Distributed Clustering: an Application on Bioinformatics

Daniela S. Santos and Ana L. C. Bazzan

Instituto de Informática, UFRGS
C.P. 15064, 91501-970, P.Alegre, RS, Brazil
{daniela.scherer, bazzan}@inf.ufrgs.br

Abstract. This paper applies Bee clustering, a biologically-inspired clustering algorithm to datasets of interest for bioinformatics. Bee clustering can handle distributed data, a typical necessity in scenarios collaborative work in genomics and proteomics where laboratories form networks and the data is distributed. Experimental evaluation shows that our algorithm inspired by the organization of bee colonies is able to find results that are comparable to those from centralized approaches.

1 Introduction

Clustering is widely used in bioinformatics to separate a data set into groups of similar objects. Clustering techniques have proven to be helpful to understand gene function, gene regulation, cellular processes, and subtypes of cells. Genes with similar expression patterns can be clustered together with similar cellular functions. This approach may further understanding of the functions of many genes for which information has not been previously available [8].

The importance of clustering is also clear in application related to biology, social sciences, computer science, medicine, and others. Consequently many clustering methods have already been developed. However classical methods have been usually developed in a centralized fashion, requiring that data be located at a single place. This means that these algorithms cannot be applied in the case of distributed data sets. Another drawback that can be seen in classical clustering methods is that these algorithms need some hints about the target clustering, such as number of clusters, the expected cluster size, or the minimum density of clusters, among others.

In [10] we address this issue with the Bee clustering algorithm. In this work we propose the application of Bee clustering algorithm to datasets of interest for bioinformatics since it can handle distributed data, a typical necessity in scenarios collaborative work in genomics and proteomics where laboratories form networks and the data is distributed.

The simplest method for clustering, the k-means algorithm, needs to know beforehand the number of groups in the data. Each centroid defines a data group and the remaining data is associated to the closest centroid.

Unfortunately, the performance of this algorithm strongly depends on the information we have about the data regarding the possible number of groups,

which poses a problem in applications where this information is not known a priori, or changes dynamically.

Another well-known method is an agglomerative hierarchical clustering algorithm. The algorithm starts with the finest partitioning possible and, in each iteration, merges the two least distant clusters. The distance between two clusters is computed as the average dissimilarity between all possible pairs of data elements within these two clusters. Hierarchical clustering methods are thought to give higher quality solutions than partitioning methods. However, their runtime scales quadratically and results depend heavily on the linkage metric used. Also, the derivation of the appropriate termination criteria can be difficult, if the correct number of clusters is not known.

Our algorithm, Bee clustering, relies on recruitment observed among honey bees. In nature, bees travel far away from the hive to collect nectar. They return to the hive with nectar and information about the nectar source to recruit other bees to that food source. This recruitment is performed by dancing, during which a bee communicates to other bees the directions, distance, and desirability of the food source. The Bee clustering algorithm uses this behavior to create groups of agents.

The rest of the paper is organized as follows. Section 2 presents clustering approaches that are related to ours. In Section 3 the details of the Bee clustering algorithm are given. Section 4 presents and discusses the results achieved, while Section 5 presents the conclusions and outlines future works.

2 Related Work

Several approaches inspired by social insects exist that deal with the clustering problem (see e.g. [7] for a brief review of the literature). Most of them are inspired by two main behaviors observed in ant colonies: ant foraging behavior, and corpse clustering.

Next, we explain these behaviors and give some examples of clustering algorithms inspired by them that are related to the Bee clustering algorithm. We start with ant colony optimization (ACO) based approaches, then present others that are based on cemetery organization, and finally some algorithms for distributed clustering.

ACO is related to ant foraging behavior i.e. ants depositing a chemical pheromone as they move from a food source to their nest, and foragers following such pheromone trails. In the algorithm presented in [11] ants visit data objects one by one and select clusters for data objects by considering pheromone information. Ants use a pheromone matrix which guides other ants towards the optimal clustering solution. After generating a population of R trial solutions, a local search is performed to further improve the fitness of these solutions. The pheromone matrix is then updated according to the quality of solutions produced by the agents.

