VENTURINI, 1999) utilizam, além da abordagem inspirada em insetos sociais, o algoritmo *K-means* com o objetivo de melhorar os resultados obtidos.

Para uma melhor visualização e compreensão das principais características dos algoritmos apresentados neste capítulo, optou-se por resumilas mostrando-as na Tabela 4.1.

Tabela 4.1: Comparação entre os algoritmos de grupamento de dados. (1) Aplicação, (2) Representação gráfica, (3) Parâmetros, (4) Definição *a priori* de informações, (5) Visão do ambiente, (6) Utilização de elementos centralizadores, (7) Algoritmo Híbrido.

| Algoritmo | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| *K-means* | numéricos | não | 1 | sim (número de grupos) | não | não | não |
| *Shelokar* | numéricos | não | 2 | sim (número de grupos) | não | sim (3 estruturas de dados) | não |
| ACOC | numéricos | não | 3 | sim (número de grupos) | sim | sim (6 estruturas de dados) | não |
| SACA | numéricos | não | 3 | não | não | não | não |
| $A^2CA$ | numéricos e binários | sim | 2 | não | não | não | não |
| AntClass | numéricos | não | 0 | não | parcial | não | sim (*K-means*) |
| CSIM | documentos | não | 4 | não | não | não | sim (*K-means*) |
| Múltiplas Colônias | numéricos e documentos | não | 7 | sim (número de colônias) | não | sim (2 estruturas de dados e um agente centralizador) | não |
| AntClust | numéricos | não | 0 | não | não | não | não |
| AntTree | numéricos | sim | 0 | não | não | não | não |
| I-PACE | numéricos | não | 1 | consulta do usuário para extração das palavras-chave | não | não | não |
| *Ensemble* distribuído | - | não | 0 | não | não | não | não |
| MOCLE | - | sim | 0 | não | não | não | sim (algoritmo genético para gerar população inicial) |

Ao se realizar a análise comparativa destas abordagens descritas anteriormente, percebeu-se que os algoritmos para agrupamento de dados inspirados em inteligência de enxames apresentam alguns problemas que são inerentes a inspiração biológica utilizada. Todos os algoritmos descritos que são inspirados em ACO são dependentes de definição *a priori* do número de grupos em que os dados serão divididos, o que se caracteriza um problema quando não se tem disponível este tipo de informação. Outra importante questão está relacionada com sua dependência de estruturas de dados, no caso a matriz de feromônio, o que além de centralizar a informação constitue-se um ponto único de falha. Já os algoritmos inspirados em organizações de cemitério não necessitam definir o número de grupos antecipadamente, tal informação emerge durante o processo de agrupamento. Porém, estes algoritmos tem como característica o fato de não gerarem explicitamente um particionamento, mas sim uma distribuição espacial dos objetos de dados. Tal distribuição espacial pode conter grupos que são óbvios para o observador humano, mas uma avaliação do agrupamento obtido só é possível mediante a recuperação destes grupos com o uso de uma ferramenta, o qual não se constitue uma tarefa trivial sem a interação humana.

## 4.5 Conclusão

Neste capítulo apresentou-se os principais algoritmos relacionados ao *bee clustering*. Tais algoritmos foram apresentados respeitando-se três categorias: algoritmos inspirados em inteligência de enxames, algoritmos distribuídos e outros algoritmos relacionados. Os algoritmos inspirados em inteligência de enxames por sua vez foram subdivididos em três classes de acordo com a inspiração biológica utilizada: algoritmos inspirados em ACO, algoritmos inspirados em organização de cemitérios e algoritmos inspirados em outros comportamentos relacionados a insetos sociais. Finalmente realizou-se uma breve análise comparativa dos trabalhos relacionados apontando suas principais características.

Como pode ser visto, o principal problema encontrado em alguns dos algoritmos apresentados neste capítulo é o fato de que seus resultados dependem tanto de informações importantes que precisam ser definidas *a priori* como de estruturas centralizadoras. Tais estruturas são constantemente acessadas e modificadas ao longo do processo de agrupamento e armazenam informações necessárias para a correta formação dos grupos, inviabilizando a aplicação destes algoritmos em domínios inerentemente distribuídos.

# 5 ABORDAGEM PROPOSTA

Nos Capítulos 2, 3 e 4 realizou-se uma revisão bibliográfica sobre os temas relacionados ao presente trabalho. Especificamente no Capítulo 4 reuniu-se os principais algoritmos para agrupamento de dados relacionados ao algoritmo proposto e realizou-se uma análise comparativa destes algoritmos. De acordo com este estudo comparativo percebeu-se que o principal problema detectado na maioria dos algoritmos apresentados encontra-se no fato de que seus resultados dependem tanto de informações importantes que precisam ser definidas *a priori* (número de grupos, densidade destes grupos, entre outros) como de estruturas centralizadoras. Tais estruturas são constantemente acessadas e modificadas ao longo do processo de agrupamento e armazenam informações necessárias para a correta formação dos grupos, inviabilizando a aplicação destes algoritmos em domínios inerentemente distribuídos, como por exemplo a Internet onde existem bases cujos dados não estão centralizados em uma única localização por motivos de segurança, privacidade, entre outros.

Desta forma, na tentativa de superar os problemas acima mencionados, o presente capítulo descreve detalhadamente o algoritmo *bee clustering*, o qual tem como objetivo formar, distribuidamente, grupos de agentes com características similares sem qualquer informação inicial relacionada ao resultado desejado, ou mesmo a necessidade do uso de parâmetros complexos.

O algoritmo *bee clustering* inspira-se principalmente no modelo matemático de recrutamento em colônias de abelhas descrito na Seção 3.4. Na natureza, as abelhas realizam a dança do recrutamento com o objetivo de recrutar outras abelhas para coletarem alimento em uma fonte de néctar de boa qualidade. Esta dança é utilizada pelo *bee clustering* para formar grupos de agentes. No entanto, no algoritmo proposto os agentes dançam para recrutar novos indivíduos para participarem de seus grupos. Através deste comportamento, os agentes do algoritmo são capazes de se organizarem em grupos de acordo com suas características ou habilidades e de maneira distribuída.

Assim, cada agente do *bee clustering* representa um objeto que precisa ser agrupado. Desta forma, o conjunto de atributos de um determinado objeto constitui o conjunto de características daquele agente que o representa. Estas características são utilizadas no processo de formação dos grupos para se avaliar a similaridade existente entre os agentes.

A Figura 5.1 mostra uma visão geral de como acontece todo o processo de agrupamento. Cada agente possui um conjunto de possíveis estados $\delta \in \mathscr{S} = \{d, v, w\}$ os quais indicam sua ações como descritas a seguir:

- $\delta = d$: significa que o agente está dançando com o objetivo de recrutar/convidar outros agentes a fazerem parte de seu grupo;
- $\delta = v$: indica que o agente está visitando um outro agente;

- $\delta = w$: significa que o agente está assistindo a dança do recrutamento com o objetivo de escolher um novo agente para visitar.

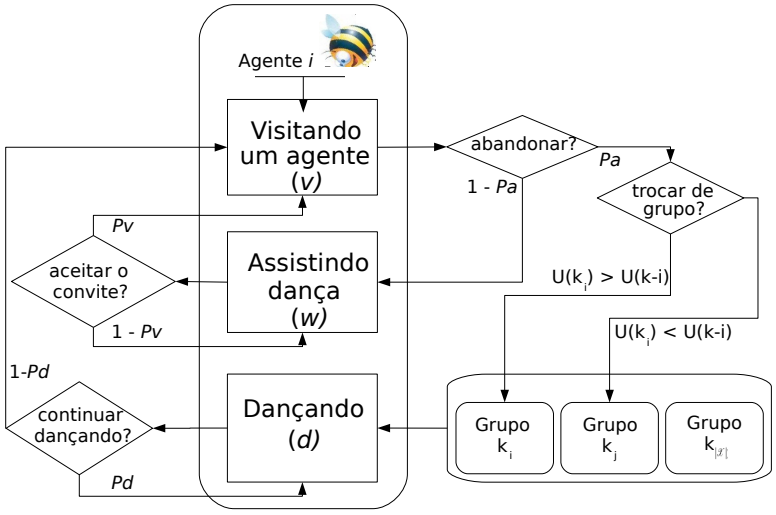


Figura 5.1: Processo de agrupamento realizado pelo algoritmo *bee clustering*

A troca de estados dos agentes acontece mediante algumas tomadas de decisões ilustradas pelos losangos na figura 5.1. Cada processo de tomada de decisão será abordado em detalhes na próxima Seção 5.1.

A Tabela 5.1 resume os principais parâmetros utilizados pelo *bee clustering*, onde: $\mathscr{A}$, que representa o conjunto de todos os agentes, possui tamanho $|\mathscr{A}|$; $\mathscr{X}$ que representa o conjunto das características dos agentes tem tamanho $|\mathscr{X}|$; $\mathscr{S}$, descrito anteriormente, representa o conjunto dos possíveis estados de cada agente; *Pa* indica a probabilidade de um agente abandonar outro de acordo com suas similaridades, *Pv* indica a probabilidade de um agente visitar outro, *Pd* representa a probabilidade de um agente continuar executando a dança do recrutamento a fim de convidar outros indivíduos para fazerem parte de seu grupo, $U(k)$ é o valor da utilidade do grupo $k$ indicando a qualidade deste grupo, $n$ indica a quantidade de agentes participantes de um grupo, $D$ representa a quantidade de agentes que estão realizando a dança do recrutamento pra um grupo em um determinado instante $t$, $c$ é um elemento calculado que representa o centróide de um grupo e $\mu$ é um parâmetro que controla o número máximo de iterações do algoritmo.

Os agentes do algoritmo proposto possuem um conhecimento limitado com relação aos demais agentes do modelo. Tal conhecimento está relacionado ao conjunto de seus estados $\mathscr{S} = \{d, v, w\}$. Quando o estado do agente é $\delta = v$ ou $\delta = d$ ele conhece apenas aqueles agentes que estão alocados em seu grupo no instante atual ($t$), ou seja, a partir do momento que um determinado agente deixa de participar de um grupo, ele é esquecido por todos os demais elementos daquele grupo e passa a ser conhecido somente pelos participantes de seu novo grupo. Da mesma forma, um agente que abandona um grupo não armazena informações relacionadas ao seu passado para que sejam utilizadas posteriormente em suas tomadas de decisões. Quando o estado do agente é $\delta = w$, ele tem conhecimento sobre os indivíduos que estão no seu grupo e também sobre aquele subconjunto de agentes cujo estado é $\delta = d$.

Tabela 5.1: Principais parâmetros do algoritmo *bee clustering*

| Parâmetro | Descrição |
|---|---|
| $\mathscr{A}$ | conjunto de agentes (tamanho $\|\mathscr{A}\|$) |
| $\mathscr{X}$ | conjunto de atributos do agente (tamanho $\|\mathscr{X}\|$) |
| $\mathscr{S} = \{v, w, d\}$ | conjunto dos estados dos agentes |
| | $v$=visitando, $w$=assistindo, $d$=dançando |
| $Pa$ | probabilidade de abandonar um agente |
| $Pv$ | probabilidade de visitar um agente |
| $Pd$ | probabilidade de continuar dançando |
| | para um grupo |
| $U(k)$ | utilidade do grupo $k$ |
| $n$ | número de agentes em um grupo |
| $D$ | número de agentes dançando para um grupo |
| $c$ | centróide de um grupo |
| $\mu$ | controla o número de iterações do algoritmo |

No início do processo de agrupamento, cada agente é considerado como um grupo. A seguir ocorrem visitas entre os agentes as quais possibilitam a verificação da similaridade existentes entre eles. Durante estas visitas os agentes decidem se deixam de participar de seu grupo para fazer parte do grupo de outros agentes. Além da decisão sobre troca de grupo, os agentes precisam decidir também sobre outras ações importantes para o processo de formação de grupos. Todas as questões relacionadas às decisões que os agentes devem tomar serão descritas detalhadamente a seguir na Seção 5.1.

