Applications of Self-Organising Multi-Agent Systems: An Initial Framework for Comparison

Carole Bernon
IRIT, University Paul Sabatier
31062 Toulouse Cedex 09, France
E-mail: bernon@irit.fr, http://www.irit.fr/SMAC

Vincent Chevrier
LORIA
BP 239
54506 Vandoeuvre Les Nancy Cedex, France
E-mail: chevrier@loria.fr, http://www.loria.fr/~chevrier/

Vincent Hilaire
SeT, UTBM
90010 Belfort Cedex, France
E-mail: Vincent.Hilaire@utbm.fr, http://set.utbm.fr/membres/hilaire/index.php

Paul Marrow
Pervasive ICT Research Centre, BT Group plc
Orion 1 PP 12, Adastral Park, Ipswich IP5 3RE, United Kingdom
E-mail: paul.marrow@bt.com, http://www.btplc.com/

Keywords: multi-agent system, self-organisation, applications, software

Received: April 18, 2005

A lot of work is devoted to formalizing and devising architectures for agents' cooperative behaviour, for coordinating the behaviour of individual agents within groups, as well as to designing agent societies using social laws. However, providing agents with abilities to automatically devise societies so as to form coherent emergent groups that coordinate their behaviour via social laws, is highly challenging. These systems are called self-organised. We are beginning to understand some of the ways in which self-organised agent systems can be devised. In this perspective, this paper provides several examples of multi-agent systems in which self-organisation, based on different mechanisms, is used to solve complex problems. Several criteria for comparison of self-organisation between the different applications are provided.

Povzetek: Članek opisuje primere in kriterije samoorgarnizacije v agentnih sistemih.

1 Introduction

Multi-Agent Systems (MAS) have attracted much attention as means of developing applications where it is beneficial to define function through many autonomous elements. As multi-agent systems get more complex, questions arise about the best way to control agent activity, and thus application performance. Centralised control of MAS is one approach, but is of limited use because of the risk of dependency on the controlling element, and the consequential lack of robustness. This also makes little sense when agents have capabilities of autonomy that can provide useful benefits in applications. Partially or completely decentralised control is an alternative, but means of implementing this without disrupting agent performance in support of applications are important. Mechanisms of self-organisation [7] are useful because agents can be organised into configurations for useful application without imposing external centralised controls.

This paper discusses several different mechanisms for generating self-organisation in multi-agent systems [8]. Reactive multi-agent systems [29] provide the basis for self-organisation in several examples as the interaction between the agents and their environment provides the flexibility for dynamic change. Cooperation drives self-organisation in the AMAS agent modelling theory [2][8]. The holon concept can also be used to define and analyse self-organising agent systems [26]. In this introductory part, these approaches are now discussed in a more detailed way.

1.1 Self-organisation by Reactive Multi-agent Systems

Reactive multi-agent systems [29] are systems made up of simply behaving units with decentralized control. Agents are situated in a dynamic environment through which they interact. They are characterized by limited (possibly no) representation of themselves, of others and of the environment. Their behaviours are based on stimulus-response rules. Decision-making is based on limited information about the environment and on limited internal states and does not refer to explicit deliberation. The individuals do not have an explicit representation of the collective task to be achieved because of their simplicity. Therefore, the solution of the problem is a consequence of successive interactions between agents and the environment. Their characteristics enable them to adapt dynamically their function or structure to changing conditions without external intervention.

Using such a model to solve a given problem requires designing a system as three components: the environment, the agent behaviours and the dynamics of the whole such that the agent society is able to fulfil its requirements with a reasonable efficiency.

1.2 Self-organisation using Cooperative Information Agents

A specific example of a platform for self-organisation using reactive multi-agents is provided by the DIET Agents platform. This platform [14][6] is a suitable basis for self-organising applications using cooperative information agents. This platform was developed as part of the EU DIET project, inspired by the way that complexity emerges in natural ecosystems.

The DIET Agents platform [14][6] is designed as a three layer architecture: (1) core layer; (2) application reusable component layer; (3) application layer. The core layer provides the minimal software needed to implement multi-agent functionality, through the DIET platform kernel. It also provides basic support for debugging and visualisation. The basic classes and elements in the DIET platform kernel are arranged around an element hierarchy: worlds, environments; agents; connections, and messages.

Agents are located in environments, and can form connections with each other through which messages can be passed. Multiple environments can be situated in worlds. Agents are initially created with only four possible behaviours: creation (of other agents); destruction (of itself); communication (with other agents); migration (between environments). But they are designed so that their properties can be extended.

The other two layers of the platform, the application reusable component (ARC) layer, and application layer support this extension. The ARC layer provides functionality that can be shared between applications, but is not essential for the DIET kernel, while the application layer provides application-specific functionality. Software for applications can be developed in this layer without having to disrupt the core layer.

This platform is appropriate for applications involving cooperative information exchange because individual agents can take on cooperative behaviour by extension of their autonomous capability. The platform is designed that agent action is resource-constrained, so that actions will stop if they start consuming too much system resources. Actions are also fail-fast; they will fail if they are not executed immediately. In this way applications requiring the interaction of many agents can be supported within realistic resource constraints.

This platform is also suitable for applications involving self-organisation because no decisions have been made about how agents should be organised, and they are free to rearrange within and between environments according to application requirements.

