AntTree: Web Document Clustering using Artificial Ants

Azzag Hanene[1] and **Christiane Guinot**[2] and **Gilles Venturini**[3]

Abstract.
We present in this work a new algorithm for document hierarchical clustering and automatic generation of portals sites. This model is inspired from the self-assembling behavior observed in real ants where ants progressively get attached to an existing support and successively to other attached ants. The artificial ants that we have defined will similarly build a tree. Each ant represents one document. The way ants move and build this tree depends on the similarity between the documents. We have tested our model on sets of web pages extracted from internet and we have successfully compared our results to those obtained by the AHC (Ascending Hierarchical Clustering).

1 Introduction

The context of this work is the automatic construction of portal sites for the Web. A portal site can be viewed as a hierarchical partitioning of a set of documents which aims at recursively reproducing the following property: at each node (or category), the sub-categories are similar to their mother, but they are as much dissimilar to each others. One of the major problems to solve in this area is the automatic definition of this hierarchy of documents which, in actual systems, must be given by a human expert [3][7][15][11]. If one works with an important number of documents, or if one wishes to let the computer autonomously do all the work, then standard approaches are useless.

We propose a new approach which builds a tree-structured partitioning of the data that checks the recursive property mentioned above. This method simulates a new biological model: the way ants build structures by assembling their bodies together. Ants start from a point and progressively become connected to this point, and recursively to these firstly connected ants, etc. They can move on the living structure to find the best place where to be connected in a distributed way. This behavior can be adapted to build a tree from the data to be clustered.

The rest of this paper is organized as follows: section 2 describes the ants biological behavior. Section 3 describes the related approaches found in the literature, and how we have modeled this behavior in the AntTree algorithm. In section 4, we have presented a comparative study between AntTree and Ascending Hierarchical Clustering (AHC) on several texts databases. We show also in this section how AntTree may efficiently generate portal sites. Section 5 concludes on this work and presents the extensions currently under study.

[1] Ecole Polytechnique de l'Université de Tours, France, email: hanene.azzag@etu.univ-tours.fr
[2] CE.R.I.E.S., 20 rue Victor Noir, 92521 Neuilly sur Seine Cedex, France, email: christiane.guinot@ceries-lab.com
[3] Ecole Polytechnique de l'Université de Tours, France, email: venturini@univ-tours.fr

2 Biological model

The numerous abilities of ants have inspired researchers for more than ten years regarding designing new clustering algorithms. The model which has been studied the most is the way ants sort objects in their nest [4][10][8][12]. These ants based algorithms may inherit from real ants interesting properties, such as the local/global optimization of the partitioning, the absence of need of a priori information on an initial partitioning or number of classes, parallelism, etc.

In this paper, we deal with a new model that can be observed in several ants species and that we briefly describe here[9][16]: these insects may become fixed to one another to build live structures with different functions. Ants may thus build ”chains of ants” in order to fill a gap between two points, or build a nest by closing the edges of a leaf, or form ”drops of ants”, a function which is not yet well understood. The general principles that rule these behaviors are the following: ants start from an initial point (called the support). They begin to connect themselves to this support and then progressively to previously connected ants. When an ant is connected, it becomes a part of the structure and other ants may move over this ant or connect themselves to it. The structure grows over time according to the local actions performed by the ants that move over the structure to find a location where they can connect. These ants are influenced by the local shape of the structure but also by a visual attractor (for instance, the point to reach). Ants which are in the middle of the structure cannot easily disconnect themselves but if they do, then parts of the structure may fall down. A phenomenon of structure decay is also apparent.