## 5.1 Tomada de decisão

Durante o processo de agrupamento, os agentes precisam tomar algumas decisões importantes relacionadas à troca de grupo, a abandonar ou não um agente que está visitando, a aceitar ou não um convite pra visitar um agente e a dançar ou não para recrutar agentes para o seu grupo. Excetuando-se a decisão sobre troca de grupo, todas as demais são tomadas de maneira probabilística. Os losangos mostrados na figura (Figura 5.1) representam os momentos em que estas decisões precisam ser tomadas e serão explicados a seguir com maiores detalhes.

### 5.1.1 Decisão sobre abandonar um agente

A primeira decisão probabilística que um agente precisa tomar no processo de agrupamento está relacionada a abandonar ou não o agente que ele está visitando. Para tomar esta decisão, é necessário que o agente verifique se existem semelhanças entre suas característica e as características do agente que está visitando. Esta verificação é realizada com a utilização da métrica da distância Euclidiana.

Supondo-se então que o agente $i$ esteja visitando o agente $j$. $i$ verifica quão similar é ao agente $j$ calculando a distância Euclidiana existente entre seu conjunto de atributos $\mathscr{X}$ e os atributos do agente $j$ utilizando a Equação 5.1, onde $d(i, j)$ é a distância Euclidiana entre $i$ e $j$ e $|\mathscr{X}|$ representa o número de atributos de $i$ e $j$.

$$d(i,j) = \sqrt{\sum_{x=1}^{|\mathscr{X}|} (i_x - j_x)^2} \tag{5.1}$$

O agente $i$ utiliza o resultado obtido com o cálculo da distância Euclidiana, $d(i,j)$, como a probabilidade *Pa* de abandonar o agente $j$. Para isso, $d(i,j)$ é normalizado e *Pa* é calculado de acordo com a Equação 5.2, onde $\sqrt{|\mathscr{X}|}$ representa a maior distância Euclidiana possível entre os agentes, já que seus atributos foram normalizados para o intervalo [0,1].

$$Pa = \frac{d(i,j)}{\sqrt{|\mathscr{X}|}} \tag{5.2}$$

Quanto menor o valor de *Pa*, menor é a probabilidade do agente $i$ abandonar o agente $j$, o que significa que os dois agentes possuem um alto índice de similaridade. Por outro lado, quanto maior o valor de *Pa*, maior é a probabilidade de $i$ abandonar $j$ já que $i$ e $j$ são agentes com baixo índice de similaridade.

É importante salientar que uma vez que uma decisão sobre abandonar ou não um indivíduo seja tomada, esta é válida somente para aquele instante $t$, ou seja, se em um instante futuro $t+1$ o mesmo indivíduo for visitado, o agente verificará novamente se o abandona ou não sem qualquer referência as escolhas efetuadas em momentos passados.

A decisão que os agentes devem tomar sobre abandonar ou não um indivíduo é importante pois seu resultado influencia suas próximas ações. Ela acontece imediatamente antes do agente decidir se troca de grupo para ajudá-lo a evitar desperdício de tempo decidindo sobre participar ou não de um grupo nos quais seus indivíduos não possuam características similares as suas.

A seguir na subseção 5.1.2 descreve-se como os agentes decidem sobre a troca de grupo.

### 5.1.2 Decisão sobre trocar de grupo

Os agentes do algoritmo *bee clustering* precisam decidir se trocam ou não para o grupo do agente o qual estão visitando. Esta tomada de decisão acontece sempre que a probabilidade de abandono *Pa* não for satisfeita para um par $(i,j)$ de agentes.

Para ajudar os agentes nesta tomada de decisão buscou-se inspiração em Agogino e Tumer (2006) onde os autores propõem um método para encontrar, de forma distribuída, o *ensemble* consenso relativo à um agrupamento de dados. Nesta abordagem, descrita anteriormente na Seção 4.3, os agentes precisam maximizar a utilidade global do agrupamento de maneira distribuída. No *bee clustering* os agentes tentam maximizar uma utilidade local relacionada ao grupo ao qual pertencem.

No algoritmo proposto os agentes decidem se trocam de grupo ou não de acordo com a qualidade do seu grupo atual. A qualidade do grupo de cada agente é avaliada através de uma utilidade $U$ que considera as características (atributos) de todos os participantes do grupo. Quanto mais similares forem os integrantes de um grupo maior será a utilidade daquele grupo. Desta forma, o valor da utilidade $U(k_i)$ do grupo $k_i$, onde o agente $i$ se encontra, deve ser maximizada e é calculado de acordo com a Equação 5.3.

$$U(k_i) = 1 - var(k_i) \tag{5.3}$$

Na Equação 5.3, $var(k_i)$ indica o valor da variância intra-*cluster* do grupo $k$ do agente $i$. Esta variância foi abordada anteriormente na Seção 2.3 como um dos métodos para se

avaliar a qualidade de um agrupamento. Porém, para o cálculo da utilidade $U$, utiliza-se somente a variância do grupo e não a variância de todo o agrupamento resultante. $var(k_i)$ precisa ser minimizada e pode ser calculada de acordo com a Equação 5.4, onde $n$ é o número de elementos dentro do grupo $k_i$, $d(i,c_i)$ é a distância Euclidiana entre o agente $i$ e o centróide $c_i$ do grupo $k_i$. O centróide $c_i$ é calculado de acordo com a Equação 5.5.

$$var(k_i) = \sqrt{\frac{1}{n}\sum_{i \in k} d^2(i,c_i)} \tag{5.4}$$

$$c_i = \frac{1}{n}\sum_{i \in C} i \tag{5.5}$$

Para verificar se abandona ou não o grupo $k_i$, o agente $i$ calcula a utilidade $U(k_i)$ e a utilidade $U-i$, a qual indica a utilidade de seu grupo $k_i$ sem a sua participação. Se $U(k_i)$ for maior que $U-i$, indicando que o a utilidade do grupo $k_i$ é melhor com a presença de $i$, o agente $i$ permanece no grupo $k_i$. Caso contrário, se $U(k_i)$ for menor que $U-i$, indicando que a utilidade do grupo de $i$ é melhor sem a sua participação, $i$ abandona $k_i$ e passa a integrar o grupo $k_j$ do agente $j$ que ele está visitando.

Como pode ser visto, $i$ não possui conhecimento algum sobre a qualidade do grupo do agente $j$ que está visitando. Desta forma, tal informação não é utilizada por $i$ no processo de tomada de decisão sobre a troca de grupo. Para esta decisão o agente $i$ utiliza somente informações a respeito do seu próprio grupo, pois parte do princípio que se na tomada de decisão sobre abandonar $j$, $Pa$ não foi satisfeita por se tratar de um agente provavelmente similar a ele, possivelmente $i$ também será similar aos agentes que pertencem ao grupo $k_j$ do agente $j$.

Em qualquer das duas situações anteriores, ou seja, trocando ou não de grupo, a próxima ação de $i$ será executar a dança do recrutamento com o objetivo de convidar outros agentes a fazerem parte de seu grupo, seja este um novo grupo ou não. Ao iniciarem a dança, os agentes precisarão decidir probabilisticamente se continuam ou não dançando para recrutar novos indivíduos para seus grupos. O mecanismo utilizado pelo algoritmo proposto para esta tomada de decisão será explicado na próxima subseção.

### 5.1.3 Decisão sobre recrutar agentes

No *bee clustering* os grupos são formados devido a um comportamento exibido pelos agentes denominado dança do recrutamento inspirado pelo modelo matemático descrito na Seção 3.4. Da mesma forma que na natureza as abelhas usam a dança para convidar outras companheiras para forragear em uma fonte de néctar de boa qualidade, no algoritmo proposto a dança também é utilizada como uma forma de convite, só que para recrutar outros indivíduos para fazerem parte de seus grupos. Assim, sempre que o estado do agente for $\delta = d$ isto indica que o agente está dançando para recrutar outros indivíduos para seu grupo. Neste caso ele precisa decidir se continua ou não neste estado.

O tempo que um agente permanece dançando para seu grupo $k$ é uma questão crucial no processo de agrupamento. Se os agentes permanecem dançando por um longo período, o algoritmo pode não funcionar corretamente porque todos os agentes tendem a dançar ao mesmo tempo, ou seja, todos tendem a possuir o mesmo estado $\delta = d$. Por outro lado, se os agentes permanecem dançando por um período muito curto, o algoritmo converge para um agrupamento com uma grande quantidade de pequenos grupos.

Desta forma, quando o estado de um agente é $\delta = d$, ele precisa decidir se continua ou não dançando para seu grupo. Para controlar o tempo que o agente permanece dançando

utilizou-se um mecanismo inspirado no modelo de divisão de trabalho exibido pelos insetos sociais, o qual foi descrito na Seção 3.2. Neste modelo, os agentes utilizam um estímulo associado a um limiar para, probabilisticamente, decidir se realizam uma tarefa ou não. No *bee clustering* os agentes utilizam este mesmo estímulo $S$ associado a um limiar $\theta$ para auxiliá-los na tomada de decisão sobre continuar ou não dançando para um determinado grupo. Os valores de $S$ e $\theta$ são utilizados no cálculo da probabilidade $Pd$ de continuar dançando de acordo com a Equação 5.6.

$$Pd = \frac{S_i^2}{S_i^2 + \theta_i^2} \tag{5.6}$$

A atualização da probabilidade de continuar dançando $Pd$ está diretamente relacionada com a qualidade do grupo do agente. Desta forma, quanto melhor a qualidade do grupo do agente $i$ maior será o valor de $Pd$, isto é, maior será a probabilidade de $i$ continuar dançando para recrutar novos agentes para seu grupo. A qualidade do grupo é medida através do cálculo da utilidade $U$ de cada grupo explicada anteriormente na subseção 5.1.1. Como os grupos são dinâmicos, ou seja, a cada ciclo podem receber novos agentes ou também perder alguns participantes, a probabilidade $Pd$ do agente $i$ é atualizada por ele a cada ciclo, enquanto seu estado for $\delta = d$, de acordo com as seguintes regras:

- Se a utilidade $U(k_i)$ do grupo $k$ do agente $i$ no tempo $t$ é maior que a utilidade $U(k_i)$ no tempo $t-1$, o estímulo de $i$ ($S_i$) é incrementado e o limiar de $i$ ($\theta_i$) é decrementado pela constante $\alpha$ aumentando assim a probabilidade $Pd$ do agente continuar dançando:

$$S_i = S_i + \alpha$$
$$\theta_i = \theta_i - \alpha$$

- Se a utilidade do grupo $U(k_i)$ no tempo $t$ é menor que a utilidade $U(k_i)$ no tempo $t-1$, $S_i$ é decrementado e $\theta_i$ é incrementado pela constante $\alpha$ diminuindo a probabilidade $Pd$ do agente continuar dançando:

$$S_i = S_i - \alpha$$
$$\theta_i = \theta_i + \alpha$$

Sempre que $Pd$ for satisfeita, o estado do agente permanecerá $\delta = d$ para que ele continue dançando e recrutando novos indivíduos para seu grupo. Quando $Pd$ não for satisfeita, indicando que a utilidade do grupo do agente está diminuindo, o agente encerra a dança de recrutamento mas permanece no mesmo grupo.

