A MULTI-AGENT ARCHITECTURE FOR AGENTS CLUSTERING

Roland Coma
Gaële Simon
Michel Coletta
Computer Science Laboratory of Le Havre (LIH)
rue Phillipe Lebon, 76058 Le Havre Cedex,
France
E-mail: roland.coma@wanadoo.fr, gsimon@iut.univ-lehavre.fr, michel.coletta@iut.univ-lehavre.fr

KEYWORDS
Agents, Organizations, Clustering, Ants algorithms, Data streams.

ABSTRACT

In this paper, a method for dynamic clustering of agents is presented. The needs, the constraints and the particularities of this kind of clustering are presented. It is then shown that standard clustering methods are not suited to theses constraints. Shared requirements with clustering methods for data streams are presented. Then a multi-agent architecture supporting dynamic clustering of agents and satisfying most of these requirements is described. This architecture couples an ants algorithm with cluster agents helping ants to converge more quickly. This architecture is being used in a complex multi-agent system whose goal is to help to manage industrial risks.

INTRODUCTION

Why Clustering Agents ?

The work presented in this paper takes place in the context of the automatic analysis of an agents population with explicit and/or implicit organizations. Explicit organizations are statically defined during the design of the system, as in Aalaadin model (Ferber and Gutknecht 1998) with groups and roles. On the contrary, implicit organizations appear during the system execution. As a consequence, each agent may interact with other agents according to these organizations. Moreover, each agent is supposed to maintain a set of variables which reflect its current internal state. For example, these variables can represent the number of its communications, its reinforcement value (for eco-agents (Ferber and Jacopin 1991) or agents with learning capabilities), or its position in a situated environment (e.g. Ants (Lumer and Faieta 1994)). In the sequel, these variables will be called measures. The definition and the choice of these measures is another research problem which is not studied in this paper.

The problem we are interested in is to analyze and extract information about the global behavior of the agents population using the population description and the agents measures. More precisely, our goal is to build tools allowing to characterize the global state of the population instead of the different agents states. This problem is often encountered when developing multiagent systems, especially using multi-layers architectures like in (Marcenac 1997). Indeed, these different layers are usually used to reason on the problem to solve at different abstraction levels. The global state of each layer often represents a particular abstraction of the problem to solve. In order to allow the higher layer to reason about this abstraction, this global state must be summarized and reified. In other words, the problem is to establish a bijection between the needed abstraction and the agents population representing this abstraction using a characterization of the global state of this agents population.

This problem must also be solved when a multiagent system requires a scale change. For instance, in (Bertelle et al. 2000), a multiagent system is used to represent a fluid flow, like rivers. In this context, each agent represents a fluid particle interacting with agents (particles) which are in its neighborhood. In this kind of system, it is sometimes useful to perform a scale change in order to reify structures corresponding to sets of particles which are in particular interactions, like an eddy of water for example.

Characterizing the global state of an agents population can be performed by representing this population in terms of structures. These structures can be defined by situating each agent by comparison with the others using their measures. As said before, each agent gives information about its internal state by a set of measures. But measures of an agent can only be interpreted comparing them with the others. This comparison can be used to determine agents groups composed of agents sharing similar values of measures. In some cases, these groups can represent implicit organizations built by the multiagent system itself during its execution.

To achieve this groups detection, we propose to use dynamic clustering techniques. In the following paragraph, what we mean by dynamic clustering of agents is presented and compared to standard clustering.

Clustering Agents : Difficulties and Constraints

Clustering (Kaufman and Rousseeuw 1990) a set of objects consists in finding clusters of objects in this set. A cluster represents a set of objects which are considered to be similar. This similarity between objects depends generally on a distance between objects' attributes, each object being represented by a vector of attributes. So, standard clustering algorithms start from a set of objects to produce a set of clusters. Applying clustering techniques to a set of agents

implies to specialize the notion of object. In this context, an object is the vector of an agent's measures. But agents evolve continuously during the system execution. This property strongly modifies the clustering context in two ways :

1. the cardinal of the set of objects to cluster can often change. Indeed, agents can appear or disappear during the clustering process;
2. objects already clustered can be modified as their corresponding agents evolve.

