The effect of using evolutionary algorithms on Ant Clustering Techniques.

Claus Aranha[1] and Hitoshi Iba[2]

[1] The University of Tokyo `caranha@iba.k.u-tokyo.ac.jp`
[2] The University of Tokyo `iba@iba.k.u-tokyo.ac.jp`

Abstract. Ant-based clustering is a biologically inspired data clustering technique. In this technique, multiple agents carry the information to be clustered, and make local comparisons. In this work we use genetic algorithms to improve the implementation and use of ant-clustering techniques.

1 Introduction

Ant colony based heuristics are currently a widely studied field of computer science. The first works were on TSP problems [6], but now they are used on many different fields, like routing algorithms and building philogenetic trees [3, 1].

An ant colony has many characteristics that are considered useful. It is composed of many agents which, although simple individually, can perform rather complex tasks as a group, without central coordination. Some examples are building an ant nest, brood pits and cemeteries, hunting and foraging food [5, 2]. The coordination of an ant colony is of local nature, composed mainly of indirect communication through pheromones (known as stigmergy), although direct interaction communication from ant to ant, in the form of antennation, and direct communication have also been observed [17].

The high number of individuals and the decentralized approach to task coordination means that ant colonies show high degrees of parallelism, self-organization and fault tolerance. All of which are desired characteristics in modern computer systems. Such ant systems can be easily implemented as cellular automata.

Clustering is the classification of similar objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset share some common trait. Clustering is used as a data processing technique in many different areas of application, such as bioinformatics, data mining, image analysis, etc [14, 12].

In nature, the *Messor Sancta* species of real ants have been observed to perform this clustering behavior, by grouping 1500 ant corpses into large piles [7] over a period of 26 hours. Besides this patchwork sorting, the *Leptothorax* ant has been observed to cluster its brood into concentric annular structures, with eggs and micro-larvae in the center, and larger offsprings located in the outer side of the structure [11].

Experiments like those serve as the inspiration for using an ant-based heuristic approach to the clustering problem. Recent developments in the ant-based clustering technique show that it is competitive when clustering datasets with high dimensionality, or databases where the number of clusters is not previously known.

However, there is a large number of parameters in the ant-clustering algorithm, which exact effects in the performance of the technique are not yet very well known. Small differences in the constants might result in large differences in the results, so that hand-tuning is a daunting task. We propose the use of Genetic algorithms to automatically choose the best parameters for a given task.

In the next session, we explain the Ant Clustering algorithm, defining its key components, then talk about the main branches in its field of research, and its perceived characteristics.

In section three, we describe our implementation of the LF algorithm, along with the modifications we made, and then describe the application of the genetic algorithm to the parameter optimization.

In section four, we present the experiments we did, and discuss the achieved results, and in section five we draw some conclusions from our work and future directions.

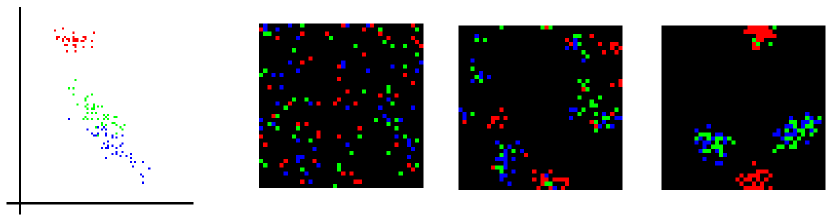

Fig. 1. Example of ant clustering algorithm. The first image to the left is the data distribution on feature space (2 dimensions out of 4 total. The data is the IRIS set of the ELENA clustering benchmarks). The next image is the random distribution of that data on the workspace (data belonging to different classes are colored differently). The last two images show later stages of the clustering process.

2 Ant Clustering

Ant-based clustering algorithms can be considered *non-hierarchical, hard, agglomerative* clustering methods. Non-hierarchical means that there is no parent-child relationship between the objects or the clusters formed by the technique.

Hard means that each object is assigned to only one cluster. Agglomerative means that the clusters are formed bottom-up - in other words, isolated objects are progressively put together to form bigger clusters.

While the first proposal of a clustering algorithm inspired by ant colonies was made by Deneubourg [5], the canonical ant clustering algorithm follows the formula described by Lumer and Fayette [13]. The basic idea of the ant cluster algorithm is described as a cellular automata like field, where artificial ants can pick and drop non-deterministically objects which represent the data objects to be clustered. As the ants pick isolated objects, and drop them near similar objects. This 'vanilla' clustering algorithm, which we'll call the LF algorithm, can be described as follows.