Enquanto $i$ está dançando existem aqueles agentes que estão assistindo sua dança com o objetivo de serem guiados para um grupo que contenha indivíduos mais similares a ele do que os integrantes de seu grupo atual. Tais agentes que estão assistindo a dança do recrutamento possuem estado $\delta = w$ e também precisam tomar uma decisão, a qual será explicada a seguir na próxima subseção.

### 5.1.4 Decisão sobre aceitar um convite

Outro momento de decisão acontece quando o agente $i$ está no estado $\delta = w$, o que significa que ele está assistindo à dança do recrutamento executada por outros agentes da colônia. Neste instante, $i$ sorteia aleatoriamente, com probabilidade uniformemente distribuída, um agente $j$ cujo estado seja $\delta = d$ e visita este agente $j$ com probabilidade $Pv$.

Para decidir probabilisticamente se visita ou não o agente $j$, $i$ calcula a probabilidade $Pv$ de acordo com a Equação 5.7, onde $D(k_j)$ é o número de agentes que está dançando para recrutar agentes para o grupo $k_j$ e $n$ é o número de agentes que pertencem ao grupo do agente $j$.

$$Pv = \frac{D(k_j)}{n} \tag{5.7}$$

Sendo assim, quanto maior o número de agentes dançando para um determinado grupo, maior será a chance de outros agentes testarem este grupo.

## 5.2 Algoritmo *bee clustering*

O algoritmo *bee clustering* é composto por uma colônia com $|\mathscr{A}|$ agentes. Cada agente possui um conjunto de atributos $\mathscr{X}$ e um conjunto de possíveis estados $\mathscr{S} = \{v, w, d\}$.

O Algoritmo 11 inicia com a criação da colônia e seus agentes e com a inicialização do parâmetro $\alpha$, o qual é utilizado para atualizar o estímulo $S$ e o limiar $\theta$ de cada agente no cálculo da probabilidade $Pd$.

Após a fase de inicialização, os agentes iniciam o processo de formação de grupos. As visitas entre os agentes são proporcionadas mediante convites efetuados por agentes cujo estado é $\delta = d$. Porém, no passo inicial (linha 5) quando nenhum agente está dançando para realizar os convites, é necessário que cada um sorteie aleatoriamente o primeiro agente a ser visitado. Todos os agentes possuem probabilidade $\frac{1}{|\mathscr{X}|}$ de serem sorteados. Este sorteio acontece somente no passo 0 do processo de agrupamento, nos demais as visitas acontecem porque alguns agentes efetuarão a dança do recrutamento. Neste instante 0 o estado de todos os agentes é $\delta = v$ (linha 7).

Supondo-se então, que no passo 0 o agente $i$ tenha sorteado $j$ para visitar. Suas próximas ações de acordo com o Algoritmo 11 estarão relacionadas ao seu estado atual. Se $i$ possuir estado $\delta = v$ significa que $i$ precisa decidir se abandona ou não o agente $j$, o qual está visitando. Para fazer isso, $i$ calcula a probabilidade $Pa$ de abandonar $j$ de acordo com a Equação 5.2. Como mencionado anteriormente na Seção 5.1 $Pa$ é inversamente proporcional à similaridade existente entre os agentes $i$ e $j$.

$i$ testa se abandona $j$ com probabilidade $Pa$. Se $i$ não abandonar $j$, então $i$ precisará decidir se troca ou não seu grupo $k_i$ pelo grupo $k_j$ do agente $j$. Esta decisão será tomada através do cálculo da utilidade $U$ de seu grupo $k_i$ de acordo com a Equação 5.3. Assim, $i$ calcula as utilidades $U(k_i)$ e $U - i$, as quais indicam as qualidades do grupo $k_i$ com e sem a sua participação respectivamente. Estes resultados indicarão ao agente $i$ qual opção é melhor para seu grupo, sair ou permanecer nele. Nestas duas situações, trocando ou não de grupo, o estado do agente $i$ mudará para $\delta = d$ indicando que a ação de $i$ no próximo passo do processo de agrupamento será dançar a fim de convidar outros agentes para participarem de seu grupo.

**Algorithm 11**: Algoritmo *Bee clustering*

```
1  initialize the colony with |𝒜| bees;
2  initialize parameter α;
3  repeat for each step
4      foreach i ∈ |𝒜| do
5          if step = 0 then
6              randomly choose an agent j;
7              δ_i ← v;
8          if δ_i = v then
9              calculate Pa (Equation 5.2);
10             if Pa then
11                 δ_i ← w;
12             else
13                 calculate utility U(k_i) and U − i (Equation 5.3);
14                 if U(k_i) < U(k − i) then
15                     k_j ⟵ i;
16                 δ_i ← d;
17         else if δ_i = d then
18             calculate Pd (Equation 5.6);
19             if Pd then
20                 keeping dancing;
21             else
22                 δ_i ← v;
23         else if δ_i = w then
24             repeat randomly choose a bee dancer j with state δ_j = d
25                 calculate Pv (Equation 5.7);
26             until Pv;
27             δ_i ← v;
28 until maxSteps;
```

Por outro lado, se o agente $i$ abandonar $j$, então, de acordo com o Algoritmo 11 (linha 11) o estado de $i$ passa a ser $\delta = w$ o que significa que no próximo passo do processo de agrupamento $i$ deverá assistir a dança do recrutamento realizada por todos aqueles agentes cujo estado é $\delta = d$ com o objetivo de escolher um novo agente $j$ pra visitar. É importante salientar que quando o agente $i$ abandona $j$, ele permanece em seu grupo $k_i$.

Seguindo o Algoritmo 11 (linha 17), se o estado do agente $i$ for $\delta = d$, significa que $i$ está dançando para recrutar outros agentes para fazerem parte de seu grupo $k_i$. Neste momento $i$ deverá calcular a probabilidade $Pd$ de continuar dançando para seu grupo de acordo com a Equação 5.6. Se o agente $i$ decidir parar de dançar, seu estado passa a ser $\delta = v$ indicando que no próximo passo do processo de agrupamento $i$ revisitará o agente $j$.

O fato do agente $i$ revisitar o agente $j$ é justificado pela dinamicidade dos grupos, ou seja, existe a possibilidade de que durante o período em que $i$ estava dançando, o agente $j$ possa ter trocado de grupo e esteja participando de um novo grupo o qual poderá conter

indivíduos similares ao agente $i$. Se ao contrário, o agente $i$ continuar dançando, continua o recrutamento de agentes para fazerem parte de seu grupo $k_i$.

Se o estado do agente $i$ for $\delta = w$ (linha 23), significa que $i$ está assistindo a dança do recrutamento, ou seja, está observando aqueles agentes cujo estado é $\delta = d$. Assim, ele aleatoriamente sorteará um novo agente $j$ para visitar. Todos os agentes com estado $\delta = d$ têm a mesma probabilidade de serem sorteados por $i$. Após sortear um agente $j$, $i$ deverá decidir se aceita ou não o convite deste agente. Para tomar esta decisão, $i$ calcula uma probabilidade $Pv$ de acordo com a Equação 5.7 que considera a proporção de agentes que estão dançando para o grupo de $j$ em relação a todos os agentes que estão participando daquele grupo. Se $i$ visitar $j$, o estado do agente $i$ passará a ser $\delta = v$ indicando que no próximo passo o processo de agrupamento reinicia e a ação de $i$ será a de visitar o agente $j$ sorteado e verificar se o abandona ou não. Se ao contrário, o agente $i$ não visitar $j$, $i$ continuará sorteando outros agentes para visitar.

O algoritmo *bee clustering* executará até que o número máximo de iterações ($maxSteps$) seja alcançado.

# 6 VALIDAÇÃO DA ABORDAGEM PROPOSTA

Para testar a eficácia do algoritmo proposto em gerar agrupamentos de agentes que representam um conjunto de dados utilizou-se dois tipos de aplicação. Na primeira aplicação, realizou-se experimentos em bases de dados reais de domínio público e uma base de dados artificiais, as quais se tem conhecimento de sua correta classificação. Tal aplicação será tratada na Seção 6.1. Como segunda aplicação utilizou-se o ambiente de simulação da *Robocup Rescue* que retrata um problema de alocação de tarefas de resgate à grupos de agentes que devem operar de forma colaborativa. Este cenário foi utilizado com a intenção de se validar a aplicação do *bee clustering* na formação destes grupos de agentes heterogêneos. A segunda aplicação será tratada em detalhes na Seção 6.2.

## 6.1 Aplicação I

Com o intuito de se verificar o desempenho do algoritmo proposto em realizar o agrupamento de um conjunto de dados, utilizou-se algumas bases de dados de domínio público: *Iris*, *yeast*, *Wine* e *glass*, as quais podem ser encontradas no *UCI Machine Learning Repository* (ASUNCION; NEWMAN, 2007). Utilizou-se ainda duas outras bases de dados: a *leukemia*, conhecida na literatura como *St Jude leukemia* (YEOH et al., 2002), a qual contém dados reais que representam problemas de bioinformática; e também o conjunto de dados ds2c2sc13 gerado artificialmente. A seguir descreve-se as características principais destas bases:

- *Iris*: base de dados que contém 3 classes as quais representam 3 tipos de flores da família das Iridáceas: Iris-Virginica, Iris-Versicolor e Iris-Setosa. Cada dado possui 4 atributos: comprimento da sépala, largura da sépala, comprimento da pétala e largura da pétala. A base de dados contém 150 amostras, sendo 50 de cada uma das classes;

- *Yeast*: contém 1484 elementos, cada um com 8 atributos. A base de dados contém 10 classes com os seguintes tamanhos: 463, 429, 244, 163, 51, 44, 35, 30, 200, e 5;

- *Wine*: contém os resultados de uma análise química dos vinhos produzidos em uma região da Itália e derivados de três produtores diferentes. A base contém 178 dados, cada um com 13 atributos. Os dados estão divididos em 03 classes as quais representam 3 tipos de vinhos e possuem os seguintes tamanhos: 59, 71 e 48;

- *Glass*: contém 214 dados relacionados a classificação de vidros. Os dados apresentam 3 estruturas diferentes, as quais são divididas em 2, 5 ou 6 classes. Cada dado é composto por 10 atributos.

- *Leukemia*: contém dados relacionados a expressão gênica relativa a um subtipo de leucemia infantil. Este conjunto de dados é composto por 271 elementos que podem ser divididos em duas estruturas diferentes: uma com 03 classes e outra com 07 classes. Cada elemento do conjunto possui 327 atributos.

- ds2c2sc13: contém dados artificiais gerados para conter 3 estruturas diferentes. Estas estruturas são heterogêneas podendo conter 2, 5 ou 13 classes com diferentes níveis de refinamento. Cada dado é composto por 2 atributos.