So, a dynamic and incremental clustering method is needed in order to modify cleverly the set of obtained clusters so that they can be the more accurate as possible with respect to the agents population state. Standard clustering methods can not support this kind of property because they suppose the set of objects to cluster to be defined at the beginning. A few, like k-Means algorithm (Jain and Dubes 1988), are incremental ones. This means that when clusters are obtained, a new object can be clustered by searching the cluster whose gravity center is the nearest from the point representing the new object. Even if this mechanism is useful for our problem, it is not enough. Indeed, in our context, the algorithm must also be able to create, destroy or modify clusters according to the objects evolution. More dynamic algorithms used for data streams clustering (Barbarà 2002) meet part of these requirements. These requirements and algorithms are shortly presented in the next section.

Another constraint our clustering method must satisfy is to allow an easy integration in a multiagent environment. This set of constraints led us to consider clustering algorithms based on ants agents (Lumer and Faieta 1994). As it is presented in this paper, the main advantages of this kind of algorithms are to be dynamic and agentified, which satisfies our constraints. Unfortunately, these algorithms do not converge quickly enough in most cases. That's why we propose to add new agents to the ant mechanism whose main goal is to optimize the convergence.

The next section deals with the ants algorithms principles and details their advantages and disadvantages. It also describes requirements and algorithms for data streams clustering. It finally presents a particular ants algorithm called AntClass (Monmarché 2000 ; Monmarché et al. 1999) combining both ants and a second clustering algorithm (k-Means algorithm). Section 3 finally describes our proposal, partially based on AntClass.

EXISTING ALGORITHMS

Traditional Algorithms

Many algorithms were proposed to cluster data. Various criteria can be used to compare them (cost, convergence, relevance of the results…). The disadvantage of these methods is to operate on a set of static data. In our case, the data are evolutionary. They characterize elements of an agents organization which can appear, disappear or change. The ants algorithms seem to be more suited to this kind of problem.

Ants Algorithms

From the observation of ants, many research have allowed to better understand their behavior , like brood sorting or cemetery organization. (Deneubourg et al. 1990). From this analysis, clustering algorithms using ants agents have been defined.
Clustering is based on a kind of aggregation phenomenon. The basic mechanism underlying this phenomenon is an attraction between objects mediated by ants: clusters of objects grow by attracting ants to deposit more objects. To obtain coherent groups, a measure of dissimilarity between objects must be used (Lumer and Faieta 1994).
These algorithms have some disadvantages, especially as far as the number of clusters is concerned. Indeed, they can build too much clusters. Moreover they often leave isolated objects which are not clustered. As a consequence, their convergence is often slow. On the other hand, this kind of algorithm seems to be promising to take into account the evolution of data needed in our problem. Indeed, their behavior gives a kind of "dynamic" property to the clustering algorithm. For instance, this algorithm gives a simple way to process new data : they just need to be placed on the ants grid as if they were initial data. Moreover, as ants always move, data evolution inside an existing cluster can be taken into account by an ant visiting the cluster. This ant will see the cluster evolution as a building error and will try to put too dissimilar objects elsewhere. As it will be presented in the section focusing on AntClass, ants algorithms provide potential basic mechanisms to process dynamic data. All the problem is to find complementary mechanisms in order to provide a good convergence.

Clustering Data Streams

In a lot of applications, it is necessary to highlight structures in data sets which are evolutionary. These kind of data sets are called data streams. This is the case, for example, in the observation of weather data, the observation of traffic, the monitoring of a set of sensors, the evolution of an epidemic. The clustering of agents shares common characteristics with these applications. The challenge is to design algorithms which can detect structural changes in data. Such dynamic clustering methods are proposed in (Barbará and Chen 2000 ; Guha et al. 2000)
Conditions to evaluate these methods have been studied recently (Barbarà 2002). The compactness of representation, the incremental processing of new data, and the identification of new data which can trouble the current clustering model, are major constraints. The requirement of compactness is obvious. In the context of agents clustering, the number of agents is not intended to always grow that's why this requirement is satisfied. The incremental processing requires to avoid an exhaustive comparison between a new data and all data already clustered. It also requires to place the new data quickly (linear cost). The ants algorithm satisfies the first condition but not the second one. The last requirement means that new structural tendencies can appear : new clusters can appear, some clusters can disappear. This requirement needs to enrich the ants algorithm by complementary techniques in order to offset the slow convergence of ants.

