

UDC, DOI: N/A

Emergent Properties for Data Distribution in a Cognitive MAS*

Andrei Olaru, Cristian Gratie, and Adina Magda Florea

University Politehnica of Bucharest,
Splaiul Independentei 313,
060042 Bucharest, Romania

cs@andreiolaru.ro, cgratie@yahoo.com, adina@cs.pub.ro

Abstract. Emergence is a key element in the research of multi-agent systems. Emergent properties provide higher level features to a system formed of simpler individuals. So far, emergence has been studied mostly through systems formed of reactive agents – that are easier to design and implement. As computational power increases and even small devices become more capable, cognitive agents become a promising solution for more complex problems. This paper presents a multi-agent system that uses cognitive agents and that is designed to manifest emergent properties: a data storage system that assures distribution and replication of data as emergent properties, without the need for these features to be controlled in a centralized way.

Keywords: multi-agent system, cognitive agent, emergence, self-organization.

1. Introduction

Although emergence and self-organization have been the subject of research for a long time, it is only relatively recently that they have come to attention in the field of multi-agent systems (MAS). Its role in MAS is crucial, as it is a concept that fits perfectly with the agent-oriented paradigm and leads to important advantages.

Several definitions of emergence exist [4] but none is yet generally accepted. Many times emergence is defined by its effect – the spontaneous formation of patterns in the structure or behaviour of certain systems.

In the context of a large number of agents, forming a complex system, emergence offers the possibility of obtaining a higher level function or property by using agents with lower level implementations. The use of emergence and self-organization allows for a considerable reduction in the complexity needed for the individual agents, therefore they can run on simpler and smaller devices. Moreover, self-organization leads to properties like robustness, flexibility, adaptivity and scalability.

* The original publication is available at
<http://www.comsis.org/ComSIS/Vol7No3/RegularPapers/paper12.htm>



Many recent papers on emergence and multi-agent systems deal with emergent properties and behaviour in systems formed of a large number of reactive agents. Reactive agents are used because they are simple and need only very limited computational capacity, but emergent functions may be more complex and the structure is robust [3, 20, 37].

Nowadays, the capabilities of even very basic computing devices have considerably increased, allowing for a much more complex internal structure of agents – the possibility to hold reasonable amounts of data and to have a more nuanced behaviour. Cognitive agents have knowledge about the surrounding environment, have goals they desire to fulfill, make plans and take action in order to fulfill them. There is important research on emergence in cognitive agent systems, but there is still much room for improvement.

One of the domains of application for agent-based self-organizing systems is Ambient Intelligence (or Aml) [11]. Software agents (particularly cognitive agents) are considered as a very adequate paradigm for Aml [27], due to the properties of agents: autonomy, proactivity, reactivity, adaptivity and reasoning. A system for Aml must deal with a great number of interconnected devices (sensors, actuators, mobile phones, laptops, desktop computers, "smart" appliances), and needs to assist the user with daily activities. Most times assistance is performed by providing the user with interesting information. In the same time, the system must be robust and resilient to failure, while being decentralized, so an approach based on self-organization fits the requirements [6, 17, 16].

The purpose of this paper is to study emergent properties in a multi-agent system formed of cognitive agents, that is, study how such a system should be designed in order to obtain global, emergent, properties. The long-term goal of this research is to apply the results to the design of a middleware for Aml that facilitates the distribution of information through a network of devices, in a self-organized manner.

A system of cognitive agents used for the storage of data has been designed and implemented. While treating, at the time, a very specific case, this prototype makes a good framework for the study of emergent properties. It was designed with the purpose of manifesting emergent properties like uniform distribution and availability of data. Individual agents were given local goals that do not directly imply that the data is uniformly distributed or that data is not completely lost. No agent has the capability to have a global image on the system or even on a significant part of it. Knowledge of the agents is limited to beliefs about their immediate or very close neighbours. Yet the system as a whole keeps data well replicated and distributed.

This work presents the design of a system that satisfies the requirements above, using techniques of self-organization and emergent properties. The presented experiments were performed not necessarily to show how the system performs under heavy loads or if it is resilient to failure (although we expect it to), but to show a kind of global behaviour that can be obtained by local interaction, in the context of using cognitive agents.

A practical application to the studied system would be, for instance, a network of sensors in a building. The sensors have low computational capabilities so their behavior must be simple and use little memory. They sense events in the environment, like temperature or the presence of people. The system is completely decentralized, but information must be available for retrieval from any of the sensing devices. Therefore, sensors aggregate their sensor information into larger chunks of data and "insert" it into the system, which, by using only local information and communication between neighbour sensors, distribute the data and make it uniformly available throughout the building. This is only an example. The system is generic and was built to fit a greater number of possible applications.

The paper is organized as follows. Section 2 is dedicated to related work in the field of emergence and multi-agent systems. Section 3 presents the main topic of this paper: a cognitive multi-agent system designed to manifest emergent properties. Section 4 describes the results obtained from experiments. The last section is dedicated to conclusions.

2. Related Work

What we know is that emergence appears in the context of complex systems [1] – systems composed of a large number of interacting individual entities. Emergence needs two levels of perspective: the inferior, or micro level of the individual entities and the superior, or macro level of the whole system. A simple definition is that "emergence is the concept of some new phenomenon arising in a system that wasn't in the system's specification