3 The AntTree algorithm

3.1 Main principles and related work

From those elements, we define the outlines of our computer model which simulates this behavior for tree building. The n ants $a_1, ..., a_n$ represent each of the n data (documents for instance) $d_1, ..., d_n$ of the database. These ants are initially placed on the support which is denoted by a_0. Then, we successively simulate one action for each ant. An ant can be in two states: it can be free (i.e. disconnected) and may thus move over the structure in order to find a place where it can connect, or it can be connected to the structure without any possibility of moving apart from to disconnect. In this work, an ant may become connected to only one other ant. Each ant a_i can have one outgoing link and severals incoming links limited to a maximum value (proportional to the numbers of data), this ensures that ants will build a tree. a_0 also have several incoming links limited to a maximum value. Ants locally perceive the structure: a moving ant a_i located over an ant a_{pos}, which is itself connected to the structure, perceives a neighborhood N_{pos}, which is limited to a_{pos}, to the (mother) ant to which a_{pos} is connected, and to the (daughter) ants which are connected to a_{pos}. a_i can perceive the similarities between the data d_i it is representing and the data represented by ants of N_{pos}. According to these similarities, a_i may either get connected to a_{pos}, or move to one of the ants in N_{pos}. Once all ants are connected to the structure, then our algorithm stops. The resulting tree can be interpreted as a partitioning of the data. The properties that we wish to obtain for data clustering in the context of portal site building are the following: each sub-tree T represents one category compound of all ants in this sub-tree. Let a be the ant which is at the root of T. We would like that 1) a is as the best representative of this category (ants placed below a are as much similar to a as possible), 2) the "daughters" ants of a which represent sub-categories of a are as dissimilar to each others as possible (well separated sub-categories). This property should be recursive and possibly checked anywhere in the hierarchical structure.

This model distinguishes itself from previously studied ant-based algorithms like those using pheromones for instance: here ants do not communicate via an external support but are themselves the support. One must notice however that tree building with ants has already been tackled in the literature. On the one hand, we mention the work done on genetic programming with artificial ants [13] and the ACO approach [2]. Ants build trees by assigning probabilities (i.e. pheromones) to symbols. On the other hand, the ACO approach has also been used in the specific context of phylogenetic tree building [1]. Ants use pheromones to determine the order of the symbols that will be used to build an unrooted tree. However, these two models are not centered on the data clustering problem and have no common points with the self-assembly behavior that we are simulating. We also mention the work done in real robotics where population of self assembling robots may build complex structures [5].

3.2 A stochastic and distributed algorithm

```
1. Initially, all ants are placed on the support and their similarity and
   dissimilarity thresholds are respectively initialized according to
   the properties defined in section 3.3
2. While there exists a non connected ant $a_i$ Do
3.     If $a_i$ is located on the support Then Support case
4.     Else Ant case
5. End While
```

Figure 1. Main algorithm of AntTree

The main algorithm is represented in figure 1. All ants use the same behavioral rules where two cases are distinguished: when the simulated ant a_i is located on the support, and when a_i is located on a connected ant.

When the simulated ant a_i is connected on the support (see figure 2), the first case to be considered is when no other ant is connected to the support (the tree is limited to the support a_0). a_i is directly connected to the support (see line 1 in figure 2). If at least one ant is already connected to the support, then a_i is moved toward the most similar ant a^+ if it is sufficiently similar to it (see line 2(b)). Else, if a_i is dissimilar enough to the other connected ants (see line 2(c)i), then it is allowed to connect to the support (and thus to create a new subcategory). Finally, if a_i fulfills none of these two

```
1. If no ant is connected yet to the support $a_0$ Then connect $a_i$ to $a_0$
2. Else
   (a) let $a^+$ be the ant connected to $a_0$ which is the most similar to $a_i$
   (b) If $\mathrm{Sim}(a_i, a^+) \geq T_{\mathrm{Sim}}(a_i)$ Then move $a_i$ toward $a^+$ /* $a_i$ is
       similar enough to $a^+$ */
   (c) Else
       i. If $\mathrm{Sim}(a_i, a^+) < T_{\mathrm{Dissim}}(a_i)$ Then /* $a_i$ is dissimilar enough
          to $a^+$ */ connect $a_i$ to the support $a_0$ (or, if no incoming link
          available on $a_0$, decrease $T_{\mathrm{Sim}}(a_i)$ and move $a_i$ toward $a^+$ )
       ii. Else decrease $T_{\mathrm{Sim}}(a_i)$ and increase $T_{\mathrm{Dissim}}(a_i)$ /* $a_i$ is more
           tolerant */
```

