VERS UNE ARCHITECTURE MULTI-AGENTS POUR DU CLUSTERING DYNAMIQUE D'UNE POPULATION D'AGENTS

Roland Coma,
Doctorant en informatique spécialité intelligence artificielle
roland.coma@univ-lehavre.fr, +33 2 32

Adresse professionnelle
Laboratoire d'Informatique du Havre (LIH)
25 rue Philippe Lebon BP 540 ★ 76058 Le Havre

Résumé : Les travaux présentés dans cet article concernent l'analyse de populations d'agents par clustering dynamique. Les agents sont des entités dynamiques, et leurs propriétés évoluent sans cesse. La principale difficulté est de trouver une technique de clustering applicable à des agents et capable de réaliser un clustering au fur et à mesure de l'arrivée des agents (aspect incrémental), mais également capable de prendre en compte les modifications des données déjà clusterisées (aspect dynamique des clusters).

Summary : The works presented in this article concern the analysis of populations of agents by dynamic clustering. The agents are dynamic entities, and thus their properties evolve ceaselessly. The main difficulty is to find a technique of clustering applicable to agents and capable of realizing a clustering according to the arrival of the agents (incrémental aspect), but also able to take into account the modifications of the data already clustered (dynamic aspect of clusters).

Mots clé : clustering dynamique, système multi-agents.

Vers une architecture multi-agents pour du clustering dynamique d'une population d'agents

L'objectif est de mettre en oeuvre des techniques de clustering dynamiques afin de mettre en évidence l'organisation d'une population d'agents par des groupes d'agents jugés similaires. Une des spécificités du problème posé est la dynamique d'un système multi-agents. Cette dynamique se traduit par deux points importants: l'évolution des agents et l'arrivée de nouveaux agents. Les algorithmes de clustering classiques ne prennent en compte que des données statiques, même si certains (K-means séquentiel, FC) intègrent l'aspect incrémental. Nous cherchons donc des algorithmes de clustering classiques qui peuvent être utilisés ou adaptées pour résoudre ce problème. Ce problème est apparu dans le cadre du développement d'un outil de veille préventive basé sur un système multi-agents Boukachour et al. (2002). Après avoir développé une couche fondée sur le paradigme des agents aspectuels permettant de représenter les différents aspects de la situation, décrit par **Error! Reference source not found.**Durant (1999), nous cherchons à classer les aspects de la situation courante afin d'analyser et de prédire la suite des événements. Pour cela, nous avons besoin d'une technique de clustering, mais le problème est que les données fournies par la couche inférieure sont dynamiques. Dans un premier temps, nous allons généraliser ce problème en considérant une population d'agents quelconque. Puis, nous décrirons plus en détail l'aspect dynamique du problème. Pour finir, nous proposerons une architecture multi-agents pour résoudre ce problème, ainsi qu'un protocole de test permettant de valider cette architecture.

1 - GENERALISATION

Dans le domaine du clustering classique, on manipule des objets et des attributs. Les objets sont des vecteurs d'attributs et les attributs sont les données du problème. L'objectif est d'obtenir des regroupements d'objets en clusters. Un cluster est un groupe d'objets jugés similaires. La quasi totalité des algorithmes de clustering classiques utilisent un calcul de distance pour évaluer la proximité ou la dissimilarité entre les différents objets qui deviennent alors des points d'un espace à n dimensions, n étant le nombre d'attributs des objets. Dans cet article, nous nous intéressons au clustering d'agents. Quels vont donc être dans ce contexte les données de l'algorithme de clustering? On suppose que chaque agent de la population d'agents à catégoriser met à disposition un ensemble de caractéristiques permettant de décrire son état courant. Par conséquent, par rapport au clustering classique, les objets sont les ensembles de caractéristiques des agents, leurs attributs sont les différentes caractéristiques de chaque agent, ce qui permet d'utiliser les mêmes calculs de distance que dans le clustering classique. Au sein d'une population d'agents, les caractéristiques de chaque agent évoluent sans cesse : c'est le problème de l'évolution des données. De plus de nouveaux agents peuvent apparaître avec leurs propres caractéristiques : c'est le problème de l'arrivée de nouvelles données. Le but est que le regroupement des agents en clusters soit pertinent à chaque instant. Les caractéristiques de ces agents sont les données du clustering, l'évolution et l'arrivée de ces nouveaux agents va donc modifier les regroupements effectués précédemment.