A Tabela 6.1 resume as principais características das bases de dados utilizadas para a validação do algoritmo proposto. Como pode ser visto na descrição anterior, algumas das bases utilizadas apresentam estruturas diferentes. É o caso da *leukemia*, *glass* e ds2c2sc13. No presente trabalho as classes nas quais os conjuntos de dados são divididos originalmente são chamadas de grupos, sendo assim o número de grupos é representado por $K$. O número de dados da base é igual ao número de agentes no algoritmo *bee clustering*, portanto é representado por $|\mathscr{A}|$. $|\mathscr{X}|$ indica a dimensionalidade dos dados da base, ou seja, o número de atributos que cada dado possui, no caso, corresponde ao mesmo número de atributos dos agentes.

Tabela 6.1: Resumo das bases de dados utilizadas.

| **Base de dados** | $K$ | $\lvert\mathscr{A}\rvert$ | $\lvert\mathscr{X}\rvert$ |
|---|---|---|---|
| *Iris* | 3 | 150 | 4 |
| *Yeast* | 10 | 1484 | 8 |
| *Wine* | 3 | 178 | 13 |
| *Leukemia* | 3 ou 7 | 271 | 327 |
| *Glass* | 2, 5 ou 6 | 214 | 10 |
| ds2c2sc13 | 2, 5 ou 13 | 588 | 2 |

Os experimentos realizados foram restringidos a estas bases de dados para possibilitar a comparação com resultados reportados na literatura e por se tratarem de bases úteis, das quais se conhece suas corretas classificações, o que facilita a avaliação do agrupamento resultante.

Os resultados utilizados para comparar o algoritmo proposto são provenientes dos seguintes algoritmos: *K-means*, *Average link* e MOCLE (apresentados na Seção 4.3), *Ant-based clustering* (Seção 3.1) e I-PACE (Seção 4.2).

### 6.1.1 Experimentos e resultados

Para se verificar a qualidade do algoritmo proposto realizou-se alguns experimentos com as bases de dados descritas anteriormente. A avaliação do agrupamento obtido com a utilização do *bee clustering* foi realizada através do cálculo das medidas *F-measure* e índice *Rand* descritas na Seção 2.3.

Os valores utilizados para os parâmetros foram: $\alpha = 0.02$, $maxSteps = |\mathscr{A}| * \mu$ e $\mu = 6$. Escolheu-se estes valores após a realização de diversos testes com diferentes valores para cada parâmetro. O número de agentes $|\mathscr{A}|$ utilizados e a quantidade de atributos $|\mathscr{X}|$ de cada agente, respectivamente recebem os valores dos tamanhos das bases de dados e o número de atributos de cada dado desta base. Desta forma, $|\mathscr{A}|$ e $|\mathscr{X}|$ possuem valores diferentes para cada base a ser utilizada.

A tabela 6.2 apresenta as médias dos resultados obtidos com o cálculo de *F-measure* e índice *Rand* provenientes de 60 repetições para o agrupamento produzido pelo algoritmo *bee clustering* em comparação com os algoritmos: *K-means*, *Average link*, *Ant-based clustering* e I-PACE. O valor do desvio padrão é mostrado entre parênteses. Além dos valores de *F-measure* e do índice *Rand* também são mostrados o número de grupos gerados pelos algoritmos.

Tabela 6.2: Média do *F-measure* e índice *Rand* e desvio padrão sobre 60 repetições para os algoritmos *K-means*, *Ant-based clustering*, *Average link*, I-PACE e *Bee clustering*.

| **Iris** | ***K-means*** | ***Ant-based Clustering*** | ***Average Link*** | **I-PACE** | ***Bee Clustering*** |
|---|---|---|---|---|---|
| *F-measure* | 0.8245 (0.0848) | 0.8168 (0.0148) | 0.8098 (0.0) | **0.8324** (0.0001) | 0.8205 (0.0380) |
| Índice *Rand* | 0.8165 (0.1012) | 0.8254 (0.0080) | 0.8223 (0.0) | **0.8300** (0.0008) | 0.8153 (0.0184) |
| Número de grupos identificados | 3 | 3.02 | 3 | 3 | 3.04 |
| ***Yeast*** | ***K-means*** | ***Ant-based Clustering*** | ***Average Link*** | **I-PACE** | ***Bee Clustering*** |
| *F-measure* | 0.4315 (0.0044) | 0.4353 (0.0345) | 0.4483 (0.0) | 0.4348 (0.0899) | **0.4893** (0.0261) |
| Índice *Rand* | 0.7506 (0.0012) | 0.6781 (0.0752) | 0.7426 (0.0) | 0.7992 (0.0009) | **0.8172** (0.0579) |
| Número de grupos identificados | 10 | 5.36 | 10 | 10 | 10.3 |
| ***Wine*** | ***K-means*** | ***Ant-based Clustering*** | ***Average Link*** | **I-PACE** | ***Bee Clustering*** |
| *F-measure* | **0.9312** (0.0615) | 0.8760 (0.0208) | 0.9255 (0.0) | 0.8991 (0.0002) | 0.8048 (0.0345) |
| Índice *Rand* | **0.9167** (0.0653) | 0.8554 (0.0191) | 0.9044 (0.0) | 0.8290 (0.0008) | 0.8153 (0.0597) |
| Número de grupos identificados | 3 | 3 | 3 | 3 | 3.1 |

Como pode ser visto na tabela 6.2, para a base de dados *Iris* e *Wine* o algoritmo proposto não foi capaz de superar os resultados obtidos pelo algoritmo I-PACE e *K-means* respectivamente, embora para o conjunto *Iris* tenha obtido resultados bem próximos. No entanto, é importante ressaltar que o algoritmo *bee clustering* dispensa a necessidade de informações *a priori*, como é o caso do *K-means*, *Average link* e I-PACE. O algoritmo I-PACE necessita a inicialização de palavras chaves relacionadas a base de dados que será agrupada, enquanto que o *K-means* e *Average link* necessitam receber como parâmetro o número de grupos em que a base de dados será dividida. Além disso, o *bee clustering* realiza o agrupamento de forma distribuída possibilitando sua aplicação em cenários onde esta característica seja desejada, como por exemplo situações onde os dados não estejam situados em uma localização central e não possam ser concentrados em um único local por motivos de segurança, privacidade, entre outros.

Na base de dados *yeast*, o algoritmo proposto superou os resultados obtidos pelos demais algoritmos tanto pelo *F-measure* quanto pelo índice *Rand* e foi capaz de, na maioria das simulações, encontrar automaticamente o correto número de grupos.

A tabela 6.3 apresenta a média e, entre parênteses, o desvio padrão obtido com o índice *Rand* para o agrupamento produzido pelo algoritmo *bee clustering* em comparação com os algoritmos: *K-means*, *Ant-based clustering* e MOCLE. As bases de dados utilizadas para este experimento foram: *leukemia*, ds2c2sc13 e *glass*. A base de dados *leukemia* apresenta 2 estruturas diferentes, enquanto que o conjunto ds2c2sc13 e *glass* apresentam 3 estruturas. Algoritmos baseados em agrupamento multi-objetivo tendem a encontrar mais do que uma estrutura em um conjunto de dados obtendo bons resultados para cada uma delas. É o caso do algoritmo MOCLE, cujos resultados foram utilizados para a comparação com aqueles obtidos pelo algoritmo proposto. Já os algoritmos de agrupamento que não possuem a característica de otimização de múltiplos objetivos (*K-means*, *Ant-based clustering*), não são capazes de encontrar estruturas diversas em uma única base de dados sem que se modifique os valores de seus parâmetros.

Embora o algoritmo *bee clustering* não tenha o propósito de realizar um agrupamento otimizando múltiplos objetivos, optou-se por conduzir os experimentos das bases *leukemia*, ds2c2sc13 e *glass* sem alterações nos valores dos seus parâmetros. Sendo assim, executou-se 30 repetições para cada base de dados e a partir do grupamento resultante do algoritmo proposto, calculou-se os valores do índice *Rand* para cada uma das estruturas, os quais encontram-se reportados na tabela 6.3.

Como pode ser visto na Tabela 6.3, os resultados do algoritmo proposto destacaram-se dos demais para uma determinada estrutura de cada conjunto. No caso da *leukemia*, a estrutura reconhecida pelo *bee clustering* foi aquela com 3 classes, pois na média das repetições ele convergiu para um agrupamento com 3.1 grupos. Além disso para esta estrutura específica o *Bee clustering* foi capaz de realizar um agrupamento melhor do que os demais algoritmos de acordo com o índice *Rand*, métrica utilizada para validação do resultado.

O mesmo acontece com a base ds2c2sc13, onde o algoritmo proposto identificou a estrutura com 13 classes, convergindo na média das repetições para um agrupamento com 14.3 grupos. Para esta estrutura o *bee clustering* obteve melhores resultados para o índice *Rand* do que os reportados pelos demais algoritmos.

Os resultados obtidos para o conjunto de dados *glass* apontaram bons resultados gerados pelo algoritmo *bee clustering* para duas de suas estruturas (com 5 e 6 grupos). Para a estrutura de 6 grupos o algoritmo proposto destacou-se dos demais tanto pelo índice *Rand* quanto pelo número de grupos encontrados automaticamente, inclusive obtendo va-

Tabela 6.3: Média do Índice *Rand* resultante de 30 repetições para os algoritmos *K-means*, *Average link*, MOCLE e *Bee clustering*.

| **Leukemia - 3 Classes** | ***K-means*** | ***Average Link*** | **MOCLE** | ***Bee Clustering*** |
|---|---|---|---|---|
| Índice *Rand* | 0.31 (0.31) | 0.32 (0) | 0.31 (0.02) | **0.49** (0.02) |
| Número de grupos identificados | 5.6 (2) | 6 (0) | 7.3 (0.5) | **3.1** (0.3) |
| **Leukemia - 7 Classes** | ***K-means*** | ***Average Link*** | **MOCLE** | ***Bee Clustering*** |
| Índice *Rand* | 0.75 (0.02) | 0.64 (0) | **0.77** (0) | 0.51 (0.09) |
| Número de grupos identificados | 8.4 (1.9) | 16 (0) | **8** (0) | 3.1 (0.3) |
| **ds2c2sc13 - 2 Classes** | ***K-means*** | ***Average Link*** | **MOCLE** | ***Bee Clustering*** |
| Índice *Rand* | **1** (0) | **1** (0) | **1** (0) | 0.52 (0.03) |
| Número de grupos identificados | **2** (0) | **2** (0) | **2** (0) | 14.3 (1.7) |
| **ds2c2sc13 - 5 Classes** | ***K-means*** | ***Average Link*** | **MOCLE** | ***Bee Clustering*** |
| Índice *Rand* | 0.79 (0.04) | **1** (0) | **1** (0) | 0.55 (0.08) |
| Número de grupos identificados | 4.7 (0.9) | **5** (0) | **5** (0) | 14.3 (1.7) |
| **ds2c2sc13 - 13 Classes** | ***K-means*** | ***Average Link*** | **MOCLE** | ***Bee Clustering*** |
| Índice *Rand* | 0.75 (0.02) | 0.64 (0) | 0.77 (0) | **0.80** (0.02) |
| Número de grupos identificados | 8.4 (1.9) | 16 (0) | 8 (0) | **14.3** (1.7) |
| ***Glass* - 2 Classes** | ***K-means*** | ***Average Link*** | **MOCLE** | ***Bee Clustering*** |
| Índice *Rand* | 0.63 (0.02) | 0.67 (0) | **0.75** (0.02) | 0.57 (0.06) |
| Número de grupos identificados | 3 (0.80) | 9 (0) | **2** (0.70) | 5.4 (0.52) |
| ***Glass* - 5 Classes** | ***K-means*** | ***Average Link*** | **MOCLE** | ***Bee Clustering*** |
| Índice *Rand* | 0.47 (0.04) | **0.56** (0) | **0.56** (0.06) | 0.51 (0.02) |
| Número de grupos identificados | 3 (0.80) | 12 (0) | 10 (3.33) | **5.4** (0.52) |
| ***Glass* - 6 Classes** | ***K-means*** | ***Average Link*** | **MOCLE** | ***Bee Clustering*** |
| Índice *Rand* | 0.23 (0.15) | 0.26 (0) | 0.27 (0.01) | **0.56** (0.07) |
| Número de grupos identificados | 5 (2.80) | 12 (0) | 4 (2.80) | **5.4** (0.52) |