1.3 Self-organisation by Cooperation in AMAS

For several years the SMAC (for Cooperative MAS) team has studied self-organisation as a means to get rid of the complexity and openness of computing applications [2]. A theory has been proposed (called AMAS for Adaptive Multi-Agent Systems) in which cooperation is the engine thanks to which the system self-organises for adapting to changes coming from its environment (see [2] and section 4.4. in [8]). Cooperation in this context is defined by three meta-rules: (1) perceived signals are understood without ambiguity, (2) received information is useful for the agent's reasoning, and (3) reasoning leads to useful actions toward other agents. Interactions between agents of the system depend only on the local view they have and their ability to cooperate with each other. These modifications make the organisation of the system also change and therefore make the global behaviour of the system emerge. At the agent level, cooperation is described in a proscriptive way: an agent knows how to detect situations it judges being non cooperative, from its point of view, and acts for always trying to remain cooperative toward others but also toward itself.

1.4 Self-organisation by Holons

According to Koestler, a holon is a self-similar structure that may consist of several holons as sub-structures [17]. The hierarchical structure composed of holons is called a holarchy. Holarchies allow the description of systems as recursive self-similar entities which constitute the holons.

We have chosen to describe the behaviour of the members of a holon and their interactions in terms of roles and organisation. These roles represent the "status" of the holon inside a specific holon. In our approach each holon may play four roles: StandAlone, Head, Part and Multi-Part. As a holon joins a HMAS (Holonic Multi-Agent System) Organisation, it has no special bindings and does not collaborate with any other holon.

This situation represents a *Stand Alone* Behaviour. In this state, the agent's decisions are not attached to any restriction but its own goals and objectives. The holon will remain in this state as long as it is satisfied. The Stand-Alone represents how "non-members" are seen by an existing holon. Following the Holonic Paradigm, the holon seen as a Stand-Alone can actually be the Representative of a holon.

As the representative, the holon plays the *Head* role. According to the objective and rules of the holon, the Head responsibilities and rights may range from merely administrative tasks to be able to take decisions concerning all members. The head is not necessarily a unique holon. After an holon starts performing the Head Role, it will be the representative of the members of his Holon at this level and therefore, able to engage the holon in new tasks.

Members not playing the *Head* role are considered as *Parts* of the holon. Once a holon is accepted in a Holon, its autonomy is reduced because of its obligations with the Holon. The degree of this autonomy lost may vary according to the holon's purpose.

The *MultiPart* Role is a special case of the Part Role. This role is played by holons belonging to more than one Holon. Interesting possibilities are available when a holon is shared.

In order to enable holons to dynamically change their roles, we define a notion of satisfaction. Each holon tries to be self-satisfied. If it cannot reach a satisfaction threshold it tries to change its role. Eventually the last concept of the framework is affinity. The affinity enables one StandAlone holon to choose with which holon to merge. It measures the compatibility of the holon's goals and services.

Self-organisation by holons uses direct interactions and cooperation (see section 4.5 in [8]).

1.5 Overview

Given the existence of multiple mechanisms for generating self-organisation in multi-agent systems, what can self-organised systems be used for? Section 2 reviews a variety of examples of MAS applications drawing upon self-organisation. Section 3 seeks to compare these applications, by identifying some criteria that are general to multi-agent systems. Section 4 provides a conclusion.

2 Applications

The diversity of approaches for stimulating self-organisation within multi-agent systems means that MAS have the potential to support a variety of applications. This section describes some example applications using MAS that draw upon self-organisation to make the applications more effective.

Two applications address problems in information retrieval, using middle agents (section 2.1) and evolutionary algorithms (section 2.2). Further applications are considered in the areas of timetabling (section 2.3), flood forecasting (section 2.4), land use allocation (section 2.5), localisation and tracking (section 2.6), adaptive meshing in wireless networks (section 2.7) and traffic simulation (section 2.8). Other examples of application can be found in [18].This wide range of examples gives an indication of the usefulness of self-organisation in achieving the complex behaviour required for real-world applications.

2.1 Self-organisation of User Communities using Middle Agents

Multi-agent systems can be used to support information exchange within user communities by providing each user with a user agent that represents their interests. But how does a user agent make contact with other user agents that represent users with common interests? Assuming that not all users know each other, which is probably realistic, a pure peer-to-peer network could be used. This would involve flooding a network with queries. But this is inefficient, and risks overloading the system with queries. Middle agents or brokers are an alternative - user agents communicate with middle agents [5]. Multi-agent systems for information exchange using middle agents have been proposed which are centralised (e.g. [22]) - all queries go to one broker - but there is a risk that they will not be so robust, if the middle agent does not perform well. In this example we consider an application where middle agents are used in a decentralised configuration.

The self-organising communities application [31] assigns each user a user agent. There are also multiple middle agents in the system. User agents do not retain a profile for their user, but they forward queries to the middle agents. The user agents carry and seek to acquire information for their users. Each user agent registers with at least one middle agent. Once they are registered with the middle agent, the middle agent can access information that the user agent holds that may be of interest to other user agents. Each middle agent receives queries from multiple user agents. Given these queries the middle agent carries out a search of the pool of information it holds from user agents already registered with it. If it can respond to the query using this information then that information is dispatched to the user agent that issued the original query and the search is rapidly completed. If not, the middle agent can interact with other middle agents to try and obtain information from them. Once the middle agent has carried out this search, it relays the result to the user agent.