It is easy to deduce that this and other ACO-based algorithms depend on a global pheromone matrix that is a single point of failure.

Another class of clustering algorithms is inspired by the way real ants clean their nests and organize dead bodies in their colonies. Here, in contrast to ACO, no pheromones are used. Rather, the environment itself provides the stigmergic component. Authors in [9] proposed a basic ant-based data clustering algorithm that associates a position on a toroidal grid with each of the data items to be clustered. The positions of these items, as well as those of the agents moving them around, are initialized randomly. These agents have a sorting behavior based on local rules. The number of moves an agent can perform is defined a priori. The agents try to pick up or drop objects on the two-dimensional board according to a local measure of similarity.

The approach proposed in [13] is based on a hypergraph to combine clustering produced by three colonies. Each ant colony projects randomly data objects onto a plane and the clustering process is done by ants picking up or dropping down objects with different probabilities. The same authors have also developed an extended version where they have added a *centralized* element to compute the clustering: a queen ant agent. This agent receives the results produced by all colonies, calculates a new similarity matrix, and broadcasts to all other colonies. Each colony re-clusters the data using the new information received.

The main problem of these approaches regards the nature of the algorithms' output. They do not generate an explicit partitioning, but a spatial distribution of the data elements. While this may contain clusters that are obvious to a human observer, an evaluation of the clustering performance requires the retrieval of these clusters, and it is not trivial to do this without human interaction.

Apart from techniques inspired by ant foraging and corpse clustering, a number of other swarm-intelligence-based behaviors have been used for clustering.

A probabilistic ant-based clustering algorithm (PACE) for distributed databases was proposed that uses chemical recognition, a behavior in which an ant identifies another group of ants from the same colony using a distinctive odor that is unique to each colony. This is used to identify and form a group of ants that carry related data. Its main characteristic is the formation of numerous zones in various distributed sites based on the user query to a distributed database. An extended version is presented in [6] – the I-PACE – where the authors propose the introduction of weights for individual or groups of data items in each zone according to their relevance to the queries with the concept of familial pheromone trail as part of an ant odor identification model to bias the movements of different types of ants towards the members of their own family. The aim is to reduce the convergence time and to improve the quality of clustering.

These approaches were developed in a distributed way; however they rely on informations that need to be given a priori.

3 Bee clustering approach

3.1 Biological Inspiration of the Algorithm

The Bee clustering algorithm has been inspired by the study of honey bee colonies and the behavior of forager bees. Honey bees collectively select the best

nectar source available using simple behavioral rules. In the process of foraging, bees travel up to 10 km from the hive to collect nectar. They return with nectar and information about the nectar source [4]. Bees have three possible behaviors:

1. to share the nectar source information by dancing, a behavior in which a bee communicates the direction, distance, and desirability of the food source to other bees, trying to recruit new bees to that food source;
2. to continue foraging without recruiting other bees;
3. to abandon the food source and go to the area inside the hive called the dance floor to observe dancing bees and select another food source.

Based on these behaviors, colonies form groups of bees that forage the nectar of best quality. The rate of dancing and abandonment are the basis of Camazine and Sneyd's mathematical model [5], which demonstrates how the properties of the system emerge from the interactions among the agents (bees).

Define the following rates or probabilities:

- PX_A and PX_B: probabilities of abandoning nectar sources A and B respectively, per foraging trip;
- PD_A and PD_B: probabilities of dancing to recruit for sources A and B respectively, where $PD = 1 - PX$;
- PF_A and PF_B: probabilities of following a bee dancing for sources A and B respectively. PF_A is computed as $PF_A = \frac{D_A d_A}{D_A d_A + D_B d_B}$, where d_A and d_B are the proportions of time that the foragers actually dance for source A and B, and D_A and D_B are the number of bees in the source A and B respectively. There is a similar equation to compute PF_B.