1.1 - Arrivée de nouvelles données

Lorsque les données à clusteriser arrivent au fur et à mesure on peut parler de clustering dynamique, dans le sens où les données arrivent dynamiquement : il s'agit d'un clustering incrémental. Il est à noter que certains algorithmes comme celui proposé par Barbarà (2000) gèrent des flux de données, le problème de l'arrivée de nouvelles données est le même pour nous. Dans le clustering classique le cardinal de l'ensemble d'objets est fixe, mais dans notre cas de nouveaux agents peuvent apparaître ou disparaître. En d'autres termes, $card(A)$ n'est pas fixe et évolue au cours du temps. On peut donc considérer que $card(A)=f(t)$.

1.2 - Évolution des données

Afin de mieux comprendre ce problème, prenons l'exemple d'une personne qui veut trier un tas de papiers administratifs, sans savoir à l'avance quelles sont les différents types de document. Cette personne doit alors examiner les documents et faire des tas. Supposons maintenant que les documents se modifient même une fois qu'ils sont déposés dans un des tas. Ce type de problème nous montre bien un aspect dynamique au sens où les données elles mêmes sont dynamiques.

Dans le clustering classique, les objets sont représentés par des vecteurs d'attributs (valeurs), mais les valeurs de ces attributs sont statiques. Par contre,

dans notre cas, un objet est l'ensemble des caractéristiques d'un agent et ses attributs sont les différentes caractéristiques de cet agent. De plus les valeurs de ces attributs peuvent varier au cours du temps. On a donc un ensemble d'agents A et un ensemble de caractéristiques C où :
-Chaque agent a le même nombre de caractéristiques n donc $card(C)$ est fixe,
-Chaque agent est représenté par un vecteur de caractéristiques donc $\forall a \in A$ on a $a \in C^n$ ou encore $a=(c_1,c_2,\ldots,c_n)$, $\forall c_i \in C$,
On peut donc de représenter les valeurs des attributs comme des fonctions du temps ainsi :
$a=(c_1(t),c_2(t),\ldots,c_n(t))$ (1)

Remarque **:** Le terme de "clustering dynamique" est ambiguë car il peut signifier que les données arrivent dynamiquement et nous sommes alors dans le cas d'un clustering d'un flux de données, ou encore que les données à clusteriser peuvent être modifiées au cours du temps même si elles sont déjà clusterisées. Pour un clustering d'agent, nous avons les deux problèmes en même temps.

2 - METHODES DE CLUSTERING CLASSIQUES

Le clustering Kamber et Han (2001,chap 8) est un mécanisme permettant de classer **intelligemment** des données dans le but de prédire des phénomènes, d'analyser des données expérimentales ou encore de travailler sur des classes de données plutôt que sur des données en très grand nombre. L'analyse par classes est une méthode importante dans l'apprentissage humain. Les enfants commencent par reconnaître les objets en les assimilant à des groupes. Le clustering permet de repérer les régions les plus denses ou encore de reconnaître des formes et de faire des liens pertinents entre les différents objets en fonction de leur attributs. Certains mécanismes font appel à des techniques de classification comme l'étude de marché des consommateurs ou en biologie avec la classification des espèces animales ou végétales.

Les algorithmes utilisent des mesures de distance pour évaluer la similarité entre les objets. La plus connue est la distance euclidienne.

2.1 - Les catégories de clustering

Il existe de nombreux algorithmes de clustering qui peuvent être classés dans les catégories suivantes :
–Méthodes par partitions,
–Méthodes hiérarchiques,
–Méthodes basées sur la densité,
–Méthodes basées sur un modèle,
–Méthodes basées sur les fourmis.

Certains algorithmes intègrent les idées de plusieurs catégories, il n'est donc pas simple de les classer parmi une de ces catégories.

Bilan : Il existe de nombreuses catégories de clustering. Malheureusement les méthodes plutôt prévues pour des données statiques ne sont pas bien adaptées ou difficilement adaptables aux problèmes posés par les aspects dynamiques de l'analyse d'une population d'agents. Seul les algorithmes fourmis semblent prometteurs. Nous nous sommes donc plus particulièrement intéressés à ce type d'algorithmes, et plus particulièrement à **ANTCLASS** proposé par Monmarché,Sliman et Venturini (1999) présenté dans la section suivante.