The middle agent then examines whether the search was successful, and if so provides a positive mark to the two user agents, both the requestor and the provider. If the search was unsuccessful, the requestor gets a negative mark, to indicate load on the system. After marks have been assigned the middle agent checks the location of the requestor and provider user agents. If the search has been successful, and the requestor and provider agents are not already registered with it, the middle agent requests the middle agent with which the provider agent is currently registered to transfer the provider agent to the group of the requestor agent. Movement of agents between groups is regulated by the awards given to user agents following responses to queries, and designed to get user agents into the same groups around middle agents where they often have queries covering common areas. In this way user communities can be built up using user agents and middle agents, without any central control on agent behaviour. This is a highly scalable process that continues to operate highly efficiently even as the number of users increases substantially.

2.2 Self-organisation through Evolving Agent Populations

We can use a MAS to represent user interests through user agents, but given that there may be many different users in different locations there may be problem in finding other users to interact with. The evolving preferences application [18] considers an application scenario where many users interact with each other via a DIET Agents platform supporting user agents. Each user deploys a user agent in a DIET environment, but because users may interact with the system in different contexts, there will be a lot of different environments. The different environments are connected in a peer network. User agents stay in their own environment, but create a population of scout agents that they send through the peer network to search out other user agents representing users with common interests.

Each scout agent carries information representing the interests of the user that it represents. It also has a preference for environments determined by a bitstring genome created when it is generated. Based on this genome it will search out other environments and interact with other scout agents in them. Scout agents then return to their home environment and report back information that they have gathered in other environments; both about different environments and about their success in interacting with other scout agents representing similar interests. On return to their home environment scout agents are destroyed, but their genome is used in an evolutionary algorithm where the selection criteria is defined by the success of the scout agent in locating environments where there are other scout agents to interact with that have similar interests. Over time the evolutionary algorithm converges to a situation where scout agents will converge in environments according to their users' preferences, so that different environments hold different user agents that can interact on behalf of their users. This shows how an evolutionary algorithm can be combined with agent interaction in a distributed network to stimulate self-organisation of agents into different environments, and thus to stimulate information exchange between users in different environments.

2.3 Self-organisation for the School Timetabling Problem

An example of a classical constraint-satisfaction problem (CSP) is the school timetabling problem in which a timetable for a certain duration must be found while respecting the explicit constraints (availability, specialisation, equipment needed…) of different stakeholders (teachers, student groups and possibly rooms) as well as their implicit constraints (for example, impossibility to be in two places at the same time). The inherent distributed aspect of the timetabling problem explains a processing by a MAS. Unlike most of the approaches using MAS (for instance [4]), agents do not use negotiation to find a solution in ETTO (Emergent Time Table Organisation), the problem solver presented here [23].

The *environment* is made of a three-dimensional virtual grid composed of cells. Each cell represents a time slot for a given day, for a given hour, and for a given lecture room.

Two kinds of cooperative *agents* were identified: a Representative Agents (RA) and Booking Agents (BA). A RA is associated with every human stakeholder, manages its constraints and represents an interface with the real world. RAs delegate time slot and room search to Bas which are the actual self-organizing agents. A BA explores the grid to find free cells and meet potential partners in order to fulfil its aim: booking a time slot for a given lecture to give (for a teacher) or to take (for a student group) in accordance with constraints used by its proxy RA.

The *behavioural model* is based on the AMAS theory, the engine of self-organisation is cooperation. Five different situations for reorganisation are identified based on the three meta-rules ensuring cooperation (see section 1.3) For instance, if a BA ba1 encounters, in a given cell, another BA ba2 with which it cannot partner (for example, two teachers meet), ba1 judges this situation as incompetence and changes its location to find a more relevant partner. Furthermore to enable a more efficient exploration of partnership possibilities, ba1 will memorise the location and the BAs it may know via ba2, to exchange them during further encounters. In a cooperative situation, a BA books the cell in which it is situated, and partners with another BA.

The positive results obtained by now show that the approach used is suited for this kind of problem. BAs are able to relax constraints to find a solution. A solution is found when constraints or stakeholders vary (added or removed) in a dynamical way. Furthermore, the ability to insert agents has enabled us to show that adding supernumerary agents helps finding a solution and gives better results. This can be explained by the fact that the added agents can disrupt others which are satisfied with a solution that could be optimised. However ETTO has weaknesses. By nature, cooperative agents in AMAS have only a limited knowledge about their environment and do not know the global goal to achieve as well as the global cost of the solution they may found. Thus they go on exploring the grid to find a more relevant solution even if the best solution is already found. An external observer has to stop the solving process when the organisation fits his requirements.

Many approaches have been used to try to solve such a problem (see for example, the “Practice and Theory of Automated Timetabling“ at the URL: http://mat.gsia.cmu.edu/PATAT04/). Most of them are (distributed) CSP-based solvers, some are agent-based solutions, some use evolutionary approaches and others ant algorithms [28]. Timetabling problems in the real world are dynamic problems, restarting from scratch each time a constraint is modified (added, removed) would not be efficient and few works are interested in this problem. Usually, the main objective is to have the smallest impact possible on the current solution as in [21] in which this is done by introducing a new search algorithm that limits the number of additional perturbations. In ETTO, self-organisation enables the system to adapt to perturbations and changes in its environment because modifying a stakeholder’s constraint makes the corresponding BA question its bookings and its possible partnership. If it judges that they are inconsistent with its new state, it tries to find new ones by roaming the grid and applying its usual behaviour.