3.2 Basics of the Clustering Algorithm

In the Bee clustering algorithm this mathematical model is one of the basis to form groups of agents with similar features. Each bee agent represents an object that needs to be grouped. Thus, the attributes of this object constitute the set of agent's features. Agents have only a limited knowledge: they only know about the agents that are placed in their groups and they cannot remember their past groups, i.e. they have no memory. Despite this, they are able to group together to form clusters.

Figure 1 depicts the scheme of the clustering process. Each agent has a set of possible states $\mathcal{S} = \{v, w, d\}$ that drive their actions. In state $\delta = d$ the agent is dancing to recruit other agents to join its group. State $\delta = v$ means that the agent is visiting the agent which is inviting it, and state $\delta = w$ means that the agent is watching the dancers to randomly choose an agent to visit. Random selections here are based on a uniform distribution because we assume that agents have little or no knowledge about other agents as this would require a high level of communication. At the beginning the state of all agents is v.

The main parameters used by the Bee clustering algorithm are: $\mathcal{D}$ (set of agents); $\mathcal{X}$ (set of agents attributes); Pa (probability of abandoning an agent);

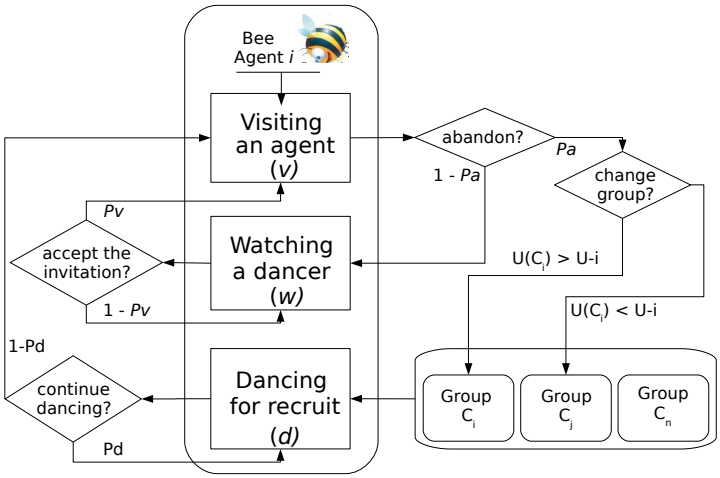

Fig. 1. Scheme of the Bee Clustering Process.

Pv (probability of visiting an agent); Pd (probability of keeping dancing for a group); $U(C)$ (utility of group C); and $maxSteps$.

Algorithm starts by considering each agent belonging to an individual group. Next, they visit other agents and decide whether or not they will change to the group of the agent they are visiting. Notice that visits happen after the dancing process. However at the beginning, when the state of all agents is $\delta = v$, each agent randomly chooses an agent j to visit.

During the clustering process agents need to make some decisions. Diamonds in Figure 1 represent these agent's decisions possibilities that are about wether or not to abandon the agent that it is visiting; to change or not to the group of the visited agent; to continue or not dancing to recruit other agents for their groups; and to visit a dancer or not.

Let us assume that agent i is in state $\delta_i = v$. This means that i needs to decide if it will abandon the agent j it is visiting. To do this i calculates the probability Pa according to Equation 1, where $|\mathcal{X}|$ represents the number of attributes of i and j. If i abandons j, i changes its state to w indicating that the next action is to observe other agents whose state is $\delta = d$.

$$Pa = \sqrt{\sum_{x=1}^{|\mathcal{X}|}(i_x - j_x)^2} \tag{1}$$

When the state of agent i is $\delta_i = w$, this means that i is observing those agents in state d and will then randomly choose an agent j. Next, i decides whether or not to accept the invitation of j. This is done with probability Pv

that is calculated as in Equation 2, where $D(C_j)$ is the number of agents dancing for C_j, and N is the number of agents inside C_j.

$$Pv = \frac{D(C_j)}{N} \tag{2}$$

If the invitation is not accepted, i will choose another agent that is dancing and decide whether or not to accept the invitation. When an invitation is accepted, the state of i changes to $\delta = v$ and the clustering process restarts.