One of main characteristic of ant clustering, is that the clustering operation happens on a toroidal bi-dimensional grid unrelated to the n-dimensional space which describes the properties of data relative to each other.

From the n-dimensional feature space we can obtain information like the disparity between two different pieces of data. However, from the bi-dimensional grid in the ant algorithm we cannot get such information directly. We will call this bi-dimensional grid the *workspace*, and the n-dimensional space the *feature space*. The workspace can be imagined as a 2 dimensional random projection of the feature space. By randomly projecting the objects into this 2 dimensional space, and performing local comparisons, we have a constant cost for clustering objects of arbitrary dimensionality.

The first step in the ant-clustering system is to distribute the data (objects) randomly into the workspace. Each object is projected onto one grid of the workspace (see the second image from the left in figure 1). Then a number n_a of ants is put on random positions of the workspace. Only one ant and only one object can be put into one grid of the work space. Each ant is also able to carry one data object with itself.

At each time step, each ant will, if loaded, try to unload its object onto its current position or, if unloaded, try to pick an object in the same grid as itself. The probability of picking or dropping an object is based on the disparity (or distance in feature space) between that object and other objects in its neighborhood. In other words, if we have a function $d(i)$ which increases as the disparity between an object i and its neighbors decreases, we can describe the pick and drop probabilities as follows:

$$P_{pick} = \left(\frac{k_p}{k_p + f(i)} \right)^2 \tag{1}$$

and:

$$P_{drop} = \left\{ \frac{f(i)}{k_d + f(i)} \right)^2 \tag{2}$$

Equation 1 gives the probability of picking up an object. k_p is a pick up threshold parameter. Equation 2 gives the probability of dropping an object. k_d is a drop threshold parameter. According to these equations, the probability of dropping an object in higher when near similar objects, and the probability of picking and object is higher when that object is isolated, or near dissimilar objects.

After picking or dropping the object, the ant will move, following some defined move policy (for instance, moving one step in a random direction). Since the workspace is toroidal, ants walking off a border will surface on the opposite border.

By following these rules, objects that are near each other in the feature space will be likely to be dropped in neighboring positions in the work space. After an initial period of random activity, a small tentative cluster of few similar objects will form. This pre-cluster acts as a stigmergic beacon so that the probability of dropping new, similar objects near it is greater than anywhere else on the workspace. This leads to a positive feedback cycle which increases the size of the cluster, until the clustering process is complete. This progress is illustrated in figure 1.

2.1 Main Works

Ramos et al. [15]. applied ant-based clustering to the classification of stone images. In their works, they noticed that the normal LF algorithm would generate a large quantity of small clusters, and that many actions were wasted when the ants moved through empty space. To address this concerns they used pheromones to guide the ant movement.

Also, they studied the performance of the algorithm on continuous clustering (i.e. when new data is added over time during the execution of the algorithm) [16], and showed that this improves clustering performance for the ant clustering system.

Handl et al. [9] changed the ants' movement policy so that the ants, after dropping an object, would "teleport" to the next isolated object, and pick it automatically. In this way, an ant would never give a step while not carrying an object, which did not add anything to the clustering effort. They also added limited local memory to each ant, which would give them "hints" to the best place to drop the carried object.

Hartmann [11] proposes the use of Neural Networks to replace the pick and drop functions. Among other works dealing with improvements to the ant-clustering algorithm we find [4, 18].

3 Implementation

To enable us to study the possibility of optimizing Ant Clustering with genetic algorithm, we implement the basic LF algorithm with some of the improvements suggested in the literature, as described below.

3.1 Details of the Implementation

The system works over data objects that can be compared by the use of Euclidean distance:

$$d(a, b) = ((a_0 - b_0)^2 + (a_1 - b_1)^2 + ... + (a_n - b_n)^2) \tag{3}$$

Where $a_k, b_k (k \in 1..n)$ are the values of the k-th feature of the n-dimensional data objects a, b. With this similarity function $d(a, b)$ we can define the neighborhood disparity function $f(i)$ for a data object i:

$$f(i) = \frac{\sum M_d - d(i, x_s)}{S_t} \tag{4}$$

Where x_s is an object within the neighborhood radius of i, M_d is the maximum distance between any two objects, and S_t is the total number of objects in the neighborhood of i. The neighborhood of an object is given by all the objects within Manhattan distance *sight* of the object (where *sight* is a configurable parameter).