3 - ANTCLASS

Jusqu'ici, nous avons cherché des algorithmes de clustering classique qui pourraient répondre au problème posé par l'aspect dynamique des données. Nous nous sommes rendus compte que l'on pouvait examiner d'autres approches comme l'hybridation de plusieurs algorithmes. Nous nous sommes donc intéressés à **AntClass** qui mélange deux algorithmes dont un algorithme de type fourmis. Ici, nous allons uniquement présenter le principe de fonctionnement de cet algorithme.

3.1 - Principe

Cet algorithme se décompose en quatre étapes :
–un algorithme de type fourmis classique pour constituer les classes initiales,
–un algorithme ACM (Algorithme des Centres Mobils) pour corriger les erreurs du premier,
–un algorithme fourmis sur les classes (clusters) proposés par l'ACM,
–un algorithme ACM pour corriger les erreurs subsistantes après l'étape précédente.

La première étape fournit une classification pertinente mais avec un certain nombre d'erreurs et d'**objets libres** dû au effet du problème de convergence des algorithmes de fourmis de classification. La deuxième permet de corriger ces erreurs mais laisse un trop grand nombre de classes. La troisième qui est la même que la première mais appliquée sur les classes permet de réduire le nombre de classes en les regroupant. Enfin, la quatrième est là pour corriger une nouvelle fois les erreurs de la précédente.

3.2 - Discussion

Rappelons que dans notre problème, les données (les caractéristiques des agents) évoluent et de nouvelles données peuvent apparaître. Ce qui n'est à priori pas le cas pour AntClass.

Avantages :
–Cela paraît intéressant de combiner des algorithmes de clustering pour atténuer les inconvénients rencontrés,
–l'utilisation de l'algorithme des fourmis paraît répondre aux contraintes dynamiques de nos données. Les fourmis se déplaçant en permanence, il est possible de prendre en compte l'arrivée de nouveaux objets. Il suffit de les placer aléatoirement sur la grille. On peut penser également que le problème de l'évolution des données peut être directement pris en compte par cet algorithme. Il suffit qu'une fourmi détecte que l'objet qui a évolué est devenu trop éloigné du centre du tas auquel il appartient (trop dissimilaire). Un autre algorithme qui interviendrait à intervalle de temps régulier corrigerait les erreurs et se chargerait des nouvelles données.

Inconvénients :
Comment déterminer les critères d'arrêt et de reprise des différents algorithmes ?
Quand appliquer le second algorithme (non fourmis) ?

Les expérimentations montrent que les algorithmes fourmis fournissent une base pour répondre aux contraintes dynamiques. Les problèmes de convergence de ce type d'algorithmes obligent à envisager une combinaison avec un autre algorithme. Les auteurs d'«AntClass» ont choisit l'ACM qui présente l'inconvénient d'être statique. Nous nous orientons plutôt vers une seconde couche plus dynamique reposant sur des agents présentée dans la section suivante.

4 - ARCHITECTURE PROPOSEE

Le but de l'architecture proposée est de mettre en oeuvre un clustering dynamique tenant compte des modifications des caractéristiques des agents au cours du temps et de l'arrivée de nouveaux agents. Cette architecture repose d'une part sur une version modifiée de l'algorithme fourmis de **AntClass** et d'autre part sur une seconde couche d'agents interagissant avec les fourmis. Le rôle de cette couche d'agents est d'améliorer la convergence des fourmis.

Nous présentons le rôle et le fonctionnement des différents composants de l'architecture. Cette architecture multi-agents permettant de gérer de la même façon les nouveaux arrivants et les modifications des anciens. Nous allons donc décrire précisément les différents composants schématisés dans la figure suivante (cf. fig. 1) nous allons également décrire un protocole de tests pour valider notre architecture.

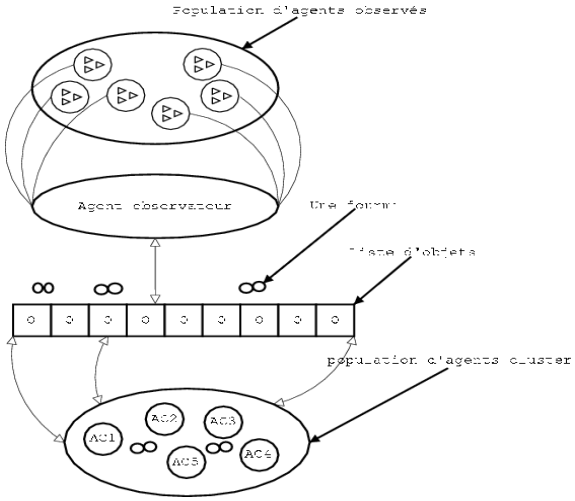

Figure 1: Le principe de notre architecture

Les différents composants de cette architecture sont :
–l'agent observateur,
–les fourmis,
–les agents cluster,
–la liste d'objets,

Une des contraintes ayant guidé la conception de cette architecture est de rendre indépendant le fonctionnement de l'organisation d'agents de bas niveau analysée et celui de la couche de clustering.