To ground the decisions about to change or not to the group of the visited agent and to continue or not dancing to recruit other agents for their groups we use not only the mathematical model of [5] but also a response threshold model described in [3], and the computation of difference utilities (e.g. as in [12]), a technique grounded on methods of statistical mechanics.

The response threshold model is used mainly to compute the probabilities of continuing dancing. The difference utilities approach is used to let agents compute the utility of a group with and without itself, so that an agent can decide whether or not it changes groups. Next we present these two formalisms.

3.3 Decision about Changing Groups

Another issue is that agents must decide whether they change groups. This is an essential decision for a good clustering result. Because we do the clustering in a distributed way, no one is in charge of maximizing the global utility. Rather, each agent is acting locally. This is a well-known issue: Tumer and Wolpert [12] for instance have shown that there is no general approach to deal with the complex question of collectives.

In [1] agents need to maximize a global utility computed over the difference between the initial and the final clustering. For this to be done in a distributed way, the computation of the global utility involves the agents broadcasting what they believe the final clusterings are. Instead of maximizing the global utility directly, an agent can compute the difference utility $D_i(z) = G(z) - G(z - i)$, where G is the global utility, $G(z)$ is the utility considering the actions of all agents, and $G(z - i)$ is the utility considering the actions of all agents but i. We remark that in [1], agents do broadcasting (while our agents only know about the cluster they belong) and that the authors use this technique to compute an ensemble of clusters, i.e. this is a different problem where the goal is to create a single clustering that best characterizes a set of clusterings, without using the original data points that were used to generate the set of clusterings.

In our clustering algorithm we use the utility difference approach to help agents to make a decision about whether or not to change groups. If the utility of agent i's group is better without i, then i decides to abandon its group and changes to the group of the agent it is visiting. Otherwise it remains in its group. To decide this, i calculates the utility of group C_i with its participation ($U(C_i)$ as defined in Equation 3) and without it, $U - i$.

In Equation 3 $var(C_i)$ is the intracluster variance of C_i, which indicates how close the agents are within the group C_i. The variance needs to be minimized

and it is calculated according to Equation 4, where N is the number of elements inside the group C_i, $d(i, c_i)$ is the Euclidean distance from i to the centroid c_i of group C_i and the centroid c_i is calculated according to $c_i = \frac{1}{N} \sum_{i \in C} i$.

$$U(C_i) = 1 - var(C_i) \tag{3}$$

$$var(C_i) = \sqrt{\frac{1}{N} \sum_{i \in C} d^2(i, c_i)} \tag{4}$$

Next, agent i compares both utilities $U(C_i)$ and $U - i$. If the latter is higher than $U(C_i)$, indicating that the group utility of i is better without its participation, i changes to group C_j of agent j ; otherwise i remains in its group C_i. In both cases i starts to dance to recruit other agents to join its group. Thus, the state of i changes to $\delta_i = d$.

3.4 Decision about Joining a Group

In the Bee clustering algorithm agents visit each other and form groups thanks to invitations made by other agents that are dancing. The time that an agent remains dancing is key. If it is too long, the model might not work because all agents will be dancing at the same time; if agents dance for too short a period, the algorithm converges to a clustering with a lot of small groups. To control this time, agents use the response threshold model, which is inspired by division of labor and task specialization among social insects. This model describes task distribution using the stimulus produced by tasks that need to be performed as well as an individual response threshold associated with each task. An insect that perceives a task stimulus higher than its associated threshold has a higher probability to perform this task.