Figure 2. Support case

conditions (similarity or dissimilarity), then it is made more tolerant in order to increase its chances to move or to become connected in the future (see line 2(c)ii).

```
1. let $a_{pos}$ denote the ant on which $a_i$ is located, and let $a_k$ denote a
   randomly selected neighbor of $a_{pos}$,
2. If $\mathrm{Sim}(a_i, a_{pos}) \geq T_{\mathrm{Sim}}(a_i)$ Then
   (a) Let $a^+$ be the neighbor ant of $a_{pos}$ which is the most similar to $a_i$
   (b) If $\mathrm{Sim}(a_i, a^+) < T_{\mathrm{Dissim}}(a_i)$ Then Connect $a_i$ to $a_{pos}$ (or, if no
       incoming link is available on $a_{pos}$, randomly move $a_i$ toward
       $a_k$)
   (c) Else decrease $T_{\mathrm{Dissim}}(a_i)$, increase $T_{\mathrm{Sim}}(a_i)$ and move $a_i$ to-
       ward $a_k$
3. Else randomly move $a_i$ toward $a_k$
```

Figure 3. Ant case

When the simulated ant is located on a connected ant a_{pos} (see figure 3), then a similar behavior is implemented. First, if a_i is similar enough to a_{pos}, then it can be considered for the creation of a new subcategory (see line 2 figure 3). However, this creation only takes place provided that a_i is dissimilar enough to a^+ (see line 2(b)). This ensures that if a sub-category is created, it will be related to its "'mother"' category and it will be as dissimilar as possible to its "'sisters"' categories. If a_i cannot be connected, then it is made more tolerant and it is randomly moved to any neighbor position around a_{pos} (see line 2(c)).

3.3 Self-adaptive thresholds

We have observed that the values of the two thresholds may vary from one database to the other, and that the automatic computation mentioned previously may not always correctly adapt these thresholds to the data. We have thus tested two methods to self-adapt those thresholds, the first method is global and consists in letting the ants modify the same thresholds values common to all ants, and the second method is local and is such that each ant adapts its own private thresholds.

In the global method, the thresholds are $T_{Sim} = 0.99 * T_{Sim-init}$ and $T_{Dissim} = 0.01 + T_{Dissim-init}$. Each time an ant fails the similarity test related to the connection, then T_{Sim} is decreased and T_{Dissim} increased, in order to make all ants more tolerant. The values $T_{Sim-init}$ and $T_{Dissim-init}$ can be initialized respectively to 1 and 0. We have observed that initializing those values respectively to $Sim_{mean-sup}$ and $Sim_{mean-inf}$ increases the convergence rate of this algorithm, where $Sim_{mean-sup}$ and $Sim_{mean-inf}$ as the mean of similarities which are above (respectively below) the mean similarities between all of the data.

In the local method, the thresholds are computed as previously mentioned but are local to each ant a_i (T_{Sim} and T_{Dissim} now becomes $T_{Sim}(a_i)$ and $T_{Dissim}(a_i)$). a_i adapts its thresholds according to the results of its actions. This algorithm may thus better adapt the thresholds to the local distribution of the data (outliers for instance). The experimental results confirms this analyzis, we have obtain competitive performances (quality, computation time) compared to standard methods.

3.4 Initial sorting of the data

First, for our algorithm, we have tried to find the best strategy for initial data sorting (this sort is performed in $O(n \, ln \, n)$ and does not increase the complexity of the algorithms).The initialization step of the algorithm influences the results, especially because the first ants will be connected to the support. Initially, data was chosen in a random way (without sorting). The second time, data had been sorted according to increasing order of the average similarity between each others. In this way, the first connected ants are those which are the less similar to all the others and therefore close to their cluster, and far away from the others. With a decreasing order, the first ants to connect are the most similar to the others. Thus an ant belonging to a different cluster will have more chance of being connected than in the increasing case. We have experimentally observed that the increasing order gives the best results. The most dissimilar data seem to be good starting points for building classes in our model.

4 Experimental study and automatic portal site generation

4.1 Comparative study

Table 1. Description of used databases (see text for more explanation)

Databases	Size (# of documents)	Size (Mb)	# of classes
Reuters	1025	4.05	9
CE.R.I.E.S.	258	3.65	17
Database 1	319	13.2	4
Database 2	524	20	7

We have evaluated AntTree on 4 databases which have from 258 to 1025 texts (see table 1). The Reuters databases contains 1025 texts extracted from the reuteurs21578 database (8653 texts). The CE.R.I.E.S. database contains 258 texts dealing with human skin (the CE.R.I.E.S. is a laboratory funded by Chanel). Database 1 consists of scientific web pages (73 about scheduling, 84 about pattern recognition, 81 about TcpIp network, 94 about vrml courses). Database 2 consists of web pages with general topics (55 about c++ courses, 82 about the Danone food company, 86 about IEEE, 90 about cinema, 50 about the Le Monde newspaper, 63 about the sfr phone company, 101 about medicine category from Google's directory). The real classes of data are of course not given to the algorithms. They are used in the final evaluation of the obtained partitioning.

