Aplicações de Agrupamento Distribuído baseado em Inteligência de Enxames em Sistemas Multiagente

Daniela Scherer dos Santos e Ana L. C. Bazzan

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

{daniela.scherer,bazzan}@inf.ufrgs.br

Abstract. *Traditional clustering methods have been usually developed in a centralized fashion; additionally, they need some hints about the target clustering (e.g. number of clusters, expected cluster size, or minimum density of clusters). However this does not meet a typical necessity in multiagent scenarios that is self-organization without central control. In this work we use a clustering algorithm that is inspired by swarm intelligence techniques, is distributed, and does not require any initial hint about the number of clusters. Tests using two applications – one employing typical public datasets and one in a dynamic scenario – show the formation of groups in a distributed way with a performance that is comparable to that achieved using centralized approaches.*

1. Introdução

Em agrupamento a busca pela melhor solução é um problema NP-Difícil. Diante da grande quantidade de possíveis caminhos para agrupar dados, é natural a existência de diversas técnicas. 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 sistemas multiagente onde o conhecimento é distribuído. Outro inconveniente é que muitos destes métodos necessitam de algumas "pistas" sobre o agrupamento desejado, como por exemplo número de grupos a serem gerados, tamanho esperado para cada grupo ou mesmo a densidade máxima ou mínima dos grupos. Além disto eles são baseados em estruturas de dados que precisam ser acessadas e modificadas a 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 se organizarem em grupos com outros agentes, de acordo com algum critério, para executar esta 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. Este é um dos focos deste trabalho. Adicionalmente, o algoritmo distribuído discutido aqui, *bee clustering*, é aplicado a conjuntos de dados comumente utilizados para testes de técnicas de agrupamento.

O *bee clustering* forma grupos de acordo com as características dos dados ou habilidades dos agentes. O algoritmo é capaz de agrupar estes dados ou agentes de uma maneira descentralizada e sem informação inicial. Isto constitui-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.

Na próxima seção são apresentadas algumas abordagens para agrupamento relacionadas ao algoritmo *bee clustering*. Na Seção 3 descreve-se este algoritmo, o qual é utilizado em dois tipos distintos de aplicações, uma típica de problemas de agrupamento e outra relacionada ao problema de alocação de tarefas em sistemas multiagente (Seção 4). As conclusões obtidas neste trabalho são relatadas na Seção 5.

2. Técnicas para agrupamento de dados

Diferentes técnicas para agrupamento de dados são descritas na literatura. Um algoritmo amplamente usado é o *K-means*, que busca minimizar a distância dos elementos a um conjunto de *k* centros de forma iterativa. Dados *n* objetos a serem agrupados em *k* grupos, o algoritmo parte de um agrupamento inicial, calcula os pontos médios dos grupos, e realiza uma iteração realocando os objetos entre os grupos de acordo com as distâncias entre os objetos e esses pontos médios. O *K-means* depende de um parâmetro (*k*=número de grupos) definido pelo usuário e exige portanto algum conhecimento prévio.

Um importante representante da classe de algoritmos baseados em inteligência de enxames e, por consequência, inspirado em comportamentos coletivos exibidos pelas colônias de insetos sociais, é o *Ant Clustering Algorithm* (ACA) [Bonabeau et al. 1999]. O ACA é inspirado no comportamento de organização de cemitérios, no qual formigas separam do restante do ninho os corpos de formigas mortas formando um cemitério. No ACA formigas movem-se aleatoriamente em uma grade bidimensional e quando encontram um objeto têm uma grande probabilidade de coletá-lo se este estiver localizado em uma região com pouca concentração de outros objetos. Da mesma forma, a probabilidade de depositar um objeto coletado 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. A abordagem é centralizada.

Dentre os algoritmos distribuídos destacamos o *Improved Probabilistic Ant based Clustering* (I-PACE) [Chandrasekar and Srinivasan 2007], por se tratar de um algoritmo para agrupamento distribuído de dados inspirado em inteligência de enxames. O I-PACE também utiliza como inspiração biológica o comportamento de organização de cemitérios 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, o I-PACE realiza o agrupamento através da formação de várias zonas de acordo com três fases: fase de consulta a base de dados, fase de atribuição de probabilidades e a fase de agrupamento.