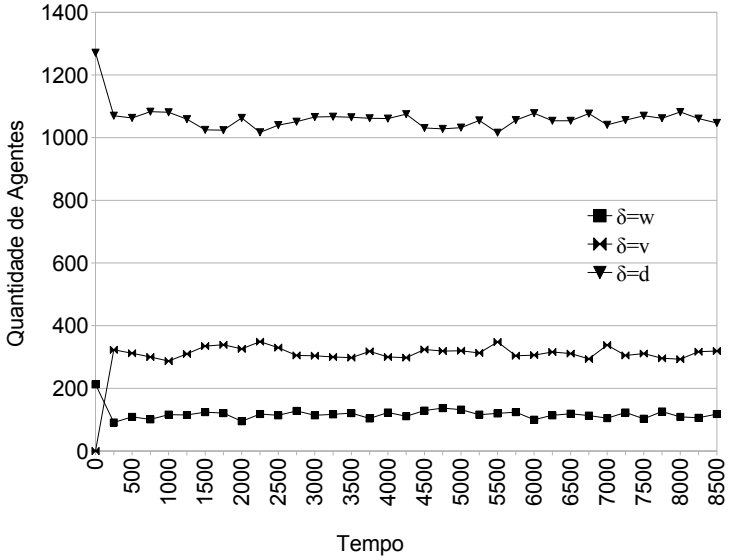


Figura 6.1: Quantidade de agentes em cada estado $\mathscr{S} = \{w, v, d\}$ durante uma simulação do algoritmo *bee clustering*.

lores bem acima daqueles apresentados pelos demais algoritmos. Já para a estrutura de 5 grupos o *bee clustering* obteve resultados próximos aos demais pois trata-se de uma configuração bastante similar aquela que ele foi capaz de reconhecer (6 grupos).

Como mencionado na Seção 4.3.2, estas comparações realizadas entre o *bee clustering* e o algoritmo MOCLE especificamente foram conduzidas com o intuito de se avaliar a possibilidade de uma futura aplicação do algoritmo proposto para a descoberta de estruturas alternativas em bases de dados de bioinformática. De acordo com os resultados obtidos com o *bee clustering* sobre as bases *leukemia*, ds2c2sc13 e *glass* percebe-se uma possível viabilidade de tal aplicação já que o algoritmo proposto foi capaz de reconhecer justamente aquelas estruturas menos óbvias, obtendo em alguns casos, resultados bem superiores aos demais algoritmos.

### 6.1.2 Comportamento dos agentes

Com o objetivo de ilustrar o comportamento dos agentes do algoritmo *bee clustering* gerou-se alguns gráficos a partir das simulações executadas para validar a abordagem. Os gráficos gerados foram obtidos durante as simulações da base de dados *yeast*.

A figura 6.1 mostra o gráfico com o número de agentes em cada estado durante a execução de uma simulação. Como pode ser visto, o estado $\delta = d$ é aquele que concentra maior quantidade de agentes durante toda a simulação. Isto acontece porque os agentes estão participando de grupos de boa qualidade o que os faz dançar para recrutar outros agentes para fazerem parte de seus grupos. Nos primeiros passos da simulação, a quantidade de agentes no estado $\delta = w$ supera a quantidade de agentes no estado $\delta = v$, pois a quantidade de visitas que resultam em abandono é grande, o que faz com que os agentes tenham que assistir a dança do recrutamento para escolher novos indivíduos para visitar.

A figura 6.2 mostra o gráfico com a dinâmica de um grupo, aleatoriamente escolhido, durante um intervalo de tempo de uma simulação do algoritmo *bee clustering*. Como pode ser visto o grupo surgiu logo nos primeiros instantes da simulação com 2 ou 3 agentes, em seguida foi recebendo uma quantidade maior de participantes até atingir a quantidade

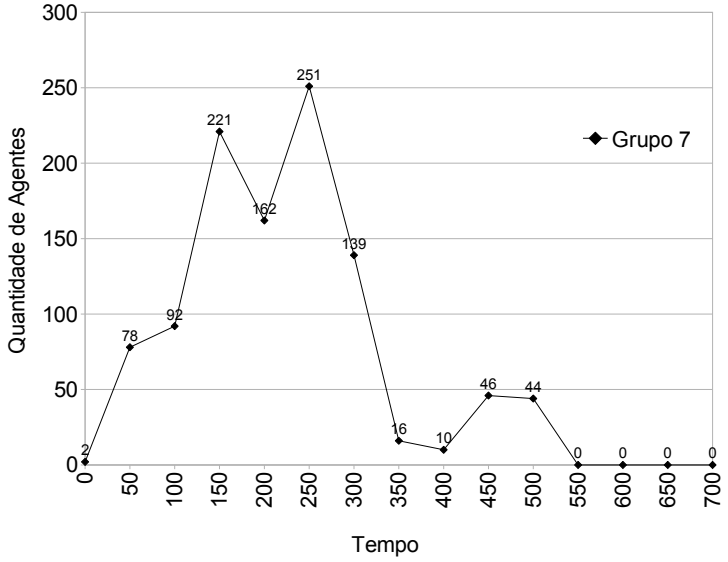


Figura 6.2: Número de agentes no grupo "7" durante uma simulação do algoritmo *bee clustering*.

máxima de 251 agentes no instante 250, quando começou a perder participantes até deixar de existir no tempo 550. Este gráfico ilustra bem o processo de formação de um grupo. Seu início com um pequena quantidade de agentes que após dançarem conseguem recrutar novos integrantes para o grupo mas começa a perder participantes a medida que sua qualidade vai decaindo. O grupo utilizado como amostra na figura 6.2 reflete a dinâmica da maioria dos grupos gerados durante a simulação do algoritmo proposto, isto significa que somente uma pequena quantidade de grupos mantêm-se representativos durante a simulação. Em média esta quantidade é igual a quantidade de classes originais dos dados.

A figura 6.3, por outro lado, mostra a dinâmica de um grupo que se fixou durante uma simulação do *bee clustering*. Como pode ser visto o grupo iniciou com uma pequena quantidade de agentes e no decorrer da simulação o grupo foi recebendo e perdendo participantes até que por volta do instante 6750 o grupo manteve-se com uma quantidade entre 200 e 240 agentes até o final da simulação.

No decorrer de uma simulação os agentes do *bee clustering* trocam de grupos de acordo com a qualidade de seu grupo. Esta dinâmica faz com que eles participem ou não de vários grupos até se fixarem naquele que possui integrantes tão similares que a saída de qualquer um resultaria na perda de utilidade do grupo. Normalmente o agente que possui características bastante similares a um único grupo tende a se alocar no grupo correto nos primeiros instantes da simulação. Já aquele agente que é muito similar a dois ou mais grupos tem a tendência de alternar sua participação entre estes grupos durante um certo período podendo se fixar ou não em um único grupo no final da simulação.

### 6.1.3 Conclusão

A aplicação I se refere a experimentos realizados para a validação do algoritmo *bee clustering* sobre bases de dados reais públicas e uma base de dados artificiais. Para a descrição dos experimentos realizados apresentou-se primeiramente as bases de dados utilizadas bem como suas principais características relacionadas a quantidade de dados, quantidade de atributos, número de classes e possíveis estruturas. A seguir fez-se refe-

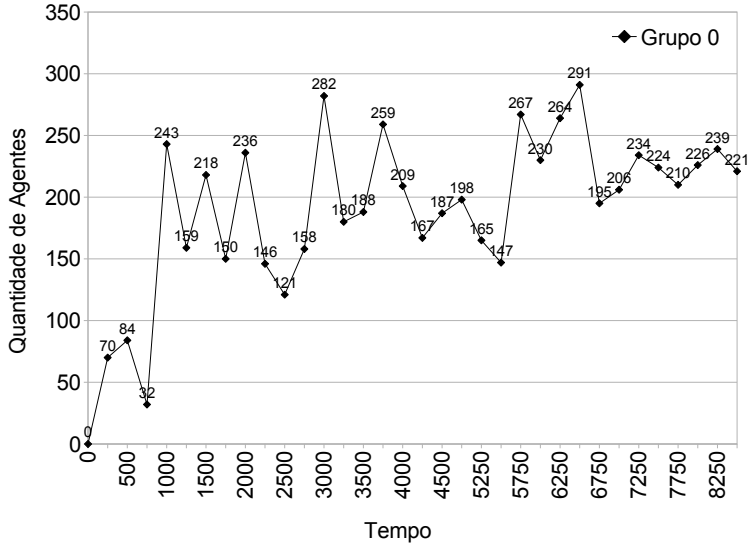


Figura 6.3: Número de agentes no grupo "0" durante uma simulação do algoritmo *bee clustering*.

rência aos resultados obtidos com o algoritmo proposto juntamente com os resultados dos algoritmos utilizados no processo de comparação. Finalmente mostrou-se graficamente o comportamento dos agentes durante uma simulação onde o *bee clustering* realiza o agrupamento dos agentes que representavam a base de dados *yeast*.

Embora o algoritmo proposto não tenha superado todos os resultados dos algoritmos com os quais foi comparado, ele apresentou resultados promissores se destacando em algumas bases de dados como foi o caso dos conjuntos *glass* e *yeast*. Para a base de dados *glass* (estrutura com 6 grupos) o *bee clustering* chegou a obter um resultado em torno de 200% melhor que o resultado obtido pelo algoritmo MOCLE.

Um ponto importante a se considerar é o fato de que a abordagem proposta realiza o agrupamento de maneira distribuída sem otimizar um objetivo global como ocorre com vários dos algoritmos com os quais foi comparado. Ao contrário, os agentes do *bee clustering* tentam maximizar somente uma utilidade local relacionada a qualidade de seu próprio grupo e mesmo assim foi capaz de atingir resultados comparáveis as abordagens centralizadas. No pior caso, o resultado do algoritmo *bee clustering* foi aproximadamente 9% inferior ao resultado do algoritmo distribuído I-PACE. Além do mais, o algoritmo proposto dispensa a necessidade de se definir *a priori* parâmetros do tipo: número de grupos, tamanho do grupo, densidade mínima ou máxima de um grupo, entre outros. O uso destes parâmetros facilita a realização do agrupamento pois eles podem ser considerados como "dicas" sobre o resultado desejado.

## 6.2 Aplicação II

Outro cenário utilizado para se verificar a eficácia do algoritmo *bee clustering* foi o ambiente de simulação *Robocup Rescue*. Tal aplicação foi realizada para se investigar a qualidade do algoritmo proposto na formação de grupos de agentes com características ou habilidades similares para a alocação de tarefas entre eles.