We then calculate the crowdedness factor, $c(i)$ by equation below:

$$c(i) = \frac{S_t^2}{S_t^2 + k_{crowd}^2} \tag{5}$$

$1 < k_{crowd} < (2 * sight)^2$ is an integer constant which tunes the behavior of $c(i)$. We multiply $f(i) * c(i)$ when dropping an object, and $f(i) * (1 - c(i))$ when picking an object, to encourage the ants to pick isolated objects, and drop them near early clustering of objects. This result is then used as the disparity value for the pick and drop probabilities given by 1 and 2.

The behavior of an ant during the clustering follows this behavior: If the ant is not carrying any object, it tries to pick the available (not being carried) objects, one at a time with probability given by eq. 1. Once it decides to pick one object, the ant is moved into that position. It starts, then, a random walk over the workspace, looking for a place with high local similarity to drop the object. At each time step, the ant walk one random step to a neighborhood cell, and try to drop its load once. If the ant tries to move to a cell where there is another ant present, it stays in the same place.

3.2 Use of Genetic Algorithms

To improve the ant clustering algorithm, we'll try to optimize its parameters (presented in table 1) by using Genetic algorithms. In [18], it is commented that the sensitivity of the many parameters in ant-clustering is a topic worthy of study, in order to improve the system. [4] also says that the high number of parameters is an encumbrance to the system.

In our evolutionary framework, each individual is represented by the set of configuration parameters in table 1. For each generation, we run the program once with each set of parameters, and take the fitness from each run.

Table 1. Parameters for the ant-clustering

Name	Value
Pick constant(K_pick)	(0..1.0)
Drop constant(K_drop)	(0..1.0)
Crowd constant(K_crowd	(1..$(2 * sight)^2$)
Sight range ($sight$)	1..10
Workspace size ($wsize$)	(integer)
Number of ants ($nants$)	(integer)

We use the Elite selection strategy for GA, where, for each generation, the best *elitesize* individuals are directly copied into the next generation, and the remaining individuals of the population are deleted and replaced by crossover between this elite.

For the crossover operator, we randomly choose two parents from the elite, and create a new individual by choosing one parameter value from each parent (equal probability for both parents). After that we run the mutation operator (with a probability equal to the mutation parameter for each individual). The mutation operator can either change the value of one parameter by 10%, or generate a new random value for that parameter.

The key in a successful application of GA to a problem is an appropriate choice of the fitness function. One of the strong points of ant clustering is the ability to auto-detect the number of clusters.

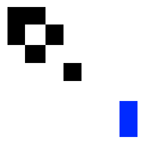

Fig. 2. Two clusters, according to the fitness definition

To extract the clusters from the workspace, we define a cluster as a group of objects within 2 units of Manhattan distance from any member of the group. In this way, in figure 2, we can see 2 such different clusters.

However, the number of clusters alone does not tell us how good the clustering is, so we must also account for the quality of the clusters. We *Average Local Linkage*, to measure the quality of one cluster. First, we take the neighborhood disparity function ($f(i)$ described in equation 4 to determine the local Linkage of one object. From this value, we calculate the ALL for the cluster as:

$$ALL(C) = \frac{\sum f(c_i)}{C_{size}} \quad (6)$$

Where C_{size} is the number of objects in the cluster, and each $c_i \in C$ is an object belonging to the cluster.

To calculate the fitness of one individual, then, we identify the clusters by using the definition in figure 2, and then averaging $ALL(C)$ for all clusters where $C_{size} > 1$.

There is, however, one extra thing that must be taken care with when calculating the fitness of ant clustering algorithms. As reported in [18], LF does not reach an stable configuration - since the pick and drop probabilities are not deterministic, the ants may pick some pieces from stabilished clusters, lowering the fitness, just to put them back a few turns later. Therefore, if we just pick any one time step, and measure the fitness at that moment, we can get a lucky high or low unstable state (see figure 3).

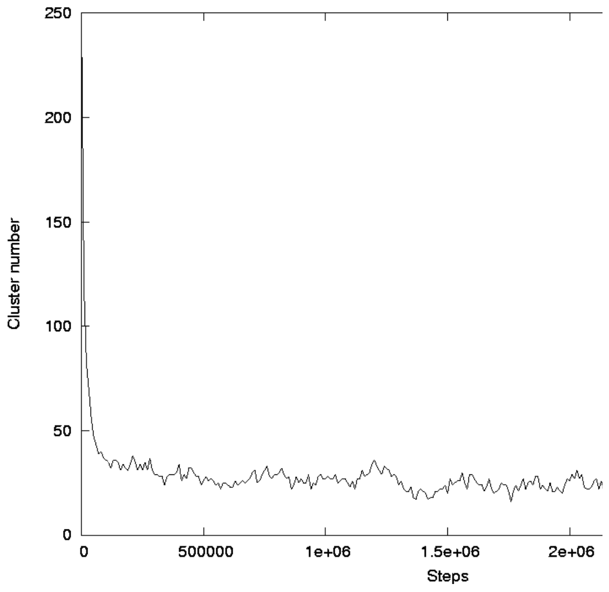

Fig. 3. Number of clusters according to the progression of the algorithm for an individual. Notice that the number of clusters stabilizes around 200.000 steps, but the actual number still varies after that.