The principle of our approach is to keep ants for their dynamic characteristic and to give to clusters they build the capacity to react. The agent approach is the best way to allot to them such a behavior. The next section describes AntClass algorithm which is an ants based algorithm. A part of this algorithm is used as a basis for our approach presented in the last part of the paper.

AntClass

AntClass (Monmarché et al. 1999 ; Monmarché 2000) is an ants-based algorithm which couples ants with a more static algorithm : k-Means. The main idea of this algorithm is to offset ants lack of convergence using a second algorithm which is static but gives good results when starting from relevant initial data partitions. As a consequence, the result produced by ants is used as a starting point for k-Means algorithm. More exactly, AntClass works in four steps: the first and the third steps are based on ant methods, the second and the fourth are based on k-Means.

In the first step, ants are used to produce initial clusters of data. At the beginning, data are randomly placed on a grid. Ants are moving randomly on this grid with a probability to change their direction. They are able to carry objects (data) and to put them on heaps. At the end of this process, an objects heap corresponds to a cluster. When an ant carries an object and when it moves on a cell containing a heap, it uses the center object of a heap O_{center} and the Euclidean distance $d(O_t,O_{center})$ to compare it with its carried object $O_{carried}$. If this distance is lower than a constant value, the ant puts $O_{carried}$ on the heap else it moves again on the grid. A carried object is always dropped after a given iteration number. When an ant visits a heap and if it doesn't carry any object, it searches the object O_{dissim} which is the farthest from the center of this heap. If $d(O_{center},O_{dissim}) \geq T_{remove}$, it catches O_{dissim} with a given probability. If an ant moves on a cell containing only one object, this one is taken under a given probability.
This first step allows to build a first set of clusters but has the common disadvantages of ants algorithms : the convergence difficulties. Indeed, isolated objects can be let on the grid even after a great number of iterations. Moreover, too much clusters are generally produced.

The second step aims at improve the result produced by the first step using k-Means. This algorithm needs initial partitions of data given by the previous step. This step allows to relevantly correct and complete ants initial work. But it generally produces a too big number of clusters. That's why ants are used again in a third step in order to reduce the number of clusters by clusters fusion.

In the third step, an ant carries a heap instead of an object. In order to do it, heaps are represented by their center object. So, an ant carries a heap $H_{carried}$ and the Euclidean distance used is $d(O_{center}(H_{carried}),O_{center}(H_{visited}))$. If this distance is lower than a constant value then the two heaps are fused. As in the first step, a heap is always dropped after a given number of iterations. That's why some heaps are not necessarily fused even if they should be.

In order to correct and complete the previous result, k-Means is used again in a last step. This time, the algorithm works only on heaps (they are seen as data) and tries to cluster heaps.

This algorithm is proved to give good results on static data (results are given in (Monmarché et al. 1999)). AntClass is not initially intended to work with dynamic data or data streams. In (Coma 2002), the ants part of AntClass has been evaluated on dynamic data. The conclusion is that data evolution can be taken into account but convergence problems already mentioned are reinforced. Nevertheless, the idea used in AntClass consisting in coupling ants with a second algorithm seems to be a good way to solve this convergence problem. Unfortunately, AntClass authors have chosen k-Means. This algorithm gives good results but, even if it is incremental, it is not dynamic. That is to say data processed by this algorithm must be defined at the beginning. New data can only be processed by placing them in existing clusters. It does not support clusters evolution. Moreover, with continuously evolving data, the second algorithm should be launched frequently (not only twice). It is very difficult to find criteria in order to determine when using it and how to combine new results with old ones.

As a consequence, in our approach, we have chosen to couple adapted ants algorithm of AntClass with a more dynamic layer based on cluster agents presented in the next section.