The evaluation of the results is performed with the number of clusters C_f, with the purity P_r of clusters (percentage of correctly clustered data in a given cluster), and with a classification error Ec (proportion of data couples which are not correctly clustered, i.e. in the same real cluster but not in the same found cluster, and vice versa:

$$Ec = \frac{2}{N(N-1)} \sum_{(i,j)\in\{1,\ldots,N\}^2, i<j} \epsilon_{ij} \qquad (1)$$

where :

$$\epsilon_{ij} = \begin{cases} 0 & \textit{if} \, (Cr_i = Cr_j \wedge Cf_i = Cf_j) \vee \\ & (Cr_i \neq Cr_j \wedge Cf_i \neq Cf_j) \\ 1 & \textit{else} \end{cases} \qquad (2)$$

The results are averaged over 15 trials. We have also presented standard deviations which are respectively denoted by σ_C, σ_P and σ_{Ec}.

We have compared our algorithm to the Hierarchical Ascending Clustering [6] which is an efficient hierarchical method currently used in the industry (see for instance the SAS software).

We have used the Ward criterion for cutting the dendrogram. The same similarity measure is used for both algorithms. It is based on the well known cosine measure [14] which encodes each text as a vector of word count. We have used a common weighting schemes, i.e. *tf-idf* (term frequency - inverse document frequency). *tf* denotes the word count of the document and *idf* denotes the inverse document frequency (document frequency is the number of documents which contain the considered word).

Table 2. Purity obtained with AntTree and AHC.

	AHC	AntTree
DataBases	P_r	$P_r[\sigma_{P_r}]$
Reuters	0.50	0.40 [0.007]
CERIES	0.30	0.37 [0.012]
Database 1	0.82	0.68 [0.012]
Database 2	0.52	0.80 [0.009]

P_r averaged purity obtained on 15 runs
σ_P standard deviations

Table 3. Results obtained by AntTree and AHC on several databases.

		AHC		AntTree	
DataBases	C_r	Ec	C_f	$Ec[\sigma_{Ec}]$	$C_f[\sigma_{C_f}]$
Reuters	9	0.21	5	0.35 [0.004]	12 [0.00]
CERIES	17	0.21	7	0.15 [0.001]	17 [0.00]
Database 1	4	0.09	7	0.29 [0.011]	7 [0.00]
Database 2	7	0.23	3	0.10 [0.006]	8 [0.00]

Ec averaged classification error obtained on 15 runs
C_r number of real clusters
C_f averaged number of clusters found on 15 runs
σ_x standard deviations

Table 4. Prossesing Time: seconds

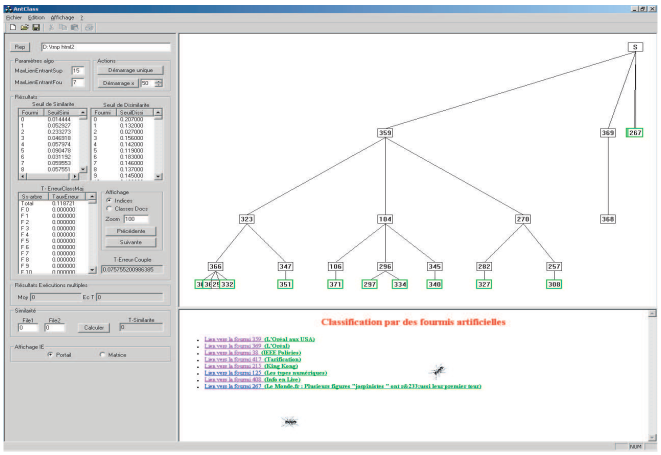
Databases	AntTree	AHC
Reuters	1.51	120
CERIES	0.04	4
Database 1	0.12	6
Database 2	0.34	25