In order to avoid that, after a given turn t, we start measuring the fitness for the next t_{fit} turns, and take the average fitness of this period as the individual's fitness.

4 Experiments

To test the influence of evolving the control parameters of the algorithm, we tested it against a simple database of four clusters, linearly separated, each with 250 objects, each object with 2 variable values (figure 4). We wanted to use this simple dataset as a preeliminar study of GA on the LF algorithm.

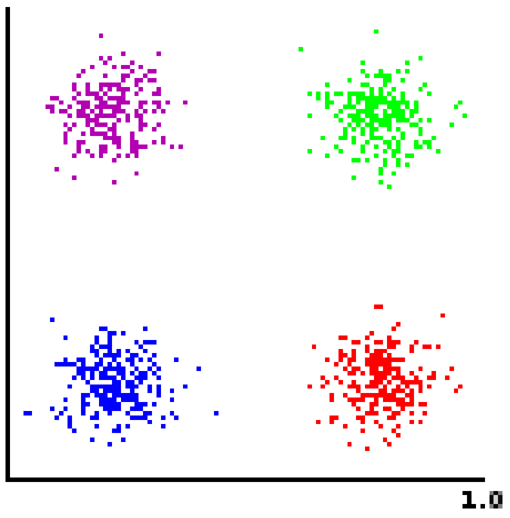

Fig. 4. Position of the data objects in feature space, forming 4 clusters

To tune the parameters, we used a genetic algorithm systems as described in the previous session. The parameters of the genetic algorithm are shown in table 2.

Table 2. Parameters for Genetic algorithm

Name	Value
Population	40
Generations	100
Elite Size	10
Mutation rate	1%

Running the experiment, we found out that in fact the evolved solutions could generate a smaller number of clusters with the passing of the generations. The number of clusters generated by the best individual of the evolved population compares favorably (figure 6) against the the fitness reached by running the algorithm with the default parameters found in the literature (pick rate 0.2, drop rate 0.5).

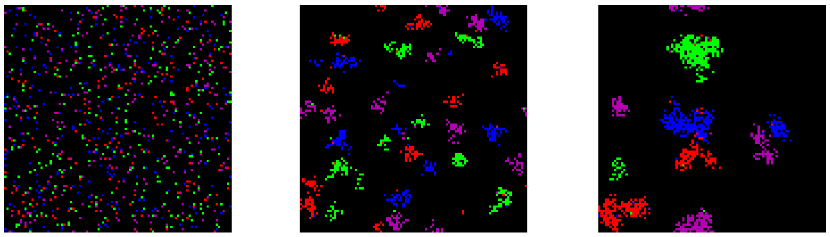

Fig. 5. Progress of the evolved solution on the 4 cluster problem at steps 0, 30.000 and 300.000

Furthermore, the population showed a strange behavior of *losing* average fitness, while *improving* the best fitness. We tried to run the tests with different fitness measures, finding similar results. This effect might be due to a faulty evolutionary strategy (figure 7).

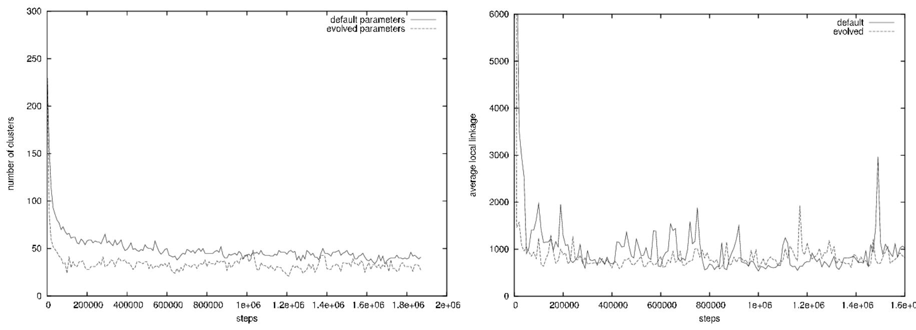

Fig. 6. Performance comparison between the default parameters, and the GA optimized ones. The graph to the left compares the number of clusters, and the graph to the right compares the average local linkage.