4.1 - L'agent Observateur

L'agent observateur scrute en permanence la couche d'agents de bas niveau à analyser. Lorsqu'il détecte un nouvel agent, il crée un objet contenant uniquement ses caractéristiques observales et le place dans une liste. Si les caractéristiques de cet agent changent, l'agent observateur est capable de le détecter et de modifier en conséquence l'objet qu'il a créé. C'est la passerelle d'information entre la couche d'agents à analyser et la couche qui réalise cette analyse par clustering.

Dans notre plate-forme, il gère un fichier scénario et se charge de mettre à jour les objets ou de créer de nouveaux objets. Lorsque l'utilisateur demande une classification, il génère un fichier ou une base de donnée contenant l'état des objets au moment où l'utilisateur a demandé sa classification. Cette étape est nécessaire pour notre protocole de tests que nous détaillerons plus tard.

4.2 - Les fourmis

Leur rôle est capital. Elles font des va-et-viens entre la liste des objets et les agents cluster. Pendant ces allers et retours, elles peuvent déposer des objets dans les agents cluster et créer de nouveaux agents cluster.

Toutefois, il est à noter que ces fourmis ont surtout un rôle transport afin de soulager le travail des agents cluster.

Ce sont elles qui réalisent le clustering. Leur comportement s'inspire de celui proposé dans **AntClass**. La principale différence réside dans le fait qu'elles ne se déplacent plus sur une grille, mais sur une liste d'objets.

Dans **AntClass**, les fourmis se déplacent aléatoirement, il est donc possible qu'elles ne passent pas sur les bons tas au bon moment. Pour limiter ce phénomène, nous obligeons les fourmis à parcourir tous les clusters avant de créer un nouveau cluster. Cependant, pour éviter que les fourmis se dirigent au même endroit en même temps, ce parcours est aléatoire : à chaque visite d'un cluster, la fourmis le marque comme visité. De plus, en procédant ainsi le nombre de clusters à un instant donné sera borné par le nombre de cluster réel. Autrement dit, nous cherchons à limiter le nombre de tas pouvant être fusionnés.

Déplacement d'une fourmi de la liste vers les agents cluster
Lorsqu'une fourmi arrive sur la liste, elle ramasse aléatoirement un objet dans la liste. Puis elle se dirige vers les agents cluster pour essayer de déposer l'objet dans un des clusters. Une fourmi qui va sur la liste ne doit pas porter d'objet.

Déplacement d'une fourmi des agents cluster vers la liste
Lorsqu'une fourmi arrive sur la liste des agents cluster, elle choisit au hasard un agent cluster et vérifie si elle peut y déposer l'objet qu'elle transporte. Si elle ne peut pas, elle en choisit un autre, et ainsi de suite. Si elle n'a pas déposé son objet dans un des agents cluster, elle crée un nouvel agent cluster contenant cet objet avant de retourner sur la liste.

4.3 - Les agents cluster

Un agent cluster représente un cluster. Son rôle est de vérifier la pertinence de son cluster. En particulier, il doit détecter les modifications des données appartenant à son cluster. Si un objet devient trop éloigné du centre du cluster, l'agent le rejette immédiatement dans la liste des objets, où les fourmis peuvent le ramasser comme s'il s'agissait d'un nouvel objet. Le but est de détecter les modification de données aussitôt, c'est pourquoi nous faisons appelle à des fourmis pour transporter les objets et ne pas surcharger le travail des agents cluster. On peut considérer qu'un agent cluster a un comportement régi par un automate simple à deux états.

–état 1 : il vérifie que tous les objets qu'il contient sont similaires,
–état 2 : il recalcule son centre, après avoir rejeté ou reçu un objet.