Results are presented in tables 2, 3 and 4 with detail of a confusion matrix in 5. Our analysis of the results is as follows: on average (see table 2), both algorithms obtain similar performances with respect to the purity of classes. Each algorithm outperforms the other in the same number of cases. In general, AntTree gives better results than AHC when data are very dissimilar from each others. However, table 3 shows that AntTree better approximate the number of classes: cutting the dendogram is difficult, while it is straightforward to interpret the results given by AntTree (the branches at the top indicate the classes). Another positive advantage of AntTree is its computation time (see table 4). The use of the tree structure clearly lowers the time complexity of AntTree. Furthermore, the memory space required is linear with the size of the database while fast implementations of AHC have higher space complexity.

Table 5. Purity matrix obtained with DataBase 2: C_f denotes the "found clusters" and C_r the real clusters.

Datas	C_r1	C_r2	C_r3	C_r4	C_r5	C_r6	C_r7	P_r
C_f1	86	0	0	0	0	0	0	1
C_f2	0	0	0	0	0	0	2	1
C_f3	0	5	89	42	2	5	14	0.56
C_f4	0	0	0	0	76	0	0	1
C_f5	0	0	0	0	0	31	3	0.91
C_f6	0	28	0	0	0	0	0	1
C_f7	0	21	1	8	3	1	80	0.70
C_f8	0	1	0	0	0	26	0	0.96

4.2 Generating a portal site



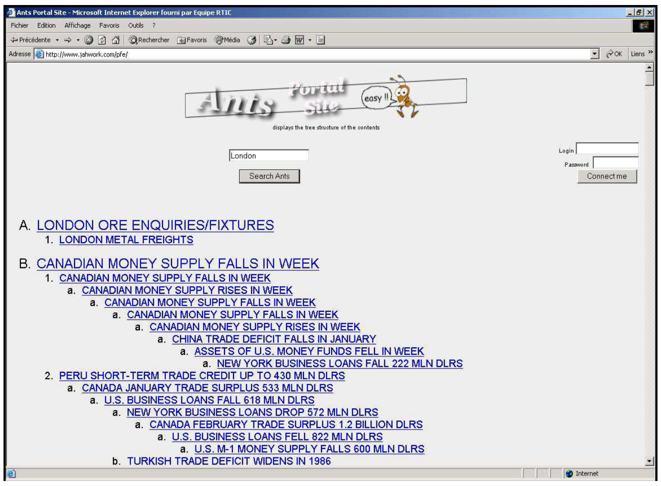
Classification par des fourmis artificielles

- Lien vers la fourmi 359 (L'Oréal aux USA)
- Lien vers la fourmi 369 (L'Oréal)
- Lien vers la fourmi 38 (IEEE Policies)
- Lien vers la fourmi 417 (Tarification)
- Lien vers la fourmi 215 (King Kong)
- Lien vers la fourmi 125 (Les types numériques)
- Lien vers la fourmi 408 (Info en Live)
- Lien vers la fourmi 267 (Le Monde.fr : Plusieurs figures "jospinistes " ont r&233;ussi leur premier tour)

Figure 4. AntTree software: interactive visualization of the tree and its HTML view

We show in figure 4 the interface of the AntTree software. This interface allows the user to explore the generated tree. We make use of a hyperbolic display to zoom on specific part of the tree without loosing the global context (general shape of the tree). The user may click on the nodes to select a document.

Once the texts have been clustered in a tree, then it is straightforward to generate the corresponding portal site. The hierarchy of documents is represented in our actual implementation as a directory tree with indentation. The tree is encoded in a database in a few seconds and the generation of HTML files is dynamic. Figure 5 gives a typical example of the portal home page obtained for Reuters (1024 documents). In figure 6 we show how we have integrated a search engine based on a word index. This index is automatically generated in the database.