2.4 Self-organisation for Flood Forecasting

Flood forecasting is a complex dynamic problem, parameters that can explain this phenomenon are numerous and heterogeneous: including hygrometry, declivity, surface, nature and permeability of the ground, rain heights, stations topologies, … Current forecasting systems have a physical approach of this phenomenon: the better these parameters are known, the better results are. Tuning these parameters for a forecast station take several months and they have to be adjusted when environmental conditions evolve.

The STAFF real-time simulator uses an adaptive model for flood forecasting, which is composed of two levels of self-organizing multi-agent systems [13].

The *environment* is made of the sensors of the Garonne river basin.

Agents of the lower level represent each physical sensor. Such an agent has to encapsulate its datum for determining its influence on the forecast that the system has to model. The goal of each upper level agent is to compute the water level variation during a unitary period (typically an hour); for that, it uses a weighted sum of agents in the lower level.

The *behavioural model* is based on the AMAS theory. The hydrological model's adaptive nature is obtained by adjustment of these weights, decided from cooperation between the agents. Agents do not know the objective of the global system, the self-organisation by cooperation between agents defines how the model has to be adjusted according to the input data, the results coming from other agents and the error made on the forecast. This makes the model generic and improves its performances.

Positive results were obtained showing that the model correctly followed the real evolution of the flood even in limit use cases (such as noisy and missing data, totally upstream stations or real-time learning) in which usual hydrological models are inadequate. The model does not need any predefined parameters because it is adjusted just once, when installed, using historical flood data. For example, one week was sufficient to adjust the 24 models currently used for the stations making flood forecasting in the Garonne river basin.

Classical physic-hydrological forecasting models are mathematical approaches that consist in generic formula which parameters are tuned from measures on ground and from historical account of flood. Neural networks have been used in flood forecasting, for instance in [30] or used with self-organising feature map [15]. In the former approach, relevant stations must be selected by hand and the learning algorithm is not a generic one. In both approaches, contrary to STAFF, there is no real-time learning.

2.5 Self-organisation for Land Use Allocation

Based on a real-world problem, we applied a self-organising approach to simulate the assignment of land-use categories in a farming territory, in the north-east of France [9][8]. This problem exhibits a function to optimise, while respecting a set of constraints, both local (compatibility of grounds and land-use categories) and global (ratio of production between land-use categories). This problem is one instance of quadratic assignment problem.

The *environment* is the set of available zones in the farming territory, each zone is featured by its surface, its distance to the village, the kind of soil, etc.

Agents are gathered into groups, each being associated to a land-use category. A group has a goal to satisfy by conquering spatial zones in the environment while respecting some constraints.

The *behavioural model* is based on a few principles inspired by the eco-problem solving approach [11]. An agent can conquer a zone in the environment and then contribute to the satisfaction of its group; the zones are more or less attractive for an agent; when searching for a zone, an agent chooses the most attractive one. If the zone is free the agent occupies it; if it is already occupied, the two agents have to fight, and the outcome is determined by the respective strengths of their groups. Finally, the strength of a group decreases while its satisfaction increases, in order to ensure that groups farther from their objectives gain an advantage over those closer.

The problem-solving process exhibits interesting properties. The system produces results that fit the expert's requirements and that are comparable with results obtained by simulated annealing.

The dynamic of the problem-solving process is convergent to a stable state (which is a solution to the problem); is an anytime process (the system can be stopped at any step and is able to produce a solution the quality of which is dependent of the number of steps). Furthermore, the model exhibits self-adaptation properties: at runtime we can add or remove zones or land-use category and the system stabilises again to a solution. We obtained the same properties when modifying a group's goal.

A lot of works on optimisation problems exist (e.g. [3]), however most of them are not self-organizing; two exceptions are built on reactive agents that self-organize. The first (Ant Colony Optimisation) is inspired by the foraging behaviour of ants [1] and the second (Particle Swarm Optimisation) by flocking [10]. Both provide results comparable to more conventional optimisation methods.

In our case, we compared our system with simulated annealing. Simulated annealing provided better mean results even if the best solutions were found by the MAS approach.

2.6 Localisation and Tracking using Self-organisation

The localisation task can be defined as finding the position of an object (or more than one), mobile or not, in a well defined referential location. The tracking problem is to provide a succession of positions that are spatially and temporally coherent. We proposed a reactive model to tackle this issue [12].

The *environment* of the agents is a representation of the real world. It is a square grid in which each state represents a target's possible position and is featured by an altitude that represents the possibility of presence of a target at this position and is provided by sensors. Environment dynamics is determined by accumulation and evaporation principles. The altitudes are refreshed (accumulation) continuously as soon as sensors can furnish data. In the absence of data, altitude is decreasing (evaporation).

Agents are equivalent to weighted particles evolving in an environment of force field.

The *behavioural model* used is inspired by a model of flocking [24] but is expressed through a formulation taken from Newtonian physics (i.e. all behaviours are expressed as a combination of classical forces). Agents are attracted by position according to their altitude and are mutually repulse each other. Agents' movements are the consequence of these forces.

We designed these antagonist behaviours to obtain a focusing of the agents on the position of highest altitude and a homogeneous spatial distribution of them in the rest of the environment (where there is a null altitude). Focusing is an emergent phenomenon, and is the solution of the problem: a group corresponds to the detection of a target.