5 Conclusions

The GA was able to find a set of parameters with a somewhat better performance when compared to hand-tuned parameters found in the literature. The

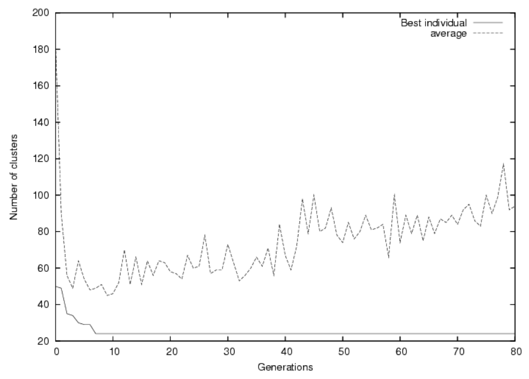

Fig. 7. Population performance based on the number of clusters. While the best individual increased his performance quickly, the population average strangely went up

average fitness of the population as a whole was increasing, while the fitness of the best individual was decreasing, which indicates that there might be other particularities for ant clustering, besides those pointed out when describing the fitness function, that need to be tended to.

Further improvement could be reached by having the parameters of the algorithm evolve during the execution of the clustering. We have had some success by dividing the clustering in two stages, one with loose k_{pick} and *sight* values, which will loosely group similar objects together, and the other with normal values to form the tight cluster. This is our current focus.

References

1. Shin Ando and Hitoshi Iba. Ant algorithm for construction of evolutionary tree. In *Proceedings of GECCO 02*. IEEE, 2002.
2. S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneydd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, 2001.
3. G. Di Caro and M. Dorigo. Antnet: Distributed stigmergic control for communication networks. *Journal of Artificial Intelligence*, Vol. 9, pp. 317–365, 1998.
4. Ling Chen, Xiao hua Xu, and Yi-Xin Chen. An adaptive ant colony clustering algorithm. In *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, pp. 1387–1392, Shangai, China, August 2004. IEEE.
5. J.L. Deneubourg, S. Gross, N. R. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting: Robot-like ants and ant-like robots. *Simulation of Adaptative Behavior: From Animals to Animats*, pp. 356–363, 1991.
6. M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992. in Italian.
7. Bonabeau E., M. Dorigo, and G. Tharaulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.
8. J. Handl, J. Knowles, and M. Dorigo. On the performance of ant-based clustering. *Frontiers in Artificial Intelligence and Applications*, Vol. 104, pp. 204–213, 2003.

9. J. Handl, J. Knowles, and M. Dorigo. Strategies for the increased robustness of ant-based clustering. *Lecture Notes in Computer Science*, Vol. 2977, pp. 90–104, 2004.
10. Julia Handl, Joshua Knowles, and Marco Dorigo. Ant-based clustering and topographic mapping. Technical Report 2004-009, IRIDIA, Belgium, May 2004.
11. Vegard Hartmann. Evolving agent swarms for clustering and sorting. In *Genetic Evolutionary Computation Conference, GECCO*, Vol. 1, pp. 217–224. ACM press, 2005.
12. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, Vol. 31, No. 3, pp. 264–323, 1999.
13. E.D. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants. In *Proc. of the Third International Conference on The Simulation of Adaptative Behavior: From Animals to Animats 3*, pp. 449–508. Mit Press, 1994.
14. John Quackenbush. Computational analysis of microarray data. *Nature Reviews*, Vol. 2, , June 2001.
15. V. Ramos, F. Muge, and P. Pina. Self-organized data and image retrieval as a consequence of inter-dynamic synergistic relationships in artificial ant colonies. In Ajuth Abraham Javier Ruiz-del Solar and Mario Koppen, editors, *2nd Inl. Conf. On Hybrid Intelligent Systems*, Vol. 87, pp. 500–509, Santiago, Chile, Dec 2002. IOS Press.
16. Vitorino Ramos and Ajith Abraham. Swarms on continuous data. In *CEC'03 Congress on Evolutionary Computation*, pp. 1370–1375. IEEE Press, December 2003.
17. Vito Trianni, Thomas H. Labella, and Marco Dorigo. Evolution of direct communication for a swarm bot performing hole avoidance. In *Ant Colony Optimization and Swarm Intelligence*. Springer, 2004.
18. A. Vizine, L.N. de Castro, E.R. Hruschka, and R.R. Gudwin. Towards improving clustering ants: An adaptative clustering algorithm. *Informatica Journal*, Vol. 29, , 2005.