AntClust: Ant Clustering and Web Usage Mining

Nicolas Labroche, Nicolas Monmarché, and Gilles Venturini

Laboratoire d'Informatique de l'Université de Tours,
École Polytechnique de l'Université de Tours-Département Informatique,
64, avenue Jean Portalis 37200 Tours, France
{labroche,monmarche,venturini}@univ-tours.fr
http://www.antsearch.univ-tours.fr/

Abstract. In this paper, we propose a new ant-based clustering algorithm called AntClust. It is inspired from the chemical recognition system of ants. In this system, the continuous interactions between the nestmates generate a "Gestalt" colonial odor. Similarly, our clustering algorithm associates an object of the data set to the odor of an ant and then simulates meetings between ants. At the end, artificial ants that share a similar odor are grouped in the same nest, which provides the expected partition. We compare AntClust to the K-Means method and to the AntClass algorithm. We present new results on artificial and real data sets. We show that AntClust performs well and can extract meaningful knowledge from real Web sessions.

1 Introduction

Numbers of computer scientists have proposed novel and successful approaches for solving problems by reproducing biological behaviors. For instance, genetic algorithms have been used in many research fields, such as clustering problems [1],[2] and optimization [3]. Other examples can be found in the modeling of collective behaviors of ants as in the well-known algorithmic approach Ant Colony Optimization (ACO)([4]) in which pheromone trails are used. Similarly, ant-based clustering algorithms have been proposed ([5], [6], [7]). In these studies, researchers have modeled real ants abilities to sort their brood. Artificial ants may carry one or more objects and may drop them according to given probabilities. These agents do not communicate directly with each other's, but they may influence themselves through the configuration of objects on the floor. Thus, after a while, these artificial ants are able to construct groups of similar objects, a problem which is known as data clustering. We focus in this paper on another important collective behavior of the real ants, namely the construction of a colonial odor and its use to determine the ant nest membership.

Introduced in [8], the AntClust algorithm reproduces the main principles of this recognition system. It is able to find automatically a good partition over artificial and real data sets. Furthermore, it does not need the number of expected clusters to converge. It can also be easily adapted to any type of data

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 25–36, 2003.

(from numerical vectors to character strings and multimedia), since a distance measure can be defined between the vectors of attributes that describe each object of the data set.

In this paper, we propose a new version of AntClust that does not need to be parameterized to produce the final partition. The paper is organized as follows: the section 2 gives a detailed description of the AntClust algorithm. The section 3 presents the experiments that have been conducted to set the parameters of AntClust regardless of the data sets. The section 4 compares the results of AntClust to those of the K-Means method (initialized with the expected number of clusters) and those of AntClass, an ant-based clustering algorithm. In the section 5, we present some of the clustering algorithms already used in the Web mining context and our very first results when we apply AntClust to real Web sessions. The last section concludes and discusses future evolutions of AntClust.

2 The AntClust Algorithm

The goal of AntClust is to solve the unsupervised clustering problem. It finds a partition, as close as possible to the natural partition of the data set, without any assumption concerning the definition of the objects or the number of expected clusters. The originality of AntClust is to model the chemical recognition system of ants to solve this problem. Real ants solve a similar problem in their every day life, when the individuals that wear the same cuticular odor gather in the same nest. AntClust associates an object of the data set to the genome of an artificial ant. Then, it simulates meetings between artificial ants to exchange their odor. We present hereafter the main principles of the chemical recognition system of ants. Then, we describe the representation and the coding of the parameters of an artificial ant and also the behavioral rules that allow the method to converge.

2.1 Principles of the Chemical Recognition System of Ants

AntClust is inspired from the chemical recognition system of ants. In this biological system, each ant possesses its own odor called label that is spread over its cuticle (its "skin"). The label is partially determined by the genome of the ant and by the substances extracted from its environment (mainly the nest materials and the food). When they meet other individuals, ants compare the perceived label to their template that they learned during their youth. This template is then updated during all their life by the mean of trophallaxies, allo-grooming and social contacts. The continuous chemical exchanges between the nestmates lead to the establishment of a colonial odor that is shared and recognized by every nestmates, according to the "Gestalt theory" [9,10].