Por fim escolhemos, para efeito de comparação, um algoritmo hierárquico, o *average link*, o qual constrói uma matriz de distâncias entre os grupos a partir do conjunto de dados que deve ser agrupado. Inicialmente cada dado é 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. O algoritmo finaliza quando o correto número de grupos é encontrado. Desta forma, os principais problemas apresentados pelo algoritmo *average link* são: a definição *a priori* do número de grupos; 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; e o algoritmo depende de informações que

encontram-se centralizadas em uma matriz de distâncias.

3. Bee Clustering

Os principais problemas detectados na maioria dos algoritmos apresentados na seção anterior estão relacionados ao fato de que seus resultados dependem tanto de informações importantes que precisam ser definidas *a priori* (número de grupos, tamanho destes grupos, entre outros) quanto de estruturas centralizadas de dados. 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.

Na tentativa de superar os problemas acima mencionados, foi desenvolvido o algoritmo *bee clustering* [Santos and Bazzan 2009], o qual tem como objetivo formar, de maneira distribuída, grupos de dados com características similares sem qualquer informação inicial relacionada ao resultado desejado. O *bee clustering* é inspirado principalmente em técnicas de inteligência de enxames como organização de colônias de abelhas e alocação de tarefas em insetos sociais. No primeiro caso trata-se do modelo matemático de recrutamento em colônias de abelhas [Camazine and Sneyd 1991]. 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 na qual uma abelha comunica a direção, distância e qualidade de uma fonte de néctar para outras abelhas. Esta dança é utilizada pelo *bee clustering* para formar grupos de abelhas. Neste algoritmo as abelhas dançam para recrutar novos indivíduos para participarem de seus grupos. O conjunto de atributos de um determinado objeto constitui o conjunto de características daquela abelha que o representa. Através deste comportamento, as abelhas do algoritmo são capazes de se organizarem em grupos de acordo com suas características ou habilidades e de maneira distribuída.

A Figura 1 mostra uma visão geral de como acontece todo o processo de agrupamento. Cada abelha possui um conjunto de possíveis estados $\mathscr{S} = \{d, v, w\}$. Quando o estado é $\delta = d$, a abelha está dançando com o objetivo de recrutar/convidar outras abelhas a fazerem parte de seu grupo; se $\delta = v$ a abelha está visitando uma outra abelha; e se $\delta = w$ a abelha está assistindo a dança do recrutamento com o objetivo de escolher uma nova abelha para visitar.

Os principais parâmetros utilizados pelo *bee clustering* são: $\mathscr{A}$ que representa o conjunto de todas as abelhas; $\mathscr{X}$ o conjunto das características das abelhas; $\mathscr{S}$ o conjunto dos possíveis estados de cada abelha; *Pa* que indica a probabilidade de uma abelha abandonar outra; *Pv* é a probabilidade de uma abelha visitar outra; *Pd* representa a probabilidade de uma abelha continuar executando a dança a fim de convidar outras para fazerem parte de seu grupo; $U(k)$ é o valor da utilidade do grupo k; n indica a quantidade de abelhas no grupo; D representa a quantidade de abelhas que estão dançando para um grupo no instante t; c é o centróide de um grupo; e μ é um parâmetro que controla o número máximo de iterações do algoritmo.

Cada abelha possui um conhecimento limitado com relação às demais abelhas.

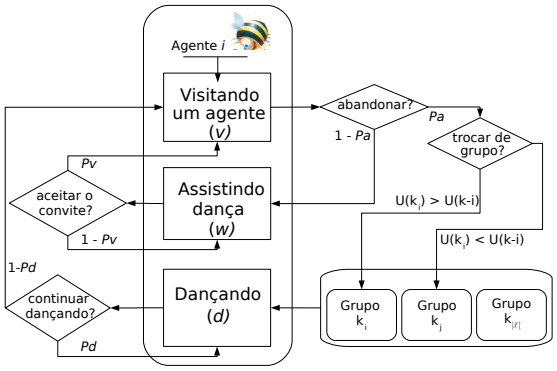

Figura 1. Processo de agrupamento realizado pelo *bee clustering*

Quando o estado da abelha é $\delta = v$ ou $\delta = d$ ela conhece apenas aquelas abelhas que pertencem ao seu grupo no instante atual (t), ou seja, a partir do momento que uma determinada abelha deixa de participar de um grupo, ela é esquecida por todas as demais integrantes daquele grupo e passa a ser conhecida somente pelas participantes de seu novo grupo. Quando o estado da abelha é $\delta = w$, ela tem conhecimento sobre os indivíduos que estão no seu grupo e também sobre aquele subconjunto de abelhas cujo estado é $\delta = d$.