In our algorithm agents use an associated stimulus and threshold to continue dancing or not. These values are associated with the quality of the agent's group. If the group has a good quality, then the stimulus increases and the threshold decreases. This way the agent has a tendency to continue dancing for its group. If, in contrast, the group has low quality then the agent stimulus decreases and the threshold increases, tending to stop dancing. This tendency can be put as a probability and, according to the response threshold model, it is calculated as in Equation 5, where S_i and θ_i are the stimulus and threshold associated with a given task. In the particular case of computing the probability of stop dancing, we call this probability Pd.

$$Pd = \frac{S_i^2}{S_i^2 + \theta_i^2} \tag{5}$$

The values of the threshold θ_i and of the stimulus S_i are initialized with $S = 1$ and $\theta = 1$ and are updated each time step provided the state is $\delta = d$. This update depends on the utility of the group. If the utility $U(C_i)$ at time t is higher than the utility at $t - 1$, then $S_i = S_i + \alpha$ and $\theta_i = \theta_i - \alpha$, increasing

the tendency to continue dancing. On the other hand if $U(C_i)$ at time t is lower than the utility at $t-1$, then $S_i = S_i - \alpha$ and $\theta_i = \theta_i + \alpha$. Here α is a parameter that must be set.

As mentioned, with probability Pd, i keeps dancing. Otherwise i stops the dance and changes to state $\delta_i = v$, indicating that it will visit agent j again.

4 Experiments and Results

We have performed experiments to investigate the quality of the Bee clustering algorithm using public domain data set as well as some datasets related to bioinformatics. In the present paper we focus on the latter. The dataset used are Leukemia [14] and Yeast [2]. The Leukemia data set contains data on gene expression related to a subtype of leukemia. It is composed by 271 elements that can be divided in 2 different substructures: one with 3 classes and another with 7 classes. Each element of the set has 327 attributes. The Yeast data set is composed by 1484 data elements with 8 attributes. It contains 10 classes predicting the cellular localization sites of proteins. The 10 clusters have the following sizes: 463, 429, 244, 163, 51, 44, 35, 30, 200, and 5.

For comparison we use the well-known k-means and an agglomerative hierarchical clustering algorithm based on the linkage metric of average link.

The value of the parameters taken by our algorithm are: $\alpha = 0.02$, $maxSteps = |\mathcal{D}| * \mu$, and $\mu = 6$. These values were chosen after several tests with different values for each parameter. $|\mathcal{D}|$ and $|\mathcal{X}|$ take the values of the size of the data set and the number of attributes respectively. This way $|\mathcal{D}|$ and $|\mathcal{X}|$ differ for the Leukemia and Yeast data sets.

To assess the performance of the clustering produced we use the Rand Index that determines the degree of similarity between the correct classification, which is known for the Leukemia and Yeast data sets, and the solution generated by a clustering algorithm. It is defined according to Equation 6, where a, b, c, and d are computed for all possible pairs of data points x and y and their respective group assignments $g_Z(x)$, $g_Z(y)$, $g_V(x)$, and $g_V(y)$, where Z is the known correct classification and V is the solution found. R is limited to the interval [0,1] and should be maximized.

$$R = \frac{a+d}{a+b+c+d}, \tag{6}$$

with:

$$\begin{aligned} a &= \{x, y | g_Z(x) = g_Z(y) \wedge g_V(x) = g_V(y)\}, \\ b &= \{x, y | g_Z(x) = g_Z(y) \wedge g_V(x) \neq g_V(y)\}, \\ c &= \{x, y | g_Z(x) \neq g_Z(y) \wedge g_V(x) = g_V(y)\}, \\ d &= \{x, y | g_Z(x) \neq g_Z(y) \wedge g_V(x) \neq g_V(y)\}. \end{aligned}$$

Table 1 shows the Rand Index for the clustering produced by the Bee clustering algorithm, in comparison to k-means and average link on the Leukemia and Yeast data sets. Entries in this table show the average and standard deviation

over 60 repetitions for the the Rand Index, and the number of groups found by each algorithm.

As seen, in the Yeast data set our approach outperforms the other algorithms in terms of the Rand Index, and it comes close to the correct determination of the number of groups. However, in our case, it is not necessary to give a priori informations about the number of clusters as it happens with k-means.

In the Leukemia data set our approach found the structure of 3 classes (average 3.1). Besides, for this case of 3 classes in terms of Rand Index, Bee clustering has outperformed the others. For the another structure of 7 classes our approach was not able to outperform the K-means and average link algorithms, although it comes close. However, it must be noticed that our algorithm does the clustering in a totally distributed way.