We compare our proposition with the Kalman filter in case of real robots' localisation [27]. The Kalman filter is better than the agent-based method when there is no noise. This advantage decreases when noise is introduced. Furthermore, the agent-based approach requires less knowledge about the problem than the Kalman one.

The approach is able, at runtime, to deal with a variable number of targets and it is possible to add or remove sensors which is very difficult to take into account with classical algorithms. As far as we know there is no self-organized approach for localization.

2.7 Self-organisation for Adaptive Meshing in Cellular Radio Networks

A distinguishing feature of cellular radio mobile networks is the rapid increase of the consumer demand and the ensuing complexity in their design and management. Responding to this demand requires the space to be partitioned between a large amount of service units or cells. The adaptive meshing problem for dimensioning considers traffic statistics as a predefined resource that must be attributed to many adaptive low power Base Transceiver Stations. The environment is discretized in meshes which contains a number of resources according to traffic statistics. Each mesh will be assigned an agent whose main and unique goal is to cover the traffic in that mesh [26]. This goal must be accomplished respecting certain constraints like geometry and the maximal traffic that an antenna can cover. The problem solving is done by building up holons which cover a resource.

In the adaptive mesh problem, a stand-alone holon must ensure the coverage of its resource, then it will try to join a mesh immediately. The only situation where it remains in a stand-alone role is when its resource can get an antenna for it alone. A holon that performs the head role will be responsible for respecting the constraints of a mesh. It will be representing a possible mesh in the system, and will accept or refuse other holon's requests to fusion according the constraints. Although all heads represent possible meshes in the system, a Holon Head can decide to leave its role if, after trying to improve the Holon's satisfaction, the satisfaction is insufficient to remain as a Holon. In order to improve the Holon's satisfaction, will accept new holons to increase the Holon covered resource, or will command member holons to leave the Holon if they don't respect the geometrical constraints or if the covered resource has exceeded the maximum.

A holon gets the Part role if negotiations with a Holon succeed. It will remain in the Holon if its satisfaction level is raising. However, it is also possible that the agent receives a command to leave the holon, in that case, it must return to Stand-Alone and restart the merging process.

The holon's identifier should give the position of the holon's resource (X, Y coordinates) and the traffic it contains.

Using these values, a holon can determine whether or not to merge. As explained before, the affinity should give a measure of the compatibility of the holon's goal and services. In this particular case, both holons will have the same goal, to ensure the coverage of their resources. Therefore, the main problem is to ensure that the geometrical constraints are respected. The affinity could be decomposed in two main parts: the distance affinity will provide a geometry dependent value used to ensure that the geometrical constraints are respected. As we need square meshes, we will use two parameters to test the distance affinity. First, we will check if the holon trying to merge is inside the acceptance distance. The Resource affinity is used to ensure that the limits of an antenna are not exceeded.

2.8 Self-organisation for Traffic Simulation

Multi-agent Systems operate within an environment and therefore, in an Agent Based Simulation (ABS) special attention must go to the analysis, model and implementation of the environment [20].

We propose the use of holarchies for the modelling of environments [25]. We simulate traffic within an industrial plant. The environment of this simulation is defined by the topology and road network of the plant. The concept of road is divided into links. A link represents a one-way lane of a road. A segment is composed of two exchange points, called input and output exchange points, and a link. Exchange points let vehicles pass from one link to the other. An exchange point is always shared by at least two segments and thus plays the multi-part role. The industrial plant is composed of a set of zones, that in turn contain Buildings and Segments. Buildings and Segments can also communicate through shared exchange points. Usually an exchange point represents a crossroad, but in can also represent an entrance used by trucks to access buildings. The agents will be the different vehicles driving through the plant. Each holon of the holarchy represents a specific context. For this simulation (HTS in the sequel) it's a specific place in the plant. These places have different granularity levels according to their level in the holarchy. During the simulation vehicle agents move from one holon to another and the granularity is chosen by execution or simulation constraints such as which features can be observed. The dynamic choice of the environment granularity level during the simulation is transparent for the agents. The problem here is the simulation of traffic and the solving process is again based upon building and re-organisation of holarchies as vehicles drive through the plant and change the holon they belong to.

This holarchy defines the organisational and topological structure in which agents will evolve. Each environmental holon will enforce contextual physical laws and represent a specific granularity level of the real plant topology. This holarchy is predefined as it represents the real plant environment. Indeed, the latter can't evolve and the physical laws we need to enforce are known *a priori*. All necessary information to simulate the traffic inside a link is local (other vehicles, road signs, etc). This makes the model easier to distribute in a network and leaves the door open to Real-time applications as well as Virtual Reality implementations.

This approach has many advantages for the simulation. Indeed, such a definition of the environment allows the progressive decomposition of the environment complexity and enables to assign environmental laws to the pertinent holon.

3 Comparison and Discussion

This part is an attempt to compare self-organised systems presented above using a list of criteria inspired by the work done in the AgentLinkIII "Self-organisation in MAS" Technical Forum Group. The first paragraph of this section lists all the criteria we use along a classification based on the level at which they can be expressed. In the next section, each criterion is discussed in more detail with regard to our different approaches.