No início do processo de agrupamento, cada abelha é considerada como um grupo individual. A seguir ocorrem visitas entre as abelhas as quais possibilitam a verificação da similaridade existentes entre elas. Durante o processo de formação dos grupos, as abelhas precisam tomar algumas decisões importantes sobre abandonar ou não uma abelha que está visitando, aceitar ou não um convite para visitar uma abelha, dançar ou não para recrutar abelhas para o seu grupo e também sobre trocar ou não de grupo. Excetuando-se a decisão sobre troca de grupo, todas as demais são tomadas de maneira probabilística. Os losangos mostrados na Figura 1 representam os momentos em que estas decisões precisam ser tomadas e serão explicados a seguir.

3.1. Decisão sobre abandonar uma abelha

A primeira decisão probabilística que uma abelha precisa tomar está relacionada a abandonar ou não a abelha que ela está visitando. Para tomar esta decisão, é necessário que ela verifique se existem semelhanças entre suas características e as da abelha que está visitando. Esta verificação é realizada através da distância Euclidiana. Supondo-se que a abelha i esteja visitando a abelha j, i verifica quão similar é à j calculando a distância Euclidiana existente entre seu conjunto de atributos $\mathscr{X}$ e os atributos da abelha j. A abelha i utiliza o resultado obtido com o cálculo da distância Euclidiana $d(i, j)$ normalizada como a probabilidade Pa de abandonar a abelha j. Quanto menor o valor de Pa, menor é a probabilidade da abelha i abandonar j, o que significa que as duas abelhas possuem um alto índice de similaridade.

3.2. Decisão sobre trocar de grupo

Sempre que a probabilidade de abandono Pa não for satisfeita para um par (i, j) de abelhas, a abelha i precisa decidir se troca ou não para o grupo da abelha j, a qual está visitando. Para esta tomada de decisão buscou-se inspiração em [Agogino and Tumer 2006]

onde os autores propõem um método para encontrar, de forma distribuída, o *ensemble* consenso relativo à um agrupamento de dados. Nesta abordagem, as abelhas precisam maximizar a utilidade global do agrupamento de maneira distribuída. No *bee clustering* as abelhas tentam maximizar uma utilidade local relacionada ao grupo ao qual pertencem. As abelhas decidem se trocam de grupo ou não de acordo com a qualidade do seu grupo atual. Esta qualidade 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 a abelha i se encontra, deve ser maximizada e é calculada da seguinte forma: $U(k_i) = 1 - var(k_i)$. $var(k_i)$ indica o valor da variância intra-*cluster* do grupo k da abelha i. $var(k_i) = \sqrt{\frac{1}{n}\sum_{i \in k} d^2(i, c_i)}$ e deve ser minimizada. n é o número de elementos dentro do grupo k_i, e $d(i, c_i)$ é a distância Euclidiana entre a abelha i e o centróide c_i do grupo k_i, sendo $c_i = \frac{1}{n}\sum_{i \in k} i$.

Para verificar se abandona ou não o grupo k_i, a abelha i calcula a utilidade $U(k_i)$ e a utilidade $U - i(k_i)$, a qual indica a utilidade de seu grupo k_i sem a sua participação. Se $U(k_i)$ for maior que $U - i(k_i)$, indicando que o a utilidade do grupo k_i é melhor com a presença de i, a abelha i permanece no grupo k_i. Caso contrário, i abandona k_i e passa a integrar o grupo k_j da abelha j que ela está visitando. Como pode ser visto, i não possui conhecimento algum sobre a qualidade do grupo da abelha j. Desta forma, tal informação não é utilizada por i no processo de tomada de decisão sobre a troca de grupo. Para esta decisão a abelha 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 uma abelha provavelmente similar a ela, possivelmente i também será similar às abelhas que pertencem ao grupo k_j da abelha j.

Trocando ou não de grupo, a próxima ação de i será executar a dança do recrutamento com o objetivo de convidar outras abelhas a fazerem parte de seu grupo, seja este um novo grupo ou não.