Table 1. Rand Index for k-means, Average link, and Bee clustering regarding the Leukemia and Yeast data sets over 60 repetitions.

LEUKEMIA - 3 classes	**k-means**	**Average link**	**Bee clustering**
Rand Index	0.31 (0.31)	0.32 (0)	**0.49**(0.02)
Number of identified groups	5.6	6	**3.1**
LEUKEMIA - 7 classes	**k-means**	**Average link**	**Bee clustering**
Rand Index	**0.75** (0.02)	0.64 (0)	0.51(0.09)
Number of identified groups	**8.4**	16	3.1
YEAST	**k-means**	**Average link**	**Bee clustering**
Rand Index	0.7506(0.001)	0.7426(0)	**0.8172** (0.05)
Number of identified groups	**10**	**10**	10.3

5 Conclusion and Future Work

This paper has shown the application of Bee clustering, a biologically-inspired algorithm to solve the problem of clustering a set of data in a decentralized fashion without any initial information. It uses a mathematical model that is based on: recruitment among honey bees (to attract more agents to a group); a threshold response model (to compute the probability of joining a group); difference utilities (to compute the probability of changing groups).

Bee Clustering was compared with the well-known k-means and the average link algorithms. Results were extremely encouraging, especially considering that the algorithm is distributed, and therefore we plan to use it in other datasets, including some that are related to experiments on microarrays, where the objective is to find correlations among several genes that are expressed.

Besides, we intend to investigate the possibility of performing the clustering process considering the optimization of multiple objectives. To address this, we plan to let agents compute several utilities (each according to one objective) to decide whether or not to change groups.

References

1. A. Agogino and K. Tumer. Efficient agent-based cluster ensembles. In P. Stone and G. Weiss, editors, *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, AAMAS '06*, pages 1079–1086, New York, NY, USA, 2006. ACM.
2. A. Asuncion and D. J. Newman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2007. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
3. E. Bonabeau, G. Theraulaz, and M. Dorigo. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, USA, 1999.
4. S. Camazine, J. D. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, Princeton, N.J., 2003.
5. S. Camazine and J. Sneyd. A model of collective nectar source selection by honey bees: Self-organization through simple rules. *Journal of Theoretical Biology*, 149(4):547–571, April 1991.
6. R. Chandrasekar and T. Srinivasan. 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, 2007.
7. J. Handl and B. Meyer. Ant-based and swarm-based clustering. *Swarm Intelligence*, 1(2):95–113, December 2007.
8. D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 16:1370–1386, 2004.
9. E. D. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants. In *Proceedings of the third international conference on Simulation of adaptive behavior: from animals to animats*, pages 501–508, Cambridge, MA, USA, 1994. MIT Press.
10. D. S. d. Santos and A. L. C. Bazzan. A biologically-inspired distributed clustering algorithm. In *Proc. of the 2009 IEEE Swarm Intelligence Symposium*, pages 160–167, Nashville, 2009. IEEE.
11. P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni. An ant colony approach for clustering. *Analytica Chimica Acta*, 509(2):187–195, 2004.
12. K. Tumer and D. Wolpert. A survey of collectives. In K. Tumer and D. Wolpert, editors, *Collectives and the Design of Complex Systems*, pages 1–42. Springer, 2004.
13. Y. Yang and M. Kamel. Clustering ensemble using swarm intelligence. In *Proceedings of the Swarm Intelligence Symposium (SIS 03)*, pages 65–71, Indianapolis, USA, April 2003.
14. E.-J. Yeoh, M. E. Ross, S. A. Shurtleff, W. K. Williams, D. Patel, R. Mahfouz, F. G. Behm, S. C. Raimondi, M. V. Relling, A. Patel, C. Cheng, D. Campana, D. Wilkins, X. Zhou, J. Li, H. Liu, C.-H. Pui, W. E. Evans, C. Naeve, L. Wong, and J. R. Downing. Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. *Cancer Cell*, 1(2):133–143, 2002.