3.1 Criteria Used

Some of the criteria we use can be considered as descriptive/static criteria of the approach whilst others are related to the dynamical aspects of the problem solving process. Criteria belonging to the first group can be stated without running the system but by "simply" looking at its description:

- Absence of external or of centralised control: no entity, external or internal to the system, is explicitly responsible for the actions of agents or for centralising information flow.
- Dynamic operation: the solution is built in a dynamic way and not by applying a predefined plan or by instantiating a predefined solution.
- Emergent properties: properties that emerge from local interactions within the system and that cannot be deduced by simply observing individual behaviours (see section 3 in [8]).
- Simple local rules: do simple individual behavioural rules lead to complex patterns?
- Reusability: is the solution (or part of it) reusable in other contexts?

On the contrary, criteria found in the second group need experiments to be tested:

- Anytime property: the system can be stopped at any step and is able to produce a solution the quality of which is dependent of the number of steps.
- Instability: is the system non-linear, is it sensitive to parameters variations?
- Adaptation: how does the system react to changes coming from the environment of the system?

3.2 Discussion

In some applications, the absence of ***external control*** may not exist and some entities may centralise information or decision. Therefore, applications can be classified from fully decentralised to partially centralised.

For example, localisation application is fully decentralised, as well as timetabling or flood forecasting.

On the contrary, in the holonic approach, Head role refers to a partial centralisation of decisions. This loss of autonomy corresponds to the holarchy handling. Moreover, the Head role may be played by the entire holon as a group.

The self-organising communities application, using middle agents, is decentralised in that information retrieval is distributed across multiple middle agents. But the evolving preferences application, which evolves environmental preferences for information exchange, includes an element of centralisation through the use of selection on scout agent populations; although this is only partial as multiple populations are selected in parallel.

In all the applications presented above, the solution is built ***dynamically.***

Emergent properties refer to ***simplicity*** of individuals, in terms of local rules or behaviours, despite their collective ability to produce a complex pattern. In other words the concepts needed to explain the global properties are not present at agents' level.
For example, in the land use allocation problem, the global constraint (ratio of production between land-use categories) is not explicitly represented at the agent level but implicitly formulated through the groups' goal and the strength of groups that affect the conflict's outcome. In the localisation problem, we need to interpret the spatial positions of the agents in order to detect groups and then obtaining the target position. In the examples of allocation and localisation, the agents' behaviour is equivalent to stimulus-response rules and therefore is simple (at least simpler than the collective patterns that emerge).

In all problems solved with self-organisation by cooperation, and following the AMAS theory, the function of the system is not known by the agents which only know their own local and simple function led by cooperation rules. By not being able to have a global knowledge, agents do not know when a solution is found, and an external observer is needed to make this detection.

For the holonic approach since holons are recursive structures the behaviour of a holon may be the result of interaction of sub-holons. Indeed, a holon which is unable to accomplish its goal will try to merge with a holon with complementary capabilities.

Emergent properties are also apparent in the self-organising communities application, which converges to a situation where groups of user agents with common interests can interact despite an arbitrary configuration of agents initially.

Following the ***anytime property***, experiments have shown that the main feature of the timetabling application is that modifications can be done without stopping the search for a solution (the schedule) while this latter is in progress, unexpected events are processed while actors are changing their constraints. The schedule is constantly changing as agents are searching for better bookings and partnerships and it becomes better as the solving makes progress. In the flood forecasting, a current solution (model) is given also at anytime, it becomes better as the learning process progresses. However, if disturbances appear, in both cases, the current solution may be questioned by agents for which unpredictable changes create non cooperative situations. Therefore the solution may be totally changed, may become totally “false” before converging again towards a good solution in a more or less great time.

In the AM there is a first step which is the construction of holarchies. After then the solution improves as the time allocated grows.

All applications based on reactive agents experimentally show their anytime ability.

In the self-organising communities application, user agents and middle agents rearrange user communities dynamically depending upon queries, and this can be stopped and resumed at any time, and thus support the anytime property. The evolving preferences application uses an evolutionary algorithm to stimulate preferences in populations of scout agents; the algorithm can be stopped and restarted but it will not continue changing for ever as an optimum will be reached.

Some solutions are ***reusable*** like the application independent framework of the holonic approach which can be applied to problem solving or simulation in different contexts. The general framework of the solution built for the timetabling problem can be also reused in other constraint satisfaction problems (such as supply chain management, for example) but rules enabling agents to detect and solve non cooperative situations are specifically suited to the problem.

Reaction to perturbations (sensitivity, robustness, adaptation or instability). Given a system that stabilizes on a state (solution), when a perturbation occurs the system can i) escape from this state and potentially reach another stable one (this is adaptation); ii) temporarily change its state and come back to the initial stable state (this is robustness), iii) change from state to state without stabilizing (instability), or iv) change of state even in case of small perturbations (sensitivity). To assess these criteria, we need to perturb the system at runtime. A perturbation can be viewed as an external event on the system in relation with the unpredictable and dynamic features of the application domain. In the timetabling application, perturbations come from the stakeholders that may change their constraints in a dynamic way, new actors can also vary at runtime. Failures of sensors can also be viewed as unexpected events for the flood forecasting application.

In land-use assignment, we did successful experiments where at runtime we changed the number of zones, the number of land use categories or by changing the goal of groups. In localization application, the system can successfully deal with situations where the number of targets and of sensors can vary at runtime. In all these situations, the systems were robust or adaptive according to the degree of the perturbation.