3.3. Decisão sobre recrutar abelhas

No *bee clustering* a dança de recrutamento é utilizada pelas abelhas para recrutar outros indivíduos para fazerem parte de seus grupos. A dança é executada por todas as abelhas cujo estado for $\delta = d$ e estas abelhas precisam decidir se continuam ou não neste estado. O tempo que uma abelha permanece dançando para seu grupo k é uma questão crucial no processo de agrupamento. Se as abelhas permanecem dançando por um longo período, todas as abelhas tendem a dançar ao mesmo tempo. Por outro lado, se as abelhas permanecem dançando por um período muito curto, o algoritmo converge para um agrupamento com uma grande quantidade de pequenos grupos. Para controlar o tempo que a abelha permanece dançando utilizou-se um mecanismo inspirado no modelo de divisão de trabalho exibido pelos insetos sociais [Bonabeau et al. 1999]. Neste modelo, as abelhas utilizam um estímulo associado a um limiar para, probabilisticamente, decidir se realizam uma tarefa ou não. No *bee clustering* as abelhas utilizam este mesmo estímulo S associado a um limiar θ para auxiliá-las na tomada de decisão sobre continuar ou não dançando para um determinado grupo. Os valores de S e θ são utilizados no cálculo da probabilidade Pd de continuar dançando: $Pd = \frac{S_i^2}{S_i^2 + \theta_i^2}$.

Quanto melhor a qualidade do grupo da abelha i, maior será o estímulo S e por-

tanto maior será o valor de Pd. Como os grupos são dinâmicos, ou seja, a cada ciclo podem receber novas abelhas ou também perder alguns participantes, a probabilidade Pd da abelha i é atualizada por ela a cada ciclo, enquanto seu estado for $\delta = d$, de acordo com as seguintes regras. Se a utilidade $U(k_i)$ do grupo k da abelha i no tempo t é maior que a utilidade $U(k_i)$ no tempo $t-1$, o estímulo S_i é incrementado e o limiar θ_i é decrementado da constante α, aumentando a probabilidade Pd da abelha continuar dançando. Ou seja: $S_i = S_i + \alpha$ e $\theta_i = \theta_i - \alpha$. 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 θ_i é incrementado: $S_i = S_i - \alpha$ e $\theta_i = \theta_i + \alpha$.

Sempre que Pd for satisfeita, o estado da abelha permanecerá $\delta = d$ para que ela continue dançando e recrutando novos indivíduos para seu grupo. Quando Pd não for satisfeita, a abelha encerra a dança de recrutamento mas permanece no mesmo grupo. Enquanto i está dançando existem abelhas que estão assistindo sua dança com o objetivo de serem guiadas para um grupo que contenha indivíduos mais similares a ele do que os integrantes de seu grupo atual.

3.4. Decisão sobre aceitar um convite

Outro momento de decisão acontece quando a abelha i está no estado $\delta = w$, o que significa que ele está assistindo à dança do recrutamento executada por outras abelhas da colônia. Neste instante, i sorteia aleatoriamente, com probabilidade uniformemente distribuída, uma abelha j cujo estado seja $\delta = d$ e visita esta abelha j com probabilidade Pv. $Pv = \frac{D(k_j)}{n}$, onde $D(k_j)$ é o número de abelhas que estão dançando para recrutar abelhas para o grupo k_j e n é o número de abelhas que pertencem ao grupo da abelha j. Sendo assim, quanto maior o número de abelhas dançando para um determinado grupo, maior será a chance de outras abelhas testarem este grupo.

4. Validação da abordagem proposta

Para testar a eficácia do algoritmo em agrupar abelhas que representam um conjunto de dados utilizou-se dois tipos de aplicação. Na primeira realizou-se experimentos em bases de dados de domínio público, das quais se conhece seu correto agrupamento. Como segunda aplicação utilizou-se o ambiente de simulação *RoboCup Rescue* que se traduz parcialmente no problema de agrupar agentes que devem operar de forma colaborativa.