Para evitar confusão na explicação da aplicação que se segue, a palavra "agente" será

utilizada para se referir aos indivíduos do ambiente de simulação *RoboCup Rescue*, sendo o termo "abelha" reservado para os agentes do algoritmo *bee clustering*. A utilização desta nomenclatura se restringirá somente a esta seção.

O ambiente de simulação da *RoboCup Rescue* é um ambiente dinâmico que simula operações de busca e resgate em uma área urbana após o acontecimento de uma catástrofe, como por exemplo um terremoto. O ambiente muda continuamente e significantemente com desabamentos de prédios, propagação de incêndios e bloqueios de estradas com escombros. Agentes de busca e resgate precisam navegar neste cenário tentando salvar vidas e limitando a propagação dos incêndios e danos as construções.

O projeto de simulação possibilita aos pesquisadores a experimentação de diferentes estratégias a serem utilizadas em situações de desastre (KITANO et al., 1999).

A seguir descreve-se as características do cenário utilizado para um correto entendimento desta aplicação do algoritmo *bee clustering* na formação de grupos de agentes.

### 6.2.1 Descrição do cenário

O simulador da *RoboCup Rescue* tenta reproduzir condições que surgem após o acontecimento de um terremoto em uma área urbana. Como entrada, o simulador recebe dados geográficos (mapas com ruas, prédios, casas, entre outros) além de informações sobre o desastre. Com estas informações o simulador modela o desabamento de prédios, bloqueio de ruas, incêndios, civis queimados e/ou feridos. O simulador modela ainda alguns agentes como policiais, bombeiros e ambulâncias que atuam para tentar amenizar a situação. Estes agentes têm habilidades específicas: ambulâncias são aptas a recuperar civis que estão queimados e transferi-los para um refúgio onde possam ser atendidos; os bombeiros estão aptos a extinguir os incêndios e os policiais podem desbloquear as ruas removendo os escombros.

Para os experimentos realizados neste trabalho utilizou-se o mapa *Kobe*, o qual é amplamente usado pela comunidade que trabalha com o simulador. A Figura 6.4 mostra este mapa em sua configuração inicial. O simulador posiciona, aleatoriamente, os agentes no mapa. Cada agente move-se de maneira aleatória para uma direção e pode perceber tarefas que precisam ser realizadas. Na Figura 6.4 os círculos vermelhos, azuis e brancos representam os agentes do cenário: bombeiros, policiais e ambulâncias respectivamente, os círculos verdes, os retângulos amarelos e as cruzes representam as tarefas: civis a serem resgatados, pontos de incêndios e bloqueios respectivamente.

A partir deste conjunto de agentes e de um conjunto de tarefas que precisam ser realizadas por eles, considera-se situações nas quais estas tarefas devem ser alocadas apropriadamente entre os grupos de agentes. Esta alocação deve ser realizada considerando-se as capacidades dos agentes com relação às características das tarefas.

Algumas tarefas precisam ser realizadas por mais de um agente de acordo com sua gravidade, o que justifica a abordagem de formação de grupos. A tarefa de resgate de civis pode diminuir com o passar do tempo pois alguns civis podem morrer, enquanto que a tarefa de controle dos pontos de incêndio tende a aumentar com a propagação do fogo para outras construções.

Para que se atinja o objetivo de formar grupos de agentes com características similares para realizarem determinada tarefa de maneira satisfatória, é necessário que se defina claramente quais são estas características que serão envolvidas para a verificação da similaridade durante o processo de agrupamento. Esta atividade contempla a fase de extração das características a qual pertence aos passos para agrupamento de dados descritos no Capítulo 2. A Tabela 6.4 apresenta os atributos utilizados relacionados aos agentes e as

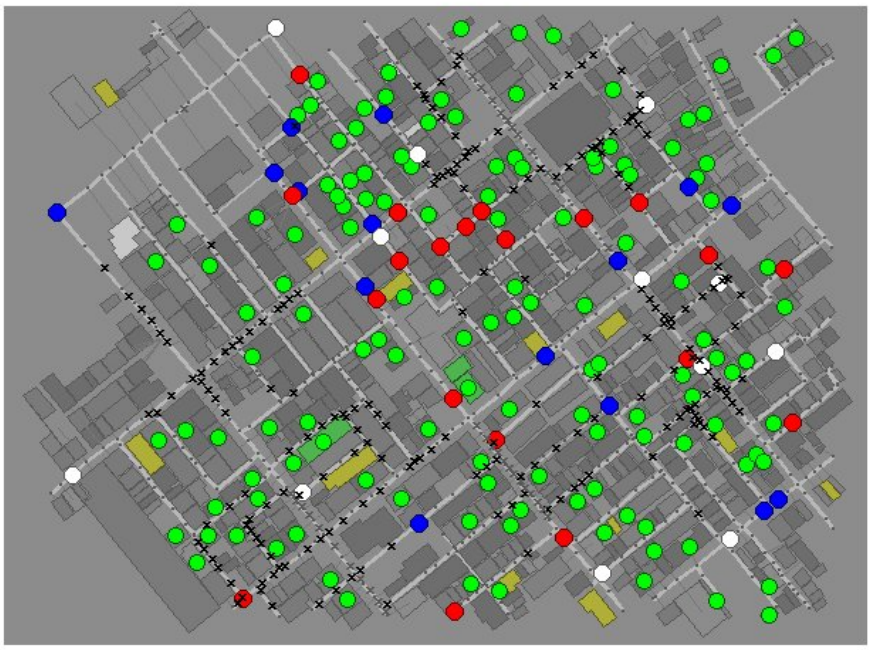

Figura 6.4: Mapa *Kobe* usado nos experimentos

tarefas. O atributo “posição” indica a posição atual do agente e da tarefa, respectivamente. O atributo “tipo” do agente indica se ele é um bombeiro, um policial ou uma ambulância. Já no caso da tarefa, este atributo indica se esta é um bloqueio, um civil a ser resgatado ou um incêndio. O atributo “gravidade” indica o nível de seriedade da tarefa, enquanto que o atributo “esforço requerido” aponta a quantidade de agentes necessária para que a tarefa seja concluída.

Neste método de validação do algoritmo *bee clustering* preocupa-se primeiramente com a alocação das tarefas do cenário. Optou-se por evitar o uso de métodos *ad hoc* e/ou heurísticas para comportamentos específicos ou comunicação desnecessária entre os agentes. Por exemplo, para se executar as simulações, desligou-se todos os componentes das centrais do simulador, os quais centralizam informações que têm por objetivo facilitar a coordenação dos agentes. Tais componentes são a central dos bombeiros, dos policiais e das ambulâncias, os quais estão encarregados de facilitar a comunicação e coordenação dos times de bombeiros, policiais e ambulâncias respectivamente.

### 6.2.2 Formação de grupos via *bee clustering*

A presente subseção aborda questões relacionadas ao algoritmo *bee clustering* para uma correta validação de sua aplicação no cenário da *RoboCup Rescue*. Após uma análise do cenário proposto nesta aplicação foram necessárias algumas adaptações nas Equações do algoritmo *bee clustering* para que este pudesse formar os grupos de maneira satisfatória.

Observações realizadas durante as execuções do simulador *RoboCup Rescue* indicam que os esforços demandados pelos bombeiros e ambulâncias para realizarem suas tarefas são mais efetivos se direcionados àquelas tarefas cuja gravidade é menos intensa. Já no

Tabela 6.4: Atributos utilizados no processo de agrupamento dos agentes no cenário da *RoboCup Rescue*

| **Atributos dos Agentes** | **Descrição** | **Atributos das Tarefas** | **Descrição** |
|---|---|---|---|
| Posição | indica a posição do agente no mapa | Posição | indica a posição da tarefa no mapa |
| Tipo | bombeiro, ambulância ou policial | Tipo | incêndio, resgate ou bloqueio |
| | | Gravidade | representa o nível de gravidade da tarefa |
| | | Esforço requerido | quantidade de agentes necessária para realizar a tarefa |

caso dos policiais, seu esforços demandados para desbloquear uma rua são mais efetivos se direcionados àqueles bloqueios mais graves, pois estes prejudicam a navegação dos demais agentes no cenário. Desta forma a Equação 5.2 que calcula a probabilidade de abandono *Pa*, foi adaptada para $Pa'$ com o intuito de refletir as necessidades do cenário e dos diferentes tipos de tarefas. Sendo assim, $Pa'$ indica a probabilidade da abelha $i$ abandonar a tarefa percebida pela abelha $j$ que está sendo visitada. $Pa'$ é calculada de acordo com a Equação 6.1.

$$Pa' = (dp(i,T_j) * \gamma) + (\beta * (1-\gamma)), \tag{6.1}$$

$$\text{onde } \beta = \begin{cases} 1-g, & \text{se } T_j \text{ é um bloqueio} \\ f, & \text{se } T_j \text{ é um incêndio} \\ h, & \text{se } T_j \text{ é um civil} \end{cases}$$

onde $\beta$ representa a gravidade da tarefa $T_j$, $dp(i,T_j)$ é a distância Euclidiana ponderada entre a abelha $i$ e a tarefa $T_j$ percebida pela abelha $j$ e pode ser calculada de acordo com a Equação 6.2. A distânica $dp(i,T_j)$ e a gravidade $\beta$ são ponderados no cálculo de $Pa'$ por $\gamma$ e $1-\gamma$. $\gamma$ poderá receber valores no intervalo [0,1]. Desta forma, quanto mais próximo de 1 for o valor de $\gamma$, maior será a influência de $dp(i,T_j)$ sobre a probabilidade $Pa'$. Caso contrário, a gravidade $\beta$ é que exercerá maior influência. Se o valor de $\gamma$ for 0.5, tanto a distância como a gravidade exercerão o mesmo peso sobre $Pa'$.

$$dp(i,T_j) = \sqrt{\sum_{x=1}^{|\mathscr{X}|} W_x(i_x - T_{jx})} \tag{6.2}$$

Para o cálculo de $dp(i,T_j)$ considerou-se os atributos "posição" e "tipo", os quais foram normalizados, juntamente com $\beta$, para o intervalo [0,1] para garantir que $Pa'$ também respeite este intervalo. $W$ é um peso utilizado para ponderar a importância de cada um destes atributos.

A introdução da distância Euclidiana ponderada para o cálculo de $Pa'$ é utilizada no lugar da distância Euclidiana simples para que o atributo "posição" não exerça domínio

sobre o atributo "tipo". Pois, o atributo "tipo" é que deve possuir um maior grau de importância para diminuir a probabilidade de uma tarefa ser atribuída a um abelha que não possa realizá-la.