Ants Portal Site - Microsoft Internet Explorer fourni par Equipe RTIC
Fichier Edition Affichage Favoris Outils ?
Précédente
Rechercher
Favoris
Média
Adresse
OK
Liens
Ants Portal Site
easy !!
Login
Password
Search Ants
Connect me
Internet

http://www.jahwork.com/pfe/

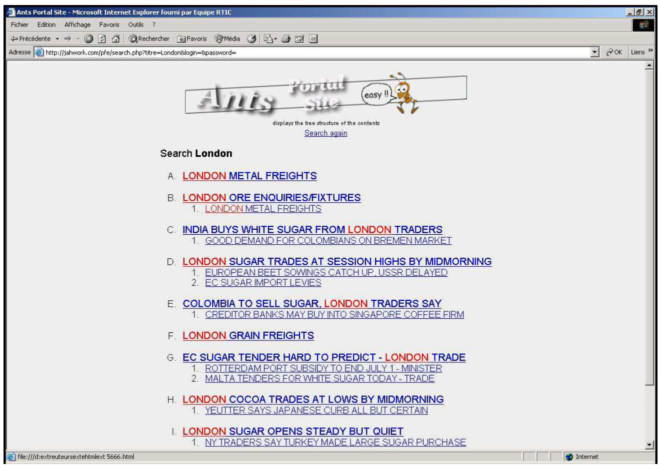
displays the tree structure of the contents

London

A. LONDON ORE ENQUIRIES/FIXTURES
   1. LONDON METAL FREIGHTS

B. CANADIAN MONEY SUPPLY FALLS IN WEEK
   1. CANADIAN MONEY SUPPLY FALLS IN WEEK
      a. CANADIAN MONEY SUPPLY RISES IN WEEK
         a. CANADIAN MONEY SUPPLY FALLS IN WEEK
            a. CANADIAN MONEY SUPPLY FALLS IN WEEK
               a. CANADIAN MONEY SUPPLY RISES IN WEEK
                  a. CHINA TRADE DEFICIT FALLS IN JANUARY
                     a. ASSETS OF U.S. MONEY FUNDS FELL IN WEEK
                        a. NEW YORK BUSINESS LOANS FALL 222 MLN DLRS
   2. PERU SHORT-TERM TRADE CREDIT UP TO 430 MLN DLRS
      a. CANADA JANUARY TRADE SURPLUS 533 MLN DLRS
         a. U.S. BUSINESS LOANS FALL 618 MLN DLRS
            a. NEW YORK BUSINESS LOANS DROP 572 MLN DLRS
               a. CANADA FEBRUARY TRADE SURPLUS 1.2 BILLION DLRS
                  a. U.S. BUSINESS LOANS FELL 822 MLN DLRS
                     a. U.S. M-1 MONEY SUPPLY FALLS 600 MLN DLRS
            b. TURKISH TRADE DEFICIT WIDENS IN 1986

Figure 5. A typical portal site generated from the Reuters (1024 texts) in a few seconds only


Ants Portal Site - Microsoft Internet Explorer fourni par Equipe RTIC
Fichier Edition Affichage Favoris Outils ?
Précédente
Rechercher
Favoris
Média
Adresse
OK
Liens
Ants Portal Site
easy !!
Internet

http://jahwork.com/pfe/search.php?titre=London&login=&password=

displays the tree structure of the contents

Search again

Search **London**

A. LONDON METAL FREIGHTS

B. LONDON ORE ENQUIRIES/FIXTURES
   1. LONDON METAL FREIGHTS

C. INDIA BUYS WHITE SUGAR FROM LONDON TRADERS
   1. GOOD DEMAND FOR COLOMBIANS ON BREMEN MARKET

D. LONDON SUGAR TRADES AT SESSION HIGHS BY MIDMORNING
   1. EUROPEAN BEET SOWINGS CATCH UP, USSR DELAYED
   2. EC SUGAR IMPORT LEVIES

E. COLOMBIA TO SELL SUGAR, LONDON TRADERS SAY
   1. CREDITOR BANKS MAY BUY INTO SINGAPORE COFFEE FIRM

F. LONDON GRAIN FREIGHTS

G. EC SUGAR TENDER HARD TO PREDICT - LONDON TRADE
   1. ROTTERDAM PORT SUBSIDY TO END JULY 1 - MINISTER
   2. MALTA TENDERS FOR WHITE SUGAR TODAY - TRADE

H. LONDON COCOA TRADES AT LOWS BY MIDMORNING
   1. YEUTTER SAYS JAPANESE CURB ALL BUT CERTAIN

I. LONDON SUGAR OPENS STEADY BUT QUIET
   1. NY TRADERS SAY TURKEY MADE LARGE SUGAR PURCHASE

file:///d:extreuteursextehtmlext 5666.html

Figure 6. Searching through the portal site

5 Conclusion and perspectives

We have shown in this paper how a new model based on the self-assembly behavior of real ants can be applied to the hierarchical clustering problem. We have shown how this model can be used to cluster documents. The comparison with an efficient hierarchical clustering algorithm (AHC) is positive especially with respect to the number of found clusters and to the computation time. As a consequence, we have shown that AntTree can be efficiently used for automatic portal site generation.

The extensions of this algorithm concern the following points: we wish to deal with larger collection of texts using a sampling strategy. The texts which are not used for building the tree are directly connected to the tree by following a path of greatest similarity (this is a fast procedure). We are currently developping an interactive editor to modify the results of the algorithm: in this way, the user has the possibility to adjust the obtained results. Finally, we want to improve our model, each ant will have the possibility to disconnect itself from its position and to move on others ants perhaps more similar than that on which it is. We also wish to generalize AntTree to the generation of graphs (and not just trees). We could generate hypertexts with the same self-assembly principles.

REFERENCES

[1] Shin Ando and Hitoshi Iba, 'Ant algorithm for construction of evolutionary tree', in *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, ed., W. B. Langdon, p. 131, New York, (9-13 July 2002). Morgan Kaufmann Publishers.

[2] L. Bianchi, L. M. Gambardella, and M. Dorigo, 'An ant colony optimization approach to the probabilistic traveling salesman problem', in *Proceedings of PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science. Springer Verlag, Berlin, Germany, (2002).

[3] D. Filo and J. Yang. Yahoo!, 1997.

[4] S. Goss and J.-L. Deneubourg, 'Harvesting by a group of robots', in *Proceedings of the First European Conference on Artificial Life*, ed., Varela, pp. 195–204, Paris, France, (1991). Toward a Practice of Autonomous Systems.

[5] K. Hosokawa, T. Fujii, H. Kaetsu, H. Asama, and I. Endo Y. Kuroda, 'Self-organizing collective robots with morphogenesis in a vertical plane', in *proc. of 1998 IEEE int conf Robotics and automation, 2858-2863*, (1998).

[6] A. K. Jain, M. N. Murty, and P. J. Flynn, 'Data clustering: a review', *ACM Computing Surveys*, **31**(3), 264–323, (1999).

[7] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, 'On semi-automated web taxonomy construction', in *WebDB*, Santa Barbara, (May 2001).

[8] P. Kuntz, D. Snyers, and P. Layzell, 'A stochastic heuristic for visualising graph clusters in a bi-dimensional space prior to partitioning', *Journal of Heuritics*, **5**(3), (October 1999).

[9] A. Lioni, C. Sauwens, G. Theraulaz, and J.-L. Deneubourg, 'The dynamics of chain formation in oecophylla longinoda', *Journal of Insect Behavior*, **14**, 679–696, (2001).

[10] 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 Behaviour*, pp. 501–508, (1994).

[11] Andrew K. McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore, 'Automating the construction of internet portals with machine learning', *Information Retrieval*, **3**(2), 127–163, (2000).

[12] N. Monmarché, 'On data clustering with artificial ants', in *AAAI-99 & GECCO-99 Workshop on Data Mining with Evolutionary Algorithms: Research Directions*, ed., A.A. Freitas, pp. 23–26, Orlando, Florida, (July 18 1999).

[13] O. Roux and C. Fonlupt, 'Ant programming: Or how to use ants for automatic programming', *From Ant Colonies to Artificial Ants: 2nd International Workshop on Ant Colony Optimization*, (2000).

[14] Gerard Salton and Christopher Buckley, 'Term weighting approaches in automatic text retrieval', in *information processing and management*, volume 25, pp. 513–523, (1988).

[15] Mark Sanderson and W. Bruce Croft, 'Deriving concept hierarchies from text', in *Research and Development in Information Retrieval*, pp. 206–213, (1999).

[16] G. Theraulaz, E. Bonabeau, C. Sauwens, J.-L. Deneubourg, A. Lioni, F. Libert, L. Passera, and R.-V. Sol, 'Model of droplet formation and dynamics in the argentine ant (linepithema humile mayr)', *Bulletin of Mathematical Biology*, (2001).