4.1. Aplicação I

Aqui foram utilizadas algumas bases de dados de domínio público: *iris*, *yeast* e *wine*, as quais podem ser encontradas no repositório da *UCI*. A base *Iris* contém 3 classes representando 3 tipos de flores. 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. A base *Yeast* contém 1484 elementos, cada um com 8 atributos, que podem ser divididos em 10 classes com os seguintes tamanhos: 463, 429, 244, 163, 51, 44, 35, 30, 200, e 5. A base *Wine* contém 178 dados, cada um com 13 atributos. Os dados estão divididos em 03 classes as quais representam 03 tipos de vinhos e possuem os seguintes tamanhos: 59, 71 e 48. No *bee clustering* as classes 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 abelhas, sendo portanto representado por $|\mathscr{A}|$. $|\mathscr{X}|$

Tabela 1. Média e desvio padrão do *F-measure* e índice *Rand* (60 repetições) para *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	0.8168	0.8098	**0.8324**	0.8205
	(0.0848)	(0.0148)	–	(0.0001)	(0.0380)
Rand	0.8165	0.8254	0.8223	**0.8300**	0.8153
	(0.1012)	(0.0080)	–	(0.0008)	(0.0184)
Grupos identificados	3	3.02	3	3	3.04
Yeast					
F-measure	0.4315	0.4353	0.4483	0.4348	**0.4893**
	(0.0044)	(0.0345)	–	(0.0899)	(0.0261)
Rand	0.7506	0.6781	0.7426	0.7992	**0.8172**
	(0.0012)	(0.0752)	–	(0.0009)	(0.0579)
Grupos identificados	10	5.36	10	10	10.3
Wine					
F-measure	**0.9312**	0.8760	0.9255	0.8991	0.8048
	(0.0615)	(0.0208)	–	(0.0002)	(0.0345)
Rand	**0.9167**	0.8554	0.9044	0.8290	0.8153
	(0.0653)	(0.0191)	–	(0.0008)	(0.0597)
Grupos identificados	3	3	3	3	3.1

indica a dimensionalidade dos dados da base, ou seja, o número de atributos que cada dado possui.

Os experimentos realizados se restringiram a estas bases de dados para possibilitar a comparação com resultados reportados na literatura. Os resultados utilizados para comparar o algoritmo são provenientes dos seguintes algoritmos: *K-means*, *ant-based clustering* (ACA), *average link* e I-PACE. A avaliação do agrupamento obtido com a utilização do *bee clustering* foi realizada através do cálculo das medidas *F-measure*, do índice *Rand* e do número de grupos gerados, como largamente utilizado na literatura.

Os valores utilizados para os parâmetros foram: $\alpha = 0.02$, $maxSteps = |\mathscr{A}| \times \mu$ e $\mu = 6$. Escolheu-se estes valores após a realização de diversos testes com diferentes valores para cada parâmetro. O número de abelhas $|\mathscr{A}|$ e a quantidade de atributos $|\mathscr{X}|$ possuem valores diferentes para cada base a ser utilizada. A tabela 1 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 já mencionados. O valor do desvio padrão é mostrado entre parênteses.

Como pode ser visto na tabela 1, para a base *yeast* o algoritmo superou os resultados obtidos pelos demais tanto pelo *F-measure* quanto pelo índice *Rand* e foi capaz de, na maioria das simulações, encontrar automaticamente o correto número de grupos. Para as bases iris e *wine* o algoritmo não foi capaz de superar os resultados obtidos pelo I-PACE e *K-means* respectivamente, embora para o conjunto iris tenha obtido resultados bem próximos. No entanto, é importante ressaltar que o *bee clustering* não requer informações *a priori*, como é o caso da maioria dos demais. Além disso, o *bee clustering* realiza o agrupamento de forma distribuída possibilitando sua aplicação em cenários onde esta característica seja necessária, como por exemplo domínios onde os dados estão distribuídos por motivos de segurança, privacidade, entre outros.

4.2. Aplicação II

Outro cenário utilizado para se verificar a eficácia do algoritmo *bee clustering* foi o ambiente de simulação *Robocup Rescue* visando se investigar o desempenho do algoritmo na