A Equação 5.3 que calcula a utilidade $U(k_i)$ do grupo $k_i$ da abelha $i$, também foi adaptada para o cenário do *RoboCup Rescue*. Assim $U(k_i)'$ deve ser calculada pela Equação 6.3, onde $u$ e $r$ indicam o esforço requerido pela tarefa $T_{k_i}$ do grupo $k$ da abelha $i$, $n$ é número de abelhas que pertencem ao grupo $k_i$ e $var(k_i)$ é a variânica intra-*cluster* de $k_i$.

$$U(k_i)' = \begin{cases} \frac{n}{u}, & \text{se } T_{k_i} \text{ é um bloqueio} \\ var(k_i), & \text{se } T_{k_i} \text{ é um incêndio} \\ \frac{n}{r}, & \text{se } T_{k_i} \text{ é um civil} \end{cases} \tag{6.3}$$

Sendo assim, para uma abelha $i$ decidir se sai do grupo $k_i$, que realizará a tarefa $T_{k_i}$, para participar do grupo da abelha $j$, ela calcula a utilidade de seu grupo de acordo com o tipo de tarefa para o qual seu grupo está sendo formado. Se o grupo de $i$ estiver sendo formado para realizar uma tarefa do tipo bloqueio ou resgate de civil, a utilidade $U(k_i)$ considera o atributo "esforço requerido"

A mudança que ocorre no múmero de tarefas no cenário da *RoboCup Rescue* representa um desafio para a formação dos grupos de agentes, pois estes precisam ser reagrupados de tempos em tempos ou mediante o acontecimento de algum evento. No presente trabalho experimentou-se o reagrupamento dos agentes em ambos os casos: orientado pelo tempo e também pelo acontecimento de um evento, onde este evento é a quantidade de agentes desagrupados. A seguir explica-se como foram conduzidos os respectivos reagrupamentos relacionados a estes dois casos.

#### *6.2.2.1 Agrupamento orientado pelo tempo*

Para testar a formação de grupos de agentes orientada pelo tempo utilizou-se a seguinte metodologia: a cada $\Delta$ passos os agentes são potencialmente reagrupados, utilizando-se o algoritmo *bee clustering*, e atribuídos a outras tarefas de acordo com o conjunto de atributos que representam suas características e também as características da tarefa percebida por ele. Os agentes então realizam a tarefa para a qual foram agrupados até terminá-la ou por $\Delta$ passos quando o agrupamento reiniciará.

Na maioria dos casos, se a tarefa que o agente $i$ estava realizando não for concluída em $\Delta$ passos, existe uma grande probabilidade de $i$ ser atribuído novamente a mesma tarefa já que um dos atributos considerados na realização do agrupamento é a distância entre o agente e a tarefa. Assim, pretende-se verificar se o reagrupamento orientado pelo tempo poderá ser considerado como uma fonte de perda de eficiência.

Quando um agente finalizar sua tarefa ele não permanecerá ocioso, ele realizará uma exploração aleatória do ambiente e escolherá a tarefa mais próxima para realizá-la até que seja reagrupado.

#### *6.2.2.2 Agrupamento orientado por evento*

Nos testes relacionados a formação de grupos de agentes orientada por evento utilizou-se a quantidade de agentes desagrupados como parâmetro. Um agente está desagrupado quando não está participando de nenhum grupo. Assim, sempre que $\rho$ agentes estiverem desagrupados no cenário, o algoritmo *bee clustering* realizará o reagrupamento destes agentes.

Nesta abordagem os agentes realizam a tarefa para a qual foram agrupados até terminá-las.

Da mesma forma que no agrupamento orientado pelo tempo, no agrupamento orientado por evento, quando um agente finalizar sua tarefa ele não permanecerá ocioso, ele realizará uma exploração aleatória do ambiente e escolherá a tarefa mais próxima para realizá-la até que seja reagrupado.

A seguir, na subseção 6.2.3 descreve-se detalhes sobre os experimentos realizados no simulador *RoboCup Rescue* bem como os resultados obtidos.

### 6.2.3 Experimentos e resultados

Para testar a habilidade do algoritmo *bee clustering* utilizou-se uma quantidade de agentes que totaliza duas vezes o número padrão do mapa *Kobe*, já que cada mapa apresenta uma configuração padrão. Desta forma, criou-se 48 agentes que deverão ser agrupados de acordo com as tarefas que devem ser realizadas. Estes agentes estão divididos em: 12 ambulâncias, 20 bombeiros e 16 policiais. Além dos agentes, criou-se também as tarefas que deverão ser realizadas. Inicialmente há 144 civis a serem resgatados e 12 pontos de incêndio a serem controlados.

Os experimentos foram realizados com $\Delta = 10, ..., 50$ para o agrupamento orientado pelo tempo e com $\rho = 20, ..., 40$ para o agrupamento orientado por evento. O reagrupamento é repetido até que a simulação atinja 300 passos (duração de tempo que é amplamente utilizada pela comunidade).

É importante ressaltar que a natureza dos experimentos realizados nesta aplicação é diferente dos experimentos realizados na aplicação I descrita anteriormente, onde se tinha conhecimento da correta classificação das bases de dados. Sendo assim, a avaliação dos resultados obtidos será conduzida de maneira diferente. Para se medir o desempenho do algoritmo utilizou-se o método padrão de pontuação da *RoboCup Rescue*, o qual pode ser computado de acordo com a Equação 6.4, onde $Q$ é a quantidade de civis vivos, $H$ e $H_{init}$ são as condições de saúde dos civis vivos no final e no início da simulação, $B$ indica a área construída que não obteve danos e $B_{max}$ é a área construída inicial. $A$ deve atingir o maior valor possível.

$$A = Q + \frac{H}{H_{init}} * \sqrt{\frac{B}{B_{max}}} \tag{6.4}$$

Os resultados obtidos nas simulações são comparados com aqueles obtidos através de uma estratégia gulosa que aloca tarefas apenas por proximidade, ou seja, agentes que são designados a realizar a tarefa mais próxima de suas localizações sem qualquer explícita coordenação ou formação de grupos. Tal estratégia é denominada *greedy*.

Os resultado obtidos com o algoritmo *bee clustering* não serão comparados com aqueles gerados pelo time ZJU (YIKUN et al., 2008), vencedor da competição *RoboCup Rescue*, pois obviamente a abordagem proposta não será capaz de superá-los, já que não aborda-se aspectos do simulador que não estejam relacionados a alocação de tarefas simplesmente.

Os parâmetros necessários para o algoritmo *bee clustering* foram utilizados com os seguintes valores: $\alpha = 0.05$ and $maxSteps = 300$. Estes valores foram escolhidos após diversos testes com diferentes valores para cada parâmetro. O parâmetro $\mathscr{A}$ que indica o conjunto de abelhas recebe os 48 agentes e $\mathscr{X}$ que representa o conjunto dos atributos das abelhas recebe os respectivos atributos dos agentes e das tarefas percebidas por eles. O peso $W$ para os atributos envolvidos no cálculo da distância Euclidiana foram os seguintes: para o atributo "posição" do agente e da tarefa $W = 0.20$, e para o atributo "tipo" de agente e "tipo" da tarefa $W = 0.80$. Como citado anteriormente, optou-se por dar maior

Tabela 6.5: Pontuação obtida de 10 repetições das simulações: *bee clustering* orientado pelo tempo ($\Delta = 30$), *bee clustering* orientado por evento ($\rho = 30$) e *greedy*.

| | ***Greedy*** | ***Bee Clustering*** | |
|---|---|---|---|
| | | orientado pelo tempo | orientado por evento |
| Melhor | 80.51 | 93.13 | 94.12 |
| Pior | 56.18 | 81.08 | 87.02 |
| Média | 72.65 | **84.97** | **90.23** |
| Desvio Padrão | 8.12 | 4.95 | 2.90 |

representatividade para o atributo "tipo" no cálculo da distância entre o agente e a tarefa para diminuir a possibilidade de um agente ser atribuído à um grupo que realizará uma tarefa para a qual ele não está apto. $\gamma = 0.2$ para que a gravidade da tarefa tenha maior importância no cálculo da probabilidade de abandono $Pa^{l}$.

Tabela 6.5 mostra a pontuação obtida quando utilizou-se o algoritmo *bee clustering* orientado pelo tempo, orientado por evento e quando simulou-se somente a estratégia *greedy*. Entradas na tabela mostram os melhores e piores resultados, a média e o desvio padrão obtido sobre 10 repetições.

Como mencionado anteriormente, optou-se por não realizar a comparação do algoritmo proposto com o time campeão da competição *RoboCup Rescue*, pois o simulador foi utilizado com o objetivo específico de se testar a eficácia do *bee clustering* na formação de grupos de agentes para alocação de tarefas. Já as abordagens propostas para a competição vão além de estratégias para resolver o problema de alocação simplesmente. Seus esforços concentram-se em vencer a competição, utilizando diversas regras e heurísticas para atingir este objetivo, o que contribui significantemente para a obtenção de ótimos resultados. Este é o caso do algoritmo do time ZJU (YIKUN et al., 2008), campeão de 2008, o qual obteve uma pontuação final de 143.98 com desvio padrão de 22.44 mediante 20 repetições no mapa *Kobe*, o mesmo utilizado pelo *bee clustering*.

### 6.2.4 Conclusão

A aplicação II refere-se aos experimentos realizados para a validação do algoritmo *bee clustering* na formação de grupos de agentes no cenário de simulação *RoboCup Rescue*. Primeiramente apresentou-se uma descrição sobre o cenário utilizado seguida pela exposição dos ajustes necessários para possibilitar a formação dos grupos de agentes deste cenário utilizando o algoritmo proposto. Logo após descreveu-se os detalhes sobre os experimentos realizados bem como os resultados obtidos com estes experimentos.

Estes resultados mostraram que a aplicação do algoritmo *bee clustering* em qualquer das duas abordagens utilizadas (orientada pelo tempo ou evento) obtém melhores pontuações do que o algoritmo *greedy* utilizado como comparação. Além disto a abordagem orientada por evento obteve melhores resultados que aquela orientada pelo tempo. Isto se deve ao fato de que quando os agentes estão atuando segundo a abordagem orientada por evento, eles executam a sua tarefa até finalizá-la, enquanto que na outra abordagem a realização da tarefa é interrompida pelo momento do reagrupamento.

Desta forma, os resultados mostram que o algoritmo proposto é eficaz e que conduz a um desvio padrão menor, pois os efeitos aleatórios que têm um papel chave na estratégia *greedy* não são importantes no *bee clustering* já que os agentes atuam de maneira mais

coordenada. Tal coordenação no entanto é atingida sem as entidades centralizadoras que são normalmente usadas pelos competidores da *RoboCup Rescue*.

# 7 CONCLUSÃO

No presente trabalho apresentou-se o *bee clustering*, um algoritmo distribuído inspirado em técnicas de inteligência de enxames para agrupamento de dados. O objetivo principal deste algoritmo é realizar, de maneira distribuída, o agrupamento de um conjunto de dados através da formação de grupos de agentes sem informações iniciais que possam contribuir para o resultado desejado, como por exemplo tamanho dos grupos ou número de grupos.

Existem inúmeros métodos propostos para solução do problema de agrupamento de dados. No entanto, métodos tradicionais têm sido desenvolvidos seguindo uma abordagem centralizada, necessitando que os dados estejam localizados em um único local, o que inviabiliza sua aplicação em contextos como por exemplo a Internet, onde os dados encontram-se distribuídos por razões de tamanho, privacidade, dinâmica, entre outros. Além disso, técnicas centralizadas dependem de estruturas de dados que precisam ser acessadas e modificadas em cada passo da operação de agrupamento, criando assim um ponto único de falha no algoritmo.

Em sistemas multiagente, agentes que desejam cooperar para realizar uma tarefa qualquer devem ter um meio de encontrar grupos de outros agentes de acordo com algum critério para executar tal tarefa de maneira eficiente. Por exemplo, em um cenário complexo composto por muitos agentes com diferentes especialidades e habilidades, estes agentes devem, de maneira distribuída, formar grupos entre si para realizar suas atividades em times com o objetivo de atingir o melhor resultado possível proveniente desta organização. Agrupamento de agentes baseando-se em características e habilidades similares, ou mesmo complementares, também pode ser visto como um problema de agrupamento de dados.