Pour juger si un objet est dissimilaire, on utilise une distance et un seuil de similarité, tout comme dans Monmarché,sliman et Venturini (1999). Le but des agents clusters est de maintenir en permanence un **cluster sain**, c'est-à-dire qui ne contient pas d'objet trop dissimilaire. Pour cela, nous définissons un paramètre D_{min} qui est la distance minimum entre le centre du cluster et un objet du cluster. La principale action de cet agent est donc de vérifier en permanence si :
$\forall i,\ d(o_i,o_c)<D_{min}$ avec
o_i : le ième objet du cluster
o_c : le centre du cluster

L'agent cluster est aussi capable de recalculer son centre s'il reçoit un nouvel objet ou s'il rejette un objet. Dans le cas où il rejette un objet, il le place dans la liste des objets. Pour recevoir un objet, il doit d'abord accueillir une fourmi puis la laisser vérifier si elle peut déposer l'objet qu'elle transporte.

Naissance d'un agent cluster
Lorsqu'une fourmi n'a pas déposé un objet après avoir parcouru toute la liste des agents cluster, elle doit créer un nouvel agent cluster pour y déposer son objet. Cet agent cluster n'a alors qu'une durée de vie limitée en nombre d'itérations. S'il reçoit au moins deux autres objets dans cet intervalle de temps, l'agent cluster peut alors calculer le nouveau centre de gravité de son cluster. Ainsi nous évitons le problème des objets isolés.

Mort d'un agent cluster
Un agent cluster de moins de trois objets à une durée de vie limité. Si pendant une durée fixée en paramètre, il ne reçoit aucun autre objet, il disparaît en rejetant avant les objets de son cluster dans la liste.

4.4 - La liste

La liste des objets permet de stocker les objets non encore classés. Les fourmis peuvent se déplacer dessus et prendre un objet. Les agents cluster rejettent également dans cette liste les objets jugés trop dissimilaires. Elle est mise à jour par l'agent observateur en cas d'arrivée de nouveaux agents de bas niveau. Cette structure permet de gérer à la fois l'arrivée de nouvelles données et la modification de données. Lorsque l'agent observateur détecte un nouvel agent, il crée un objet qu'il place dans la liste d'objet. Lorsqu'un agent cluster rejette un objet, il le place également dans la liste d'objets. De plus cette

liste est visité par les fourmis. Il est donc clair, qu'elle doit être accessible par ces trois agents.

4.5 - Le protocole de tests

Pour tester notre architecture nous proposons un protocole permettant de comparer les résultats obtenus par notre architecture à des résultats obtenus par des méthodes de clustering classiques. Les données traitées par notre architecture sont dynamiques, il est donc nécessaire de les figer à l'instant où l'utilisateur demande un résultat.

Le fichier généré à cet instant par l'agent observateur va alors servir de base pour appliquer un clustering classique. Ce résultat sera ensuite comparé à celui fourni par notre architecture. Ainsi, nous pourrons déterminer si notre architecture donne des résultats cohérents par rapport au clustering classique sur des données statiques.

4.6 - Bilan

L'arrivée de nouvelles données est prise en charge par cette architecture. Lorsqu'un nouvel agent apparaît au sein de la population d'agents de bas niveau, l'agent observateur copie ses caractéristiques dans un objet et envoie cet objet dans la liste où il sera traité comme les autres objets. De plus, lorsqu'un objet subit une modification qui le rend trop dissimilaire au sein de son cluster, l'agent cluster repère instantanément cet objet et le rejette immédiatement dans la liste où là encore, il sera traité comme les autres objets.

5 - CONCLUSION ET PERSPECTIVES

Nous avons réalisé ce travail, afin de développer un système multi-agents représentant une situation courante grâce au paradigme des agents aspectuels. Nous cherchons un moyen de regrouper les agents en fonction de leurs caractéristiques, pour pouvoir analyser cette situation et éventuellement prédire les événements suivants. Il s'agit donc d'un clustering mais avec des données dynamiques d'où le terme de «clustering dynamique». Nous voulons également étendre cet notion de clustering dynamique à d'autres systèmes multi-agents.

Le premier intérêt de ce travail est donc de définir la notion de clustering dynamique pour un système multi-agents : de nouveaux agents peuvent apparaître ou disparaître et les agents peuvent évoluer. Dans un premier temps, nous avons donc fourni une formalisation des données de ce problème. Cette formalisation nous à permis de mieux cerner les contraintes que nous avons à prendre en compte.