The timetabling application is a good example of a non-linear complex system in which simple and small variations (personal constraints, for example) may imply major changes in the problem solution. Partnerships an agent makes can disturb other agents, thus a little modification in the timetable can question the current solution which may vary greatly. The system is sensitive to perturbations, but is also able to adapt to these changes. Indeed, the very essence of systems built by applying the AMAS theory is adaptation which is obtained by enabling agents to locally decide to change their interactions with each others using cooperation as a local criterion.

In the AM and HTS the adaptation is done by the reorganisation of holarchies and is the basis of the approach.

In the self-organising communities application, the behaviour of middle agents is designed to respond to perturbations from users introduced via their user agents. In this example a perturbation is in the form of a novel query. This may provide information about a change of interests of the user and hence of the user agent, and as a result the user agent may move from one community around a middle agent to another. The use of rewards for successful responses to queries provides a mechanism to react to perturbations. In the evolving preferences application such response to perturbation is reflected in changes in selection pressure on the scout agent population, and hence changes in the outcome of the evolutionary algorithm.

This framework is still a tentative way to compare self-organised applications and it can be improved in two directions.

The first is the criteria list itself: it is still subject to discussion as the definition of criteria can be questioned and other criteria can be added to refine this comparison framework.

The second is related to the kind of answer for each criterion; it is currently subjective according to the interpretation of the definition. It would be of interest, when criteria are well established, to provide measurements of them. For example, measuring the decentralisation degree by the percentage of agents in the system that are directly involved by the decision of another one (this can be measured by the number of agent in a holon when the head agent takes a decision).

4 Conclusion

Multi-agent systems can be developed in many different ways. The autonomous nature of individual agents means that complex properties can emerge at the multi-agent system level. Self-organisation can be a useful way of controlling and regulating this complexity, especially when seeking to support an application. In a general way, applications that are too complex to give an a priori algorithm, that are plunged into open and real environments (the Internet, for instance) and for which a perfect design cannot be guaranteed can benefit from self-organisation.

This paper has presented several examples of applications based on multi-agent systems that use self-organising behaviour among the agents to facilitate application properties. The AMAS theory which uses self-organisation by cooperation has been successfully applied to various application domains: simulation, e-commerce, network management, collective robotics, mechanical design in avionics, flood forecasting, and biological modelling. Reactive multi-agent systems have proved useful in diverse application areas such as localisation in mobile robotics They have also provided the basis for cooperative information agents for information retrieval. Agents based on self-organisation through holons have been useful in meshing for cellular networks, and in traffic simulation, for example.

Self-organising multi-agent systems are at an early stage of development, with many different mechanisms of self-organisation to be explored. Despite this a number of examples have been outlined here, and already a diversity of application areas are being explored. We can anticipate self-organisation being of further relevance for applications of multi-agent systems in the future.

The framework of comparison we have provided has proven its usefulness to understand these approaches even if it has to be considered as a first and tentative approach that needs to be improved. It is obvious that the choice of one approach is application-dependent and these criteria may be of some help in this choice.

The authors would like to thank the participants of the two meetings of the AgentLinkIII "Self-organisation in MAS" TFG for their fruitful discussions.

References

[1] Bonabeau E., Dorigo M., and Théraulaz G. (1999) *Swarm Intelligence: From Natural to Artificial Systems.* Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, New York, NY, USA.

[2] Gleizes M.-P., Camp, V. and Glize P. (1999) A Theory of Emergent Computation Based on Cooperative Self-Organisation for Adaptive Artificial Systems, *4th European Congress of Systems Science,* Valencia.

[3] Corne D., Dorigo M., and Glover F. (1999) *New Ideas in Optimization*, Mac Graw Hill.

[4] De Causmaecker P., Ouelhadj D., and Vanden Berghe G. (2003) Agents in Timetabling Problems. Proc. of the *1st Multidisciplinary International Conference on Scheduling Theory and Applications*, pp. 67-71, UK.

[5] Decker K., Sycara K. and Williamson M. (1997) Middle-Agents for the Internet. Proc. of *International Joint Conference on Artificial Intelligence (IJCAI-97)*, Japan, pp. 172-175.

[6] DIET Agents platform: http://diet-agents.sourceforge.net/index.html

[7] Di Marzo Serugendo G., Foukia N., Hassas S., Karageorgos A., Kouadri Mostéfaoui S., Rana O. F., Ulieru M., Valckenaers P., and Van Aart C., (2004) Self-Organising Applications: Paradigms and Applications. *Engineering Self-Organising Systems: Nature-Inspired Approaches to Software Engineering***,** G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, F. Zambonelli (Eds), Lecture Notes in Artificial Intelligence 2977, Springer-Verlag, Berlin, pp. 1-19.

[8] Di Marzo Serugendo G., Gleizes M-P., and Karageorgos A., (2005) Self-Organisation and Emergence in MAS: An Overview, *Informatica*, this issue, Ljubljana, Slovenia.

[9] Dury A., Le Ber F., and Chevrier V. (1998) A Reactive Approach for Solving Constraint Satisfaction Problems: Assigning Land Use to Farming Territories, In Proc. of *Agents Theories, Architectures and Languages 98 (ATAL'98)*, Lecture Notes in Artificial Intelligence 1555 "Intelligent Agents V", J.P. Muller, M.P. Singh et A. S. Rao (eds), Springer-Verlag, pp. 397-412.

[10] Eberhart R., Kennedy J., and Shi Y. (2001) *Swarm Intelligence*, Morgan Kaufmann Publishers.

[11] Ferber J., and Jacopin E. (1990) The Framework of Eco-problem Solving. In *Proceedings of the European Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'90)*, pp. 181-193.

[12] Gechter F., Chevrier V., and Charpillet F. (2004) Localizing and Tracking Targets with a Reactive Multi-Agent System, In *Second European Workshop on Multi-Agent Systems (EUMAS'04)*, pp. 255-262.

[13] Georgé J.-P., Gleizes M.-P. Glize P., and Régis C. (2003) Real-time Simulation for Flood Forecast: an Adaptive Multi-Agent System STAFF, *Proceedings of the AISB'03 Symposium on Adaptive Agents and Multi-Agent Systems*, University of Wales, Aberystwyth, pp. 7-11.

[14] Hoile C., Wang F., Bonsma E., and Marrow P. (2002) Core Specification and Experiments in DIET: a Decentralised Ecosystem-Inspired Mobile Agent System, Proc. *1st Intl. Conf. Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pp. 623-630.

[15] Hsu K., Sorooshian S., Gupta H. Y., Gao X., and Imam B. (2002) Hydrologic Modeling and Analysis Using a Self-Organizing Linear Output Network, In Rizzoli A.E. and Jakeman A.J. (eds), Integrated Assessment and Decision Support, Proceedings of the *First Biennial Meeting of the International Environmental Modelling and Software Society* (iEMSs'0), Manno, Switzerland, pp. 172-177.

[16] Kaplansky E., Kendall G., Meisels A., and Hussin N. (2004) Distributed Examination Timetabling, In Proc. of the *5th International Conference of the Practice and Theory of Automated Timetabling (PATAT)*, Pittsburg, USA, pp. 511-516.

[17] Koestler A. (1990) *The Ghost in the Machine*, Reprint edition, Penguin, East Rutherford, NJ, USA.

[18] Mano J.-P., Bourjot C., Lopardo G., and Glize P. (2005) Bio-inspired Mechanisms for Artificial Self-organised Systems, *Informatica*, this issue, Ljubljana, Slovenia.

[19] Marrow P., Hoile C., Wang F., and Bonsma E. (2003) Evolving Preferences among Emergent Groups of Agents, In *Adaptive Agents and Multi-Agent Systems*, E. Alonso, D. Kudenko & D. Kazakov (eds.), Lecture Notes in Artificial Intelligence 2636, Springer-Verlag, Berlin, pp. 157-173.

[20] Michel F., Gouaich A., and Ferber J. (2003) Weak Interaction and Strong Interaction in Agent Based Simulations. *Multi-Agent Based Simulation III.* D. Hales et al. (Eds), Lecture Notes in Artificial Intelligence 2927, Springer-Verlag, Berlin, pp. 43-56.

[21] Müller T., and Rudova H. (2004) Minimal Perturbation Problem in Course Timetabling, In *Proc. of the 5th International Conference of the Practice and Theory of Automated Timetabling (PATAT'04)*, Pittsburg, USA, pp. 283-304.

[22] Paolucci, M., Niu Z., Sycara C., Domashev S., Owens S. and Van Velsen, M. (2000) Matchmaking to Support Intelligent Agents for Portfolio Management, In *Proc. of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, Calif., AAAI Press, pp. 1125-1126.

[23] Picard G., Bernon C., and Gleizes M.-P., (2005) ETTO: Emergent Timetabling by Cooperative Self-Organisation, *Third International Workshop on Engineering Self-Organising Applications (ESOA'05)*, Utrecht, The Netherlands, pp. 31-45.

[24] Reynolds C. W. (1987) Flocks, Herds, and Schools: A Distributed Behavioral Model, In *Computer Graphics, SIGGRAPH Conference Proceedings*, pp. 25–34.

[25] Rodriguez S., Hilaire V., and Koukam (2005) A. Holonic Modelling of Environments for Situated Multi-Agent Systems, Submitted to *E4MAS'05*.

[26] Rodriguez S., Hilaire V., and Koukam A. (2003) Towards a Methodological Framework for Holonic Multi-agent Systems, In *Proceedings of the Fourth Workshop on Engineering Societies in the Agents World (ESAW'03), pp. 31-45*.

[27] Roumeliotis S.I., Sukhatme G.S., and Bekey G. (1999) Circumventing Dynamic Modeling: Evaluation of the Error-State Kalman Filter applied to Mobile Robot Localization, *IEEE International Conference on Robotics and Automation*, IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 1656-1663.

[28] Socha K., Sampels M., and Manfrin M. (2003) Ant Algorithms for the University Timetabling Problem with Regard to The-State-of-the-Art, In *Proceedings of the 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization (EvoCOP'03)*, Essex, UK, pp. 334-345.

[29] Van Parunak H. (1997) Go to the Ant: Engineering Principles from Natural Multi-Agent Systems, *Annals of Operations Research 75*, pp. 69-101.

[30] Vergnes J.-C. (1995) Etude de modèles de prévision à la station de Nant - Exploitation de l'utilisation des réseaux neuronaux en prévision de crues, *Report ENSEEIHT/DIREN.*

[31] Wang F. (2002) Self-organising Communities Formed by Middle Agents, Proc. *1st Intl. Conf. Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, pp. 1333-1339.

[32] Yokoo M., Durfee E., Ishida Y., and Kubawara K. (1998) The Distributed Constraint Satisfaction Problem: Formalization and Algorithms, *IEEE Transactions on Knowledge and Data Engineering*, 10, pp. 673-685.