Na tentativa de mitigar tais problemas o algoritmo *bee clustering* utiliza um modelo matemático para formar, distribuidamente, grupos de agentes com características similares. Cada agente representa um objeto que precisa ser agrupado. Sendo assim, os atributos dos dados constituem o conjunto de características do agentes. Estes agentes possuem conhecimento limitado aos integrantes de seu grupo e relacionado ao estado em que se encontram. Além disso, os agentes não são capazes de lembrar as características de agentes de grupos passados. No entanto, mesmo com limitação de conhecimento e sem a otimização de um objetivo global, eles estão aptos a agruparem-se entre si, de maneira que agentes com características similares pertençam ao mesmo grupo.

Os resultados obtidos nos dois cenários apresentados no Capítulo 6 mostraram-se promissores e indicam que o algoritmo *bee clustering* realiza comparativamente bem o agrupamento nos experimentos realizados, já que atingiu bons resultados, especialmente considerando que o algoritmo é distribuído.

A principal limitação apresentada pelo algoritmo proposto está relacionado ao seu

tempo de convergência, o qual além de ser proporcional ao tamanho da base de dados utilizada, também está ligada à dimensionalidade destes dados. Uma possível alternativa seria utilizar outra métrica como valor da utilidade do grupo dos agentes, já que a atual considera o elemento centróide e o seu cálculo é influenciado pela quantidade de atributos dos agentes que participam deste grupo.

Concluindo, o algoritmo *bee clustering* possui várias características que fazem dele uma abordagem útil como: computação distribuída, inexistência de estruturas complexas que caracterizam ponto único de falha, o algoritmo não depende de informações iniciais relacionadas ao número de grupos ou tamanho dos grupos para gerar o agrupamento.

Como trabalho futuro, pretende-se dar continuidade a investigação relacionada a aplicação do algoritmo proposto na área de bioinformática com o objetivo de se descobrir estruturas alternativas nos conjuntos de dados. Resultados preliminares obtidos neste trabalho, que são relacionados a esta aplicação, apontam para uma direção bastante promissora.

Pretende-se ainda, como trabalho futuro, investigar a possibilidade de se realizar o processo de agrupamento considerando a otimização de múltiplos objetivos. Para isso os agentes utilizariam várias utilidades relacionadas aos seus grupos, cada uma otimizando um objetivo diferente.

# REFERÊNCIAS

AGOGINO, A.; TUMER, K. Efficient agent-based cluster ensembles. In: AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, AAMAS '06, 2006, New York, NY, USA. **Proceedings...** ACM, 2006. p.1079–1086.

ANDERBERG, M. R. **Cluster Analysis for Applications**. New York: Academic Press, 1973. 359p.

ASUNCION, A.; NEWMAN, D. J. **UCI Machine Learning Repository**. Disponível em: <http://www.ics.uci.edu/∼mlearn/MLRepository.html>, University of California, Irvine, School of Information and Computer Sciences.

AZZAG, M. et al. AntTree: a new model for clustering with artificial ants. In: EVOLUTIONARY COMPUTATION (CEC 03), 2003, Camberra, Australia. **Proceedings...** IEEE Press, 2003. v.4, p.2642–2647.

BALL, G. H.; HALL, D. J. **ISODATA**: a novel method of data analysis and pattern classification. Stanford, CA: Stanford Research Institute, 1965. NTIS Report. (AD699–616).

BERKHIN, P. **Survey of Clustering Data Mining Techniques**. San Jose, CA: Accrue Software, 2002. Technical Report.

BIN, W. et al. CSIM: a document clustering algorithm based on swarm intelligence. In: EVOLUTIONARY COMPUTATION (CEC 02), 2002, Los Alamitos, CA, USA. **Proceedings...** New York: IEEE Computer Society, 2002. v.1, p.477–482.

BIN, W.; ZHONGZHI, S. A clustering algorithm based on swarm intelligence. In: INTERNATIONAL CONFERENCES ON INFOTECH AND INFONET (ICII), 2001, Beijing. **Proceedings...** [S.l.: s.n.], 2001. v.3, p.58–66.

BONABEAU, E.; THERAULAZ, G.; DORIGO, M. **Swarm Intelligence**: from natural to artificial systems. New York, USA: Oxford University Press, 1999. 307p.

CAMAZINE, S. et al. **Self-Organization in Biological Systems**. Princeton, N.J.: Princeton University Press, 2003. 538p.

CAMAZINE, S.; SNEYD, J. A Model of Collective Nectar Source Selection by Honey Bees: self-organization through simple rules. **Journal of Theoretical Biology**, [S.l.], v.149, n.4, p.547–571, April 1991.

CARO, D. G.; DORIGO, M. **AntNet**: a mobile agents approach for adaptive routing. Belgium: Université Libre de Bruxelles, Belgium, 1997. Technical Report. (IRIDIA/97–12).

CHANDRASEKAR, R.; SRINIVASAN, T. An Improved Probabilistic Ant based Clustering for Distributed Databases. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, IJCAI, 20., 2007, Hyderabad, India. **Proceedings...** [S.l.: s.n.], 2007. p.2701–2706.

CHANDRASEKAR, R.; VIJAYKUMAR, V.; SRINIVASAN, T. Probabilistic Ant based Clustering for Distributed Databases. In: INTERNATIONAL IEEE CONFERENCE ON INTELLIGENT SYSTEMS, 3., 2006. **Proceedings...** [S.l.: s.n.], 2006. p.538–545.

DORIGO, M.; DI CARO, G. The Ant Colony Optimization Meta-Heuristic. In: CORNE, D.; DORIGO, M.; GLOVER, F. (Ed.). **New Ideas in Optimization**. London: McGraw-Hill, 1999. p.11–32.

DORIGO, M.; DI CARO, G.; GAMBARDELLA, L. Ant Colony Optimization: A new meta-heuristic. In: CONGRESS ON EVOLUTIONARY COMPUTATION, 1999, Mayflower Hotel, Washington D.C., USA. **Proceedings...** IEEE Press, 1999. v.2, p.1470–1477.

DORIGO, M.; GAMBARDELLA, L. M. Ant Colony System: a cooperative learning approach to the traveling salesman problem. **IEEE Transactions on Evolutionary Computation**, [S.l.], v.1, n.1, p.53–66, April 1997.

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**, [S.l.], v.26, n.1, p.29–41, 1996.

FACELI, K.; CARVALHO, A. C. P. L. F.; SOUTO, M. C. P. Multi-objective Clustering Ensemble. In: SIXTH INTERNATIONAL CONFERENCE ON HYBRID INTELLIGENT SYSTEMS (HIS 06), 2006, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2006. p.51.

GAMBARDELLA, L. M.; TAILLARD, E. D.; DORIGO, M. Ant colony for the QAP. **Journal of Operational Research Society**, [S.l.], v.50, p.167–176, 1998.

GORDON, D. The Organization of Work in Social Insect Colonies. **Nature**, [S.l.], v.380, p.121–124, 1996.

HANDL, J.; KONWLES, J. Exploiting the trade-off - The benefits of multiple objectives in data clustering. In: THIRD INTERNATIONAL CONFERENCE ON EVOLUTIONARY MULTI-CRITERION OPTIMIZATION (EMO 2005), 2005. **Proceedings...** Springer Verlag, 2005. p.547–560.

JAIN, A. K.; DUBES, R. C. **Algorithms for Clustering Data**. New Jersey: Prentice-Hall, 1988. 320p.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data Clustering: a review. **ACM Computing Surveys**, New York, NY, USA, v.31, n.3, p.264–323, September 1999.

JOHNSON, E. **Emergência**: a vida integrada de formigas, cérebros, cidades e softwares. Rio de Janeiro, BR: Jorge Zahar, 2003. 232p.

KAO, Y.; CHENG, K. An ACO-Based Clustering Algorithm. In: FIFTH INTERNATIONAL WORKSHOP ON ANT COLONY OPTIMIZATION AND SWARM INTELLIGENCE - ANTS 2006, 2006, Brussels, Belgium. **Proceedings...** Springer, 2006. p.340–347. (Lecture Notes in Computer Science, v.4150).

KITANO, H. et al. RoboCup Rescue: search and rescue in large-scale disasters as adomain for autonomous agents research. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC), 1999, Tokyo, Japan. **Proceedings...** [S.l.: s.n.], 1999. v.6, p.739–743.

KOGAN, J.; NICHOLAS, C.; TEBOULLE, M. **Grouping Multidimensional Data**: recent advances in clustering. Secaucus, NJ, USA: Springer-Verlag, 2006. 268p.

LABROCHE, N.; MONMARCHÉ, N.; VENTURINI, G. A new clustering algorithm based on the chemical recognition system of ants. In: EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE (ECAI 2002), 15., 2002, France. **Proceedings...** IOS Press, 2002. p.345–349.

LUMER, E. D.; FAIETA, B. Diversity and adaptation in populations of clustering ants. In: SIMULATION OF ADAPTIVE BEHAVIOR: FROM ANIMALS TO ANIMATS, 1994, Cambridge, MA, USA. **Proceedings...** MIT Press, 1994. p.501–508.

MONMARCHÉ, N.; SLIMANE, M.; VENTURINI, G. **AntClass**: discovery of clusters in numeric data by an hybridization of an ant colony with the k-means algorithm. E3i Tours: Laboratoire d'Informatique de l'Université de Tours, 1999. Rapport interne. (213).

ROMESBURG, H. C. **Cluster Analysis for Researchers**. North Carolina: Lulu Press, 2004. 344p.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence**: a modern approach. [S.l.]: Prentice-Hall, 1995. 932p.

SANTOS, D. S. d.; OLIVEIRA, D.; BAZZAN, A. L. C. A Multiagent, Multiobjective Clustering Algorithm. In: CAO, L. (Ed.). **Data Mining and Multiagent Integration**. [S.l.]: Springer, 2009. Under revision.

SEELEY, D.; CAMAZINE, S.; SNEYD, J. Collective Decision-Making in Honey Bees: how colonies choose nectar sources. **Behavioral Ecology and Sociobiology**, [S.l.], v.28, p.277–290, 1991.

SHELOKAR, P. S.; JAYARAMAN, V. K.; KULKARNI, B. D. An ant colony approach for clustering. **Analytica Chimica Acta**, [S.l.], v.509, n.2, p.187–195, 2004.

STREHL, A.; GHOSH, J. Cluster Ensembles: a knowledge reuse framework for combining partitionings. In: EIGHTEENTH NATIONAL CONFERENCE INTELLIGENCE, 2002, Menlo Park, CA, USA. **Proceedings...** American Association for Artificial Intelligence, 2002. p.93–98.

VIZINE, A. L. et al. Towards Improving Clustering Ants: an adaptive ant clustering algorithm. **Informatica (Slovenia)**, [S.l.], v.29, n.2, p.143–154, 2005.

WITOLD, P. **Knowledge-Based Clustering**: from data to information granules. Hoboken, NJ: Wiley-Interscience, 2005. 336p.

YANG, Y.; KAMEL, M. An aggregated clustering approach using multi-ant colonies algorithms. **Pattern Recongnition**, New York, NY, USA, v.39, n.7, p.1278–1289, 2006.

YEOH, E. et al. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. **Cancer Cell**, [S.l.], v.1, n.2, p.133–143, 2002.

YIKUN, T. et al. **Robocup Rescue 2008 Repository - ZJUBase team**. 2008.