OUR APPROACH

This section presents a multiagent architecture for agents clustering. This architecture couples ants with a second agents layer called cluster agents. This architecture allows to cluster evolving data coming from agents properties included in an observed population. From an agent point of view, this architecture must provide an abstraction of the observed agents population in terms of data allowing to perform the clustering. It also must provide a "real time" representation of the clusters corresponding to the current global state of the observed population. From the clustering point of view, this architecture must allow to manage data which can be modified, disappear or be added over time.

Figure 1 shows the proposed architecture which consists in five main components :

- the observed agents population
- the Observer agent
- Ants
- Cluster agents
- The objects list

The next sections describe the different components of this architecture. The last one shows how this proposal can answer to requirements associated to data stream clustering (presented previously).

Observed Agents Population

The agents of this population can be any kind of agent. The only hypothesis is that they must provide a set of measures to be observed. These measures are supposed to summarize a

state corresponding to a certain point of view on the agents of the observed population.

Observer Agent

The goal of this agent is to scrutinize the agents population to observe in order to build a set of data to cluster. A data (an object) is made of the set of measures provided by each agent to observe. These data are put together in an objects list. Each time an agent is created, the observer agent must create a new data containing its measures and add it to the objects list. If this agent's measures change, the observer agent must also take this evolution into account by modifying the corresponding object in the objects list. The observer agent represents an information gateway between the observed agent layer and the clustering layer.

Ants

They are central in the clustering process. Their behavior is adapted from those of AntClass. The main difference concerns their motion. In AntClass, data are put on a grid, ants move on this grid that's why clusters are also built on this grid. In our architecture, data are put in an objects list and clusters are represented by cluster agents. That's why ants move along the list to pick up an object or move inside the cluster agents organization to add an object to a cluster.

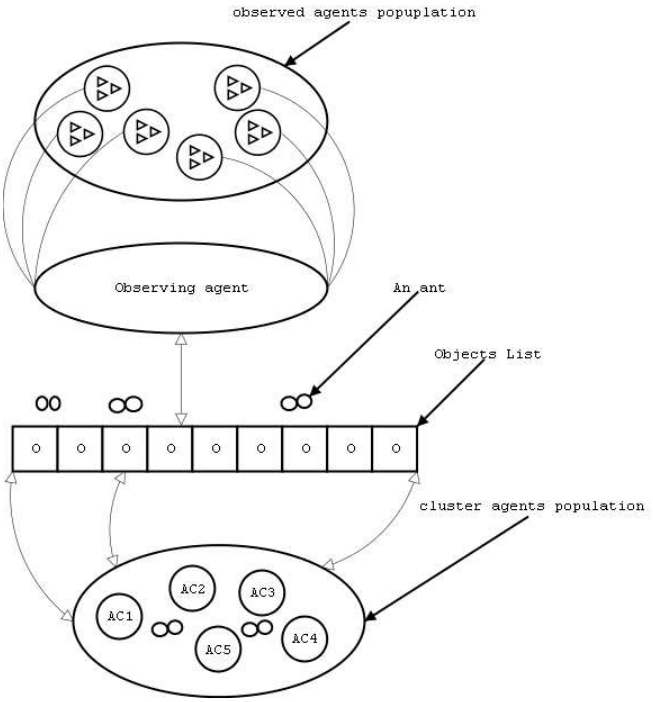

Figure 1 : System Architecture

In "AntClass", it is possible that ants don't move on the good heap for their object (because of random). Therefore, an ant drops automatically its object in an empty cell after a given number of iterations. It explains that some data are not always placed in the good cluster. That explains partially the convergence problem. In order to reduce this phenomenon, our ants must visit all existing cluster agents before dropping their object. Moreover, dropping an object in an empty cell consists here in creating a new cluster agent containing this object. Ants move at random among cluster agents.

Ants' Motion

Each ant begins by picking up a random object in the list. Then the ant moves inside the cluster agents population in order to try to drop this object in a "good cluster" agent. As long as an ant has an object, it moves only inside the cluster agents population. After having visited the different cluster agents, if ant has not already dropped its object, it creates a new cluster agent.

When an ant visits a cluster agent, it verifies if it is possible to add its object to the corresponding cluster. This verification is made using a distance measure like in AntClass. After its carried object has been dropped, the ant can come back to the objects lists in order to choose a new object to cluster.