Dans le but de voir s'il était possible d'utiliser ou d'adapter un algorithme existant nous avons classé différents algorithmes de clustering classiques en fonction des contraintes de ce problème. Cependant, nous nous sommes aperçu que la plupart de ces algorithmes ne sont pas adaptés aux contraintes de notre problème. Seul les algorithmes fourmis, semblent pouvoir résoudre en partie notre problème car ils peuvent s'adapter à l'arrivée des nouvelles données mais aussi à l'évolution des données.

Nous avons donc étudié «AntClass», un algorithme de clustering reposant en partie sur un algorithme de type fourmis. Puis nous avons expérimenté cet algorithme sur des données dynamiques. Les résultats nous ont montrés que cet algorithme contient la base pour résoudre en partie notre problème.

Nous avons donc choisi de combiner un algorithme de type fourmi avec un système multi-agents pour exploiter le caractère dynamique de l'algorithme fourmi, tout en assurant une bonne convergence. Nous proposons donc une architecture permettant de mettre en oeuvre un clustering dynamique d'une population d'agents. Nous donnons également les spécifications de l'architecture et d'un protocole de tests permettant une validation future par rapport des techniques de clustering classiques.

Les tests sont actuellement en cours et les premiers résultats montrent que les clusters sont purs, mais souvent trop nombreux. Le comportement des agents clusters doit donc être complété afin d'autoriser par exemple des fusions permettant de limiter le nombre de clusters produits.

Les travaux de D. Servat porte sur le regroupement de gôutes d'eau. Ces travaux sont similaires aux notres. Néanmoins, il s'agit d'un domaine bien précis le ruissellement des goûte d'eau, Servat, Perrier et Treuil (1998a, 1998b). De plus, les regroupements sont effectués par les agents de bas niveau, alors que nous avons fait le choix de ne pas influencer le travail de ces agents car nous avons voulu que notre architecture puisse s'appliquer à tous les systèmes multi-agents. Cependant, les performances de cet algorithme pourraient être intéressantes à comparer avec les performances de notre architecture.

Remerciements

L'implémentation est réalisé par M. coletta et G. Simon.

Je tiens à remercier tous les membres de l'équipe de recherche **Agents et Risques Majeurs** du **Laboratoire d'Informatique du Havre**, dans laquelle ont été effectués ces travaux.

Je salue également **G. Simon**, **M. Coletta** et **B. Mermet** pour l'aide qu'ils m'ont apporté à la rédaction de cet article.

RÉFÉRENCES

Barbará, D. "Using the fractal dimension to cluster datasets". Proc. of the International Conference in Knowledge Discovery and Data Mining (ACM-SIGKDD) (2000).

H. Boukachour, G. Simon, M. Coletta, T. Galinho, P. Person, F. Serin, ''Système de veille préventive : modélisation par orga-nisations d'agents'', IC'2002, 13eme journées fran-cophones d'Ingéniérie des Connaissances, Rouen, 28-30 mai 2002, p 187-195.

Durant, S. ''Représentation de points de vues multiples dans une situation d'urgence: modélisation par organisation d'agents''. PhD thesis, Université du Havre (1999).

Kamber M., Han J., "Data mining concepts and techniques", Morgan Kaufmann publishers, chapitre 8, (2001).

McKenna, S., Gong, S.,. "Modelling facial colour and identity with gaussian mixtures". Proc. of the Pattern Recognition, volume 31, pages 1883–1892 (1998).

Monmarché, N., Slimane, M., G. Venturini, "On improving clustering in numerical databases with artificial ants". Proc. of the Floreano, D., Nicoud, J.,, editors, 5th European Conference on Artificial Life (ECAL'99), Lecture Notes in Artificial Intelligence, volume 1674, pages 626–635, Swiss Federal Institute of Technology, Lausanne, Switzerland. Springer-Verla.

R. Coma, G. Simon, M. Coletta, ''A multi-agent architecture for agents clustering'',
ABS'2003, 4th Workshop on Agent-Based Simulation, p 15-20, Montpellier, April 28-30 2003.

D. Servat, E. Perrier, J.-P. Treuil, A. Drogoul, "Towards virtual experiment laboratories: How multi-agent simulations can cope with multiple scales of analysis and viewpoints". Lecture Notes in Computer Science, Vol. 1434, p. 205–? ? (1998a).

D. Servat, E. Perrier, J.-P. Treuil, A. Drogoul , "When agents emerge from agents: Introducing multi-scale viewpoints in multi-agent simulations". Proc. of the MABS, pages 183–198 (1998b).