2.2 The Artificial Ants Model

An artificial ant can be considered as a set of parameters that evolve according to behavioral rules. These rules reproduce the main principles of the recognition system and apply when two ants meet.

For one ant i, we define the parameters and properties listed hereafter.

The label $Label_i$ indicates the belonging nest of the ant and is simply coded by a number. At the beginning of the algorithm, the ant does not belong to a nest, so $Label_i = 0$. The label evolves until the ant finds the nest that best corresponds to its genome.

The genome $Genome_i$ corresponds to an object of the data set. It is not modified during the algorithm. When they meet, ants compare their genome to evaluate their similarity.

The template $Template_i$ or T_i is an acceptance threshold that is coded by a real value between 0 and 1. It is learned during an initialization period, similar to the ontogenesis period of the real ants, in which each artificial ant i meets other ants, and each time evaluates the similarity between their genomes. The resulting acceptance threshold T_i is a function of the maximal $Max(Sim(i,\cdot))$ and mean $\overline{Sim}(i,\cdot)$ similarities observed during this period. T_i is dynamic and is updated after each meeting realized by the ant i, as the similarities observed may have changed. The following equation shows how this threshold is learned and then updated:

$$T_i \leftarrow \frac{\overline{Sim}(i,\cdot) + Max(Sim(i,\cdot))}{2} \tag{1}$$

Once artificial ants have learned their template, they use it during their meetings to decide if they should accept the encountered ants. We define the acceptance mechanism between two ants i and j as a symmetric relation $A(i,j)$ in which the genomes similarity is compared to both templates as follows:

$$A(i,j) \Leftrightarrow (Sim(i,j) > T_i) \wedge (Sim(i,j) > T_j) \tag{2}$$

We state that there is "positive meeting" when there is acceptance between ants.

The estimator M_i indicates the proportion of meetings with nestmates. This estimator is set to 0 at the beginning of the algorithm. It is increased each time the ant i meets another ant with the same label (a nestmate) and decreased in the opposite case. M_i enables each ant to estimate the size of its nest.

The estimator M_i^+ reflects the proportion of positive meetings with nestmates of the ant i. In fact, this estimator measures how well accepted is the ant i in its own nest. It is roughly similar to M_i but add the "acceptance notion". It is increased when ant i meets and accepts a nestmate and decreased when there is no acceptance with the encountered nestmate.

The age A_i is set to 0 and is increased each time the ant i meets another ant. It is used to update the maximal and mean similarities values and thus the value of the acceptance threshold of the ant $Template_i$.

At each iteration, AntClust randomly selects two ants, simulates meetings between them and applies a set of behavioral rules that enable the proper convergence of the method.

The 1^{st} rule applies when two ants whith no nest meet and accept each other. In this case, a new nest is created. This rule initiates the gathering of similar ants in the very first clusters. These clusters "seeds" are then used to generate the final clusters according to the other rules.

The 2^{nd} rule applies when an ant with no nest meets and accepts an ant that already belongs to a nest. In this case, the ant that is alone joins the other in its nest. This rule enlarges the existing clusters by adding similar ants.

The 3^{rd} rule increments the estimators M and M^+ in case of acceptance between two ants that belong to the same nest. Each ant, as it meets a nestmate and tolerates it, imagines that its nest is bigger and, as there is acceptance, feels more integrated in its nest.

The 4^{th} rule applies when two nestmates meet and do not accept each other. In this case, the worst integrated ant is ejected from the nest. That rule permits to remove non-optimally clustered ants to change their nest and try to find a more appropriate one.

The 5^{th} rule applies when two ants that belong to a distinct nest meet and accept each other. This rule is very important because it allows the gathering of similar clusters, the small one being progressively absorbed by the big one.

The AntClust algorithm can be summarized as follows:

Algorithm 1: AntClust main algorithm

AntClust()

(1) Initialization of the ants:
(2) $\forall$ ants $i \in [1, N]$
(3) $Genome_i \leftarrow i^{th}$ object of the data set
(4) $Label_i \leftarrow 0$
(5) $Template_i$ is learned during N_{App} iterations
(6) $M_i \leftarrow 0$, $M_i^+ \leftarrow 0$, $A_i \leftarrow 0$
(7) $Nb_{Iter} \leftarrow 75 * N$
(8) Simulate Nb_{Iter} meetings between two randomly chosen ants
(9) Delete the nests that are not interesting with a probability P_{del}
(10) Re-assign each ant that has no more nest to the nest of the most similar ant.