Cluster Agents

A cluster agent represents a cluster (a set of objects) and its goal is to verify the relevance of its cluster. First of all, when an ant drops its carried object in the cluster of a cluster agent CA, CA must update the value of the cluster center. Moreover, it has to detect objects (data) evolution. If an object becomes too different from the cluster center, the cluster agent must reject it in the objects list and update the cluster center again. This mechanism allows to set up a kind of competition between ants and cluster agents. This competition must help to improve the global convergence of the clustering.

To evaluate if an object is too dissimilar from the cluster center, the cluster agent uses a distance measure and a dissimilarity threshold like in k-Means used by AntClass. The purpose is to keep a "good cluster", that is to say it does not include too dissimilar objects. Let D_{min} be the minimum distance between the cluster center and a cluster object. The cluster agent must always verify that :

$\forall i,\ d(o_i,o_c) < D_{min}$ with

o_i : the i^{th} cluster object

o_c: the cluster center.

Cluster Agent Birth and Death

When an ant can not drop its object in existing clusters, it must create a new cluster containing this object. This new cluster agent has a limited life time. Indeed, if it doesn't receive any new object during a too long period or if it rejects too much objects during the same period, the cardinal of its corresponding cluster becomes too small. In this case, the cluster agent dies. This allows to take into account data evolution making some clusters non relevant anymore. Moreover, just before dying, a cluster agent rejects the objects of its cluster in the objects list. This mechanism allows ants to build new clusters more representative of the new data or the new state of data induced by the evolution of the observed agents population.

The Objects List

The objects list allows to store non clustered objects. Two kinds of objects can be found in this list :

- objects corresponding to new agents which have never been clustered.
- objects which have been rejected from their initial cluster by cluster agents.

This structure allows us to manage both new data arrival and data evolution.

Discussion and Analysis

We have defined a multi-agent clustering method allowing to cluster evolving data corresponding to properties of agents belonging to an observed population. Clusters are represented by cluster agents which evolve with data, taking into account their evolution. Ants are used to cluster new data or move rejected data in new clusters. Cluster agents and ants are in a competition process from which can emerge a relevant clustering of the current data representing the global state of the observed agents population.

In (Barbarà 2002), Barbarà has defined three requirements for data streams clustering algorithms (see section on this subject). Our problem is not so far from the data streams clustering problem so it is interesting to see how our approach meet these requirements.

The first one is the compactness of data and clusters representation. For data, it depends entirely on the kind of properties observed in agents. Clusters are represented by their center point. Moreover, their behavior ensures a controlled increase of their number. Indeed, if the cardinal of their corresponding cluster stays too small, they die.

The second one is a fast incremental processing of new data points. Our own problem requires more than that : a fast incremental processing of new data points or old data points modifications. The task of ants ensures an incremental processing, taking easily new data into account. Data evolution is detected very quickly thanks to the data reject mechanism of cluster agents. Nevertheless, in order these evolutions to be processed quickly, it requires that ants place the rejected data in new clusters quickly too. This is not always ensured. However, as ants do not move on a grid but on an objects list, we can think that rejected objects will be found by ants more quickly than in AntClass.

The last requirement is the identification of outliers. An outlier is a new data (or in our context an evolving data) which can not be placed in any existing cluster. These data are processed by ants when trying to put their object in a cluster. In order to achieve this task, they visit each cluster until they have found a satisfying cluster. If it is not the case, they create a new cluster agent. In that case, the object corresponds to an outlier. These outliers can be of two kinds : new data never clustered or rejected data clustered before. Indeed, if a cluster agent does not include enough data during a too long period, it dies. This corresponds to a part of outliers detection.

CONCLUSION

In this paper, a multi-agent architecture for agents clustering has been presented. This architecture is based on an ants algorithm coupled with a cluster agents layer helping ants for the global convergence of the clustering method. After having presented the needs for agents clustering and the associated constraints, we have presented a review of clustering methods suitable to this context. Methods based on ants appear to be the most interesting ones. That's why, as in AntClass, we have chosen to couple an ants algorithm with a second agents layer.
This architecture is currently being implemented and the validation of the clustering method is beginning.