formação de grupos de abelhas em ambientes extremos e difíceis. Este ambiente é **dinâmico** pois simula operações de busca e resgate após o acontecimento de um terremoto. O ambiente muda continuamente pelos desabamentos de prédios, propagação de incêndios e bloqueios por escombros. Agentes de busca e resgate precisam navegar neste cenário tentando salvar vidas e limitar a propagação dos incêndios. Como entrada, o simulador recebe dados geográficos (mapas) e outras informações com as quais modela desabamentos, bloqueios de ruas, incêndios e civis feridos. O simulador modela ainda agentes policiais, bombeiros e ambulâncias. Estes agentes têm habilidades específicas: ambulâncias são aptas a localizar civis nos escombros e transferi-los; os bombeiros são aptos a extinguir os incêndios e os policiais podem desbloquear ruas. Entretanto, seu agrupamento depende de diversos atributos. O atributo "posição" indica a posição atual do agente e da tarefa. O atributo "tipo" do agente indica se ele é bombeiro, policial ou 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. Para os experimentos realizados neste trabalho utilizou-se o mapa *Kobe*, o qual é amplamente usado pela comunidade que trabalha com o simulador. Cada agente move-se de maneira aleatória para uma direção e pode perceber tarefas. 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.

Nesta aplicação a probabilidade de abandono *Pa* foi adaptada com o intuito de refletir as necessidades específicas do cenário (que tem tipos distintos de agentes e tarefas): $Pa' = (dp(i,T_j) \times \gamma) + (\beta \times (1-\gamma))$, onde β representa a gravidade da tarefa T_j e $dp(i,T_j)$ é a distância Euclidiana ponderada entre a abelha i e a tarefa T_j percebida pela abelha j. A distância $dp(i,T_j)$ e a gravidade β são ponderados no cálculo de Pa' por γ e $1-\gamma$ respectivamente com $\gamma \in [0,1]$. Para o cálculo de $dp(i,T_j)$ considerou-se os atributos "posição" e "tipo", os quais foram normalizados, juntamente com β, para o intervalo [0,1] para garantir que Pa' também respeite este intervalo. A utilidade $U(k_i)$ é dada por: $\frac{n}{u}$ se a tarefa é um bloqueio; $var(T_{k_i})$ se é um incêndio; e $\frac{n}{r}$ se é resgate de civis. 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 variância intra-*cluster* de k_i. Sendo assim, para uma abelha i decidir se sai do grupo k_i e se engaja no grupo da abelha j, ela calcula a utilidade de seu grupo de acordo com o tipo de tarefa que seu grupo vai realizar.

A frequencia de mudanças em relação ao número de tarefas no cenário da *RoboCup Rescue* é um desafio para a formação de grupos de agentes, pois estes precisam ser reagrupados a intervalos constantes de tempo, ou mediante o acontecimento de algum evento. No presente trabalho experimentou-se o reagrupamento dos agentes em ambos os casos: orientado a tempo e também a evento, onde este evento ocorre quando um determinado número de agentes termina sua tarefa. Para testar a formação de grupos de agentes orientada a tempo utilizou-se o seguinte método: a cada Δ passos os agentes são reagrupados via o *bee clustering*, e atribuídos a outras tarefas de acordo com o conjunto de atributos que representam suas características e também as características das tarefas percebida por ele. Os agentes então realizam a tarefa para a qual foram agrupados até terminá-la ou por Δ passos. Na maioria dos casos, se a tarefa que o agente i estava reali-

zando não for concluída em Δ passos, existe uma grande probabilidade de i ser atribuído novamente à mesma tarefa já que um dos atributos considerados na realização do agrupamento é a distância entre o agente e a tarefa. Quando um agente finalizar sua tarefa ele não permanecerá ocioso, mas sim realizará uma exploração aleatória do ambiente e escolherá a tarefa mais próxima para realizá-la até que seja reagrupado. Nos testes relacionados a formação de grupos de agentes orientada a evento utilizou-se a quantidade de agentes ociosos como parâmetro. Assim, sempre que ρ agentes estiverem ociosos no cenário, o *bee clustering* realizará o reagrupamento apenas destes agentes.

No mapa *Kobe* foram inseridos: 12 ambulâncias, 20 bombeiros e 16 policiais (48 agentes a serem agrupados), bem como as seguintes tarefas: 144 civis a serem resgatados, 12 pontos de incêndio a serem controlados, além dos bloqueios criados pela dinâmica do cenário. Os experimentos foram realizados com $\Delta = 10, ..., 50$ para o agrupamento orientado a tempo e com $\rho = 20, ..., 40$ para o agrupamento orientado a evento. O reagrupamento é repetido até que a simulação atinja 300 passos (duração de tempo que é amplamente utilizada pela comunidade), quando o score é calculado através da fórmula: $A = Q + \frac{H}{H_{init}} \times \sqrt{\frac{B}{B_{max}}}$. Aqui 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 teve danos e B_{max} é a área construída inicial. O score A deve ser o maior possível.

É importante ressaltar que a natureza dos experimentos realizados nesta aplicação é diferente daqueles realizados na aplicação I onde se tinha conhecimento da correta classificação das bases de dados. Aqui os resultados obtidos nas simulações são comparados com aqueles obtidos através de uma estratégia gulosa que aloca tarefas apenas por proximidade, ou seja, agentes são designados a realizar a tarefa mais próxima sem qualquer coordenação explícita ou formação de grupos. 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, não mostrados aqui devido à falta de espaço. A Tabela 2 mostra o score obtido com o algoritmo *bee clustering* orientado a tempo, orientado a evento e com a estratégia gulosa. Estes resultados mostraram que a aplicação do *bee clustering* em qualquer das duas abordagens utilizadas (orientada a tempo ou a evento) obtém melhores pontuações do que o algoritmo guloso. Além disto a abordagem orientada a evento obteve melhores resultados que aquela orientada a tempo. Isto se deve ao fato de que quando os agentes estão atuando segundo a abordagem orientada a evento, eles executam a sua tarefa até finalizá-la, enquanto que na outra abordagem a realização da tarefa pode ser interrompida. Desta forma, os resultados mostram que o algoritmo é eficaz e que conduz a um desvio padrão menor, pois os efeitos aleatórios que têm um papel chave na estratégia gulosa não são importantes no *bee clustering* já que os agentes atuam de maneira mais coordenada. Tal coordenação no entanto é atingida sem as os recursos que são normalmente usadas pelos competidores da *RoboCup Rescue*.

5. Conclusão

Neste trabalho foram apresentadas duas aplicações do algoritmo *bee clustering*. O objetivo principal deste é realizar, de maneira distribuída, o agrupamento de um conjunto de dados, sem informações iniciais, como por exemplo tamanho dos grupos ou número de grupos. A primeira aplicação tem o objetivo de comparar o *bee clustering* com outros

Tabela 2. Score (10 repetições): *bee clustering* orientado a tempo ($\Delta = 30$), *bee clustering* orientado a evento ($\rho = 30$) e algoritmo guloso.

	guloso	***Bee Clustering***	
		orientado a tempo	orientado por evento
Média	72.65	**84.97**	**90.23**
Desvio Padrão	8.12	4.95	2.90

métodos de agrupamento em aplicações típicas. A segunda aplicação visa um problema clássico em sistemas multiagente, o agrupamento de agentes com finalidade de alocação de tarefas, o que é sabidamente um problema complexo. Os resultados obtidos nos dois cenários mostraram-se promissores e indicam que o *bee clustering* realiza bem os agrupamentos, especialmente considerando que o algoritmo é distribuído. Em suma, o *bee clustering* possui várias características úteis como: computação distribuída, inexistência de estruturas complexas que caracterizam ponto único de falha, não depende de informações iniciais relacionadas ao número ou tamanho dos grupos.

A principal limitação apresentada pelo algoritmo está relacionado ao seu tempo de convergência, o qual além de ser proporcional ao tamanho da base de dados utilizada, também está ligado à dimensionalidade destes dados. Uma possível alternativa seria utilizar outra métrica como valor da utilidade, já que a atual considera o elemento centróide.

Agradecimentos

Pesquisa parcialmente financiada pelo *Air Force Office of Scientific Research* (AFOSR) (FA9550-06-1-0517) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Referências

Agogino, A. and Tumer, K. (2006). Efficient agent-based cluster ensembles. In Stone, P. and Weiss, G., editors, *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, AAMAS '06*, pages 1079–1086, New York, NY, USA. ACM.

Bonabeau, E., Theraulaz, G., and Dorigo, M. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, USA.

Camazine, S. and Sneyd, J. (1991). A model of collective nectar source selection by honey bees: Self-organization through simple rules. *Journal of Theoretical Biology*, 149(4):547–571.

Chandrasekar, R. and Srinivasan, T. (2007). An improved probabilistic ant based clustering for distributed databases. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI*, pages 2701–2706, Hyderabad, India.

Santos, D. S. d. and Bazzan, A. L. C. (2009). A biologically-inspired distributed clustering algorithm. In *Proc. of the 2009 IEEE Swarm Intelligence Symposium*, pages 160–167, Nashville. IEEE.