3 AntClust Parameters Settings

It has been shown in [8] that the quality of the convergence of AntClust mainly depends on three major parameters, namely the number of iterations fixed to learn the template N_{App}, the number of iterations of the meeting step Nb_{Iter} and finally, the method that is used to filter the nests. We describe hereafter how we can fix the value of these parameters regardless of the structure of the data sets. First, we present our measure of the performance of the algorithm and the data sets used for evaluation.

3.1 Performance Measure

To express the performance of the method we define C_s as $1-C_e$, where C_e is the clustering error. We choose an error measure adapted from the measure developed by Fowlkes and Mallows as used in [11]. The measure evaluates the differences between two partitions by comparing each pair of objects and by verifying each time if they are clustered similarly or not. Let P_i be the expected partition and P_a the output partition of AntClust. The clustering success $C_s(P_i, Pa)$ can be defined as follows:

$$C_s(P_i, P_a) = 1 - \frac{2}{N(N-1)} \times \sum_{(m,n)\in[1,,N]^2, m<n} \epsilon_{mn} \tag{3}$$

where:

$$\epsilon_{mn} = \begin{cases} 0 \text{ if } (P_i(m) = P_i(n) \wedge P_a(m) = P_a(n)) \vee \\ (P_i(m) \neq P_i(n) \wedge P_a(m) \neq P_a(n)) \\ 1 \text{ otherwise} \end{cases} \tag{4}$$

with N the number of objects in the original data set. $P_i(o_b)$ (resp. $P_a(o_b)$) is the cluster number of the object o_b in the partition P_i (resp. P_a).

We use artificial data sets named $Art_{1,2,3,4,5,6}$ for our evaluations. We generate them according to gaussian or uniform laws with distinct difficulties (irrelevant attributes, clusters overlap).

3.2 How Many Meetings?

For the time being, the user has to specify the number of iterations during which artificial ants meet, if she wants to adapt the algorithm to the size of the data set that is explored. AntClust has a default value that is set to 50000 iterations. Nevertheless, in the Web mining context, it is usual to work with larger data sets. In this case, AntClust cannot guarantee a proper convergence since some ants are assigned to a nest without having realized any meeting. Our idea is to consider that the number of iterations may be linearly linked to the number of ants (i.e. the number of objects in the data set). In the method, at each iteration, 2 ants are randomly and uniformly selected in the population. Let α denote the minimal number of iterations each ant has to perform to ensure the convergence of the algorithm. The following equation shows how we associate the total number of iterations Nb_{Iter} to the number of ants N:

$$Nb_{Iter} = \frac{1}{2} * \alpha * N \tag{5}$$

We evaluate the performance of the method for several values of α between 10 and 500 meetings per ant. The goal is to verify that α can be initialized regardless of the size of the data set. The figure 1 shows the results in term of mean clustering success. We have conducted 10 tests for each value of α and each artificial data set.

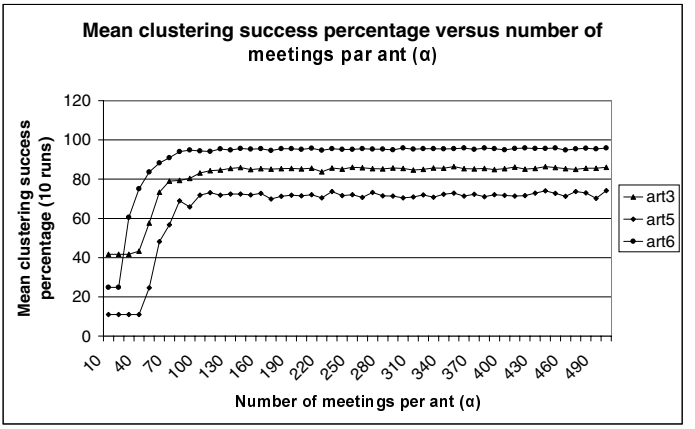

Fig. 1. Mean clustering success over 10 runs for each value of $\alpha \in [10, 500]$.

According to this figure, we can see that the values of the mean clustering success converge quite quickly regardless of the data set. Although the data sets size ranges from 200 to 1100 objects, the convergence seems to operate for the same value of α in every case. Thus, we may consider that there is a minimal number of meetings that each ant should realize to ensure the proper convergence of the method. We consider that we can set experimentally the value of α to 150.

As a consequence, the meetings step of AntClust can be solved in linear time with the number of objects in the data set.

3.3 How Many Iterations to Learn the Template?

We now focus on the number of iterations needed to learn the template. We think that the template learning process cannot be longer, in term of number of iterations, than the meetings step studied before. Let β be the number of meetings needed to learn the template per ant. As for the number of iterations of the meeting step, we test several values of β expressed as a percentage of the value of α ranging from 0% to 100%. The figure 2 presents the results that we obtained for each artificial data sets in term of mean clustering success according to the value of β.

As with the previous experiment, the results of mean clustering success converges even if the limit is not as clearly expressed in the plotted graph. It is important to notice that β is not necessarily positively linked to the performance. For example, the mean clustering success of Art_6 remains stable as β increases, whereas the performances of Art_4 decrease. This can be explained by the fact that the number of clusters is taken into account to compute C_s and that as β increases the error in the estimated number of clusters also increases

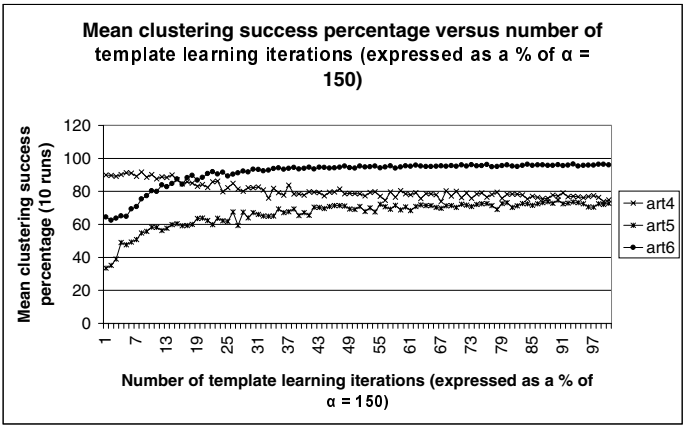

Fig. 2. Mean clustering success over 10 runs for each $\beta \in [1, 100]\%$ of α.

for Art_4. In fact, by increasing β, artificial ants become too sensitive and find too much clusters. As a conclusion, we set β as $0.5 * \alpha$, that is to say, $N_{App} \leftarrow 75$.

3.4 The Nest Deletion Method

At the end of AntClust, the nests that are not sufficiently "interesting" are deleted and the ants are reassigned to the nest of the most similar ant. This method allows suppressing noise in the final partition. For the time being, the nest deletion criterion is only based on the number of ants that belong to this nest and a threshold fixed by the user. The default value of this threshold is equal to a percentage of the total number of ants (generally 15%). This approach, although efficient, is limited, because the algorithm could not find more than a fixed number of clusters of the same size. Our optimization replaces this deterministic method by a probabilistic one that is more adaptive. We compute for each nest η a probability P_{del} to be deleted. This probability depends mainly on the mean integration of the ants in the nest η, $\overline{M_\eta^+}$, and the number of ants in the nest N_η as in the previous version. For the nest η, $P_{del}(\eta)$ is given as follows:

$$P_{del}(\eta) \leftarrow (1 - \nu) * \overline{M_\eta^+} + \nu * \frac{N_\eta}{N} \tag{6}$$

Several experimentations have been conducted and revealed that the value of $\nu = 0.2$ provides the best partitions for the set of data we tested. Hence, this method is more interesting than the previous one, because it allows to better appreciate the number of clusters thanks to its probabilistic nature.

4 Experiments and Results

In this section, we compare AntClust to the K-Means method and the AntClass algorithm to evaluate the performance of our method. First, we will briefly describe both methods. Second, we introduce the main properties of the data sets that are used for the comparison. Then, we present our results over artificial and real data sets.

4.1 K-Means and AntClass

We use two clustering algorithms to evaluate AntClust. First, we choose to apply a traditional K-Means approach because, it can perform well in a short time. This method needs an initial K-partition of the data set that is refined gradually. At each iteration, the objects are assigned to the most similar center of the clusters. The algorithm stops when the intra class inertia becomes stable. In our test, we generate randomly the initial partitions with the number of clusters that is expected in the data set, to get the best results that a K-Means approach can give.

Second, we compare AntClust to AntClass, an other ant-based clustering algorithm. This is an hybrid algorithm in which artificial ants create a first partition of the data set that is used by a K-Means algorithm as initial partition. This scheme is repeated twice in order to get the final partition. The ant-based part of AntClass relies on the establishment of a 2 dimensional grid. The objects and the artificial ants are randomly placed on the grid. At each iteration, ants move and have a probability to pick-up or to drop an object depending if they have already one or not. Artificial ants generate heaps of similar objects that define the partition.

4.2 Data Sets and Experimental Protocol

We evaluate the clustering methods over several data sets that represent distinct clustering difficulties in the same experimental conditions in order to better appreciate the performance of each of them. First, we test K-Means, AntClass and AntClust over the artificial data sets that have been previously introduced in the section 3.1, namely $Art_{1,2,3,4,5,6}$.

Second, we test the algorithms over real data sets such as Iris, Pima, Soybean, Glass and Thyroid. We expect these data sets to be more difficult to be cluster, since they may be noisier than artificial data sets. We introduce in the table 1 the parameters that characterize the artificial and real data sets used for our evaluations. The fields for each data set are: the number of objects (N), their associated number of attributes (M), and the number of clusters (K).

For each data set, we run 50 times each method and compute the mean clustering success (see section 3.1), the mean number of clusters found and their respective standard deviations. The next section presents our results.

Table 1. Main characteristics of the data sets.

	Art_1	Art_2	Art_3	Art_4	Art_5	Art_6
N	400	1000	1100	200	900	400
M	2	2	2	2	2	8
K	4	2	4	2	9	4
	Iris	*Glass*	*Pima*	*Soybean*	*Thyroid*	
N	150	214	798	47	215	
M	4	9	8	35	5	
K	3	7	2	4	3	

4.3 Results with Artificial and Real Data Sets

The table 2 shows the results obtained over the artificial and real data sets. We can see in this table that K-Means gives, in general, the best results in term of clustering success and obviously in term of number of clusters found as they are provided to the algorithm. AntClust has the best clustering results only twice for the data sets *Soybean* and *Thyroid* that are little data sets (with respectively 47 and 215 objects). Nevertheless, AntClust shows its ability to treat large data sets with Art_2 and Art_3. AntClass performs better than AntClust three times, for Art_1, Art_5 and $Iris$. For Art_1 and Art_5, AntClass better estimates the number of expected clusters, but for $Iris$, the situation is reversed. This means that in this case, AntClust creates poorer quality clusters than those of AntClass. In all the other experiments K-Means and AntClust are more efficient than AntClass. One thing to point out is that when K-Means fails (for example in the data sets *Glass* and *Pima*), AntClass (which is partially based on K-Means method) and AntClust also behaves poorly. Nevertheless, AntClust performs well in general and can be even more efficient than K-Means for which the number of clusters is provided.

To complete the evaluation of AntClust, we compare its complexity to those of K-Means and AntClass. Let C_x be the complexity of the algorithm x. AntClust can be splitted in several steps: the template learning process ($\theta(N \times N_{App})$), the creation of the nests ($\theta(N \times Nb_{Iter})$) and finally the nests deletion process and the re-assignment of the ants to a nest. This step runs in quadratic time since each ant that has no more nest has to find the the most similar ant that belongs to a nest. The complexity of AntClust is $C_{AntClust} = \theta(N^2)$ in the worst case. The K-Means algorithm's complexity is known to be $C_{KMeans} = \theta(N)$ and AntClass has also a linear complexity $C_{AntClass} = \theta(N)$. Experimentally, our tests revealed that K-Means is the quickest method and that AntClust runs faster than AntClass.

Table 2. Mean number of clusters (# clusters) and mean success (Success) and their standard deviation for each data set and each method computed over 50 runs.

	# clusters						Success					
	K-Means		**AntClass**		**AntClust**		**K-Means**		**AntClass**		**AntClust**	
Data sets	mean	[std]	mean	[std]	mean	[std]	mean	[std]	mean	[std]	mean	[std]
Art_1	3.98	[0.14]	4.22	[1.15]	4.70	[0.95]	0.89	[0.00]	0.85	[0.05]	0.78	[0.03]
Art_2	2.00	[0.00]	12.32	[2.01]	2.30	[0.51]	0.96	[0.00]	0.59	[0.01]	0.93	[0.02]
Art_3	3.84	[0.37]	14.66	[2.68]	2.72	[0,88]	0.78	[0.02]	0.65	[0.01]	0.85	[0.02]
Art_4	2.00	[0.00]	1.68	[0.84]	4.18	[0.83]	1.00	[0.00]	0.71	[0.23]	0.77	[0.05]
Art_5	8.10	[0.75]	11.36	[1.94]	6.74	[1.66]	0.91	[0.02]	0.92	[0.01]	0.74	[0.02]
Art_6	4.00	[0.00]	3.74	[1.38]	4.06	[0.24]	0.99	[0.04]	0.89	[0.13]	0.95	[0.01]
Iris	2.96	[0.20]	3.52	[1.39]	2.82	[0.75]	0.86	[0.03]	0.81	[0.08]	0.78	[0.01]
Glass	6.88	[0.32]	5.60	[2.01]	5.90	[1.23]	0.68	[0.01]	0.60	[0.06]	0.64	[0.02]
Pima	2.00	[0.00]	6.10	[1.84]	10.66	[2.33]	0.56	[0.00]	0.53	[0.02]	0.54	[0.01]
Soybean	3.96	[0.20]	1.60	[0.49]	4.16	[0.55]	0.91	[0.08]	0.46	[0.17]	0.93	[0.04]
Thyroid	3.00	[0.00]	5.84	[1.33]	4.62	[0.90]	0.82	[0.00]	0.78	[0.09]	0.84	[0.03]

5 AntClust for Web Usage Mining

For the time being, a lot of research efforts have been conducted to cluster user sessions extracted from Web servers log files. The recurrent problem in this area is that clustering algorithms must be able to treat large data sets with an affordable computing time. Actually, a single Web server log file may contain several hundred thousand requests for Web pages. As a consequence, researchers try to use algorithms that run fast in their first approaches.

In [12], Yan et al. use a "First leader clustering algorithm". The sessions are expressed as numerical vectors containing for each Web page, the number of recorded impacts. Although results are very promising and quickly computed, this method is limited since the final partition depends on the order of the sessions in the data set.

Heer and Chi introduce in [13] the Wavefront algorithm that improves the initialization step of the K-Means method. The cluster seeds are randomly generated according to an estimated center of gravity of the data set. According to the authors, this method allows a quicker convergence. In their work, the Web sessions are expressed as multi-modal vectors that take into account the navigation of the users (the time spent for each page) and model a page as a combination of structure and content information.

Finally, Estivill-Castro et al. propose in [14] a robust clustering algorithm that is mainly a K-Means algorithm in which the median estimator is used instead of the mean estimator. The algorithm is then more resistant to the noise in the data sets.

There are two major limitations to use "K-Means like" algorithms in the Web usage mining context. First, the number of clusters K has to be provided to ensure a good convergence of the method, but it can't be easily set, unless

the user is able to guess how people navigate on her Web site, which is exactly the goal of the clustering process. Second, mean values have to be computed and limit the coding of the Web sessions to numerical expressions (difficulties for keywords or multimedia contents coding). Finally, mean values may not have any meaning in the Web sessions context.

5.1 Web Session Data

The last data set that we explore with AntClust is a Web server log file. We sort and filter this raw file to get a data set composed of users sessions. A session captures the activity of a user on a Web site during a specified period of time. The sessions have been recorded for one month in October 2001, on a web site of the University of Tours that contains computer sciences courses. The 1064 reconstructed sessions contain 667 unique sessions, that is to say, sessions that come from an IP number that has been seen only once in the requests log file. Consequently, our sessions may reflect a lot of distinct behaviors from the users, and thus may be very noisy. Nevertheless, as there are few hyperlinks between the online courses, we expect the clusters to be representative of a minimum of courses in order to be valuable and understandable. Similarly to other works, we coded the Web sessions as vectors where each component corresponds to the number of hits recorded for each page.

5.2 Results

When applied on Web sessions, AntClust finds 17 clusters. The 3 largest clusters contain the half of the sessions. Although the two largest clusters refer to 2 or 3 online computer sciences courses, the others generally reflect interests for only one course. This probably means that a majority of users went to the Web site with no specific goal and look at several courses to evaluate the content of the diploma. The other users were certainly already students and were searching for lecture notes relative to one topic. This little experiment proves that AntClust is able to generate a non-noisy partition of Web sessions that can help understanding the interests of the Web users. AntClust spends approximately 1.33 minutes at 650 MHz to cluster the user sessions, which is an affordable time.

6 Conclusion and Perspectives

AntClust is a new clustering algorithm that models the chemical recognition system of real ants. It associates one object of the data set to one ant and defines the expected partition as a set of nests. In this paper, we show how the internal parameters of the method can be set, regardless of the structure of the data set to be explored. Furthermore, we develop a non-deterministic approach to delete the uninteresting nest and to re-affect the ants which nest has been deleted. We evaluate its performance against K-Means and AntClass with artificial and real data sets. We show that AntClust can even do better than K-Means for two data

sets. When applied to Web sessions, AntClust finds meaningful clusters that can help understanding the interests of the users of the Web site. Our future works will try to apply AntClust to larger Web data sets to evaluate its robustness. In these experiments we will compare AntClust to other clustering algorithms in the Web context. We also plan to develop an incremental version of AntClust and our first results seem to be promising.

References

1. Y. Chiou and L. W. Lan, "Genetic clustering algorithms," *European journal of Operational Research*, no. 135, pp. 413–427, 2001.
2. L. Y. Tseng and S. B. Yang, "Genetic clustering algorithms," *European journal of Operational Research*, no. 135, pp. 413–427, 2001.
3. N. Monmarché, G. Venturini, and M. Slimane, "On how *Pachycondyla apicalis* ants suggest a new search algorithm," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 937–946, 2000.
4. A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the First European Conference on Artificial Life* (F. Varela and P. Bourgine, eds.), pp. 134–142, MIT Press, Cambridge, Massachusetts, 1991.
5. E. Lumer and B. Faieta, "Diversity and adaptation in populations of clustering ants," in Cliff *et al.* [15], pp. 501–508.
6. P. Kuntz and D. Snyers, "Emergent colonization and graph partitioning," in Cliff *et al.* [15], pp. 494–500.
7. N. Monmarché, M. Slimane, and G. Venturini, "On improving clustering in numerical databases with artificial ants," in *Lecture Notes in Artificial Intelligence* (D. Floreano, J. Nicoud, and F. Mondala, eds.), (Swiss Federal Institute of Technology, Lausanne, Switzerland), pp. 626–635, Springer-Verlag, 13–17 September 1999.
8. N. Labroche, N. Monmarché, and G. Venturini, "A new clustering algorithm based on the chemical recognition system of ants," in *Proc. of 15th European Conference on Artificial Intelligence (ECAI 2002), Lyon FRANCE*, pp. 345–349, 2002.
9. B. Hölldobler and E. Wilson, *The Ants.* Springer Verlag, Berlin, Germany, 1990.
10. N. Carlin and B. Hölldobler, "The kin recognition system of carpenter ants(camponotus spp.). i. hierarchical cues in small colonies," *Behav Ecol Sociobiol*, vol. 19, pp. 123–134, 1986.
11. J. Heer and E. Chi, "Mining the structure of user activity using cluster stability," in *Proceedings of the Workshop on Web Analytics, SIAM Conference on Data Mining (Arlington VA, April 2002).*, 2002.
12. T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal, "From user access patterns to dynamic hypertext linking," in *Proc. of 5th WWW*, pp. 1007–1014, 1996.
13. J. Heer and E. Chi, "Identification of web user traffic composition using multimodal clustering and information scent," 2001.
14. V. Estivill-Castro and J. Yang, "Categorizing visitors dynamically by fast and robust clustering of access logs," *Lecture Notes in Computer Science*, vol. 2198, pp. 498–509, 2001.
15. D. Cliff, P. Husbands, J. Meyer, and S. W., eds., *Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3*, MIT Press, Cambridge, Massachusetts, 1994.