This architecture is currently used in a preventive monitoring multi-agent system. project (Boukachour et al. 2002). The goal of this kind of system is to offer to managers the more relevant information as possible about the current situation in order to take good decisions. In this context, the agents population to observe represents different pieces of information about the analyzed situation. Information are obtained by interactions with users, by queries on multiple databases and by sensors. This explains that this agents population may contain redundant or useless pieces of information. As the situation evolves, the corresponding agents modify themselves. We try to use our dynamic clustering architecture in this context in order to obtain a synthesized view of the important points of the situation in order to be able to compare it with similar older situations. This synthesized view must allow to focus on only highly relevant information so as to find the best similar situations which will provide the basis for the decisions to take.

In the future, we think to complete the cluster agents behavior so as to allow clusters fusion or division. Moreover, their reject mechanism could be perhaps improved using fractal dimension of clusters as it is proposed in (Barbarà and Chen 2000) (instead of the distance measure from the cluster center as in k-Means). Indeed, in the Fractal Clustering Algorithm, each cluster has a fractal dimension which is updated for each new data. A new data is put in the cluster whose variation of this fractal dimension is the smallest. This algorithm seems to give good results for data streams. As a consequence, it should improve clusters agents work.

REFERENCES

Barbarà D. and Chen P 2000. "Using the fractal dimension to cluster data sets" In *Proceedings of the* ACM-*SIGKDD International Conference on Knowledge and Data Mining*, Boston.

Barbarà D. 2002. "Requirements for Clustering Data Streams" *SIGKDD Explorations 3*, No. 2, 23-27.

Boukachour H., Simon G., Coletta M., Galinho T., Person P. and Serin F. 2002. "Système de veille préventive : modélisation par organisations d' agents" In *Proceedings of Ingénierie des Connaissances (IC'2002)*, INSA de Rouen, France, 187-195.

Bertelle C., Olivier D., Jay V., Tranouez P. and Cardon A. 2000. "A multi-agent system integrating vortex methods for fluid flow computation" In *16th IMACS congress*, Lausanne, Switzerland.

Coma R. 2002. " Dynamic clustering for a multi-agent system". *DEA report*, LIH,.Le Havre, France.

Deneubourg J.L. , Goss S., Franks N., Sendova-Franks A., Detrain C. and Chretien L. 1990. "The dynamics of collective sorting: robot-like ant and ant-like robots" in Meyer and Wilson, 356-365.

Ferber J. and Gutknecht O. 1998. "Aaladin : a meta-model for the analysis and design of organisations in multi-agent systems" In *Proceedings of ICMAS'98*, IEEE, 155-176.

Ferber J. and Jacopin E. 1991. "The framework of Eco Problem Solving " In *Proceedings of the 2nd European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW-91)"*, Elsevier North-Holland, 103-114.

Guha S. ; Mishra N.; Mortwani R.: and O' Callaghan L. 2000. "Clustering data streams" In *Proceedings of the Annual Symposium on Foundations of Computer Science*.

Jain A.K. and Dubes R.C. 1988. *Algorithms for Clustering Data* Prentice-Hall Advanced Reference Series.

Kaufman L. And Rousseeuw P.J. 1990. *Finding groups in data : an introduction to cluster analysis*. John Wiley and sons, New York.

Lumer E.D. and Faieta B. 1994. "Diversity and Adaptation in Populations of Clustering Ants" In *proceedings of the third International Conference on Simulation of Adaptative Behavior : From Animals to Animats 3 (SAB'94)*, D. Cliff, P. Husbands, J.A. Meyer, S.W. Wilson (Eds), MIT-Press, 501-508.

Marcenac P. 1997. "Modélisation de systèmes complexes par agents" *Techniques et Sciences Informatiques* 16, No. 8, Hermès, 1013-1038.

Monmarché N. 2000. "Algorithmes de fourmis artificielles : applications à la classification et à l' optimisation" *PhD Thesis*, Tours, France.

Monmarché N.; Slimane M. and Venturini G. 1999. "AntClass : discovery of clusters in numeric data by an hybridization of an ant colony with the Kmeans algorithm", Research report 213, Laboratoire d'Informatique de l'Université de Tours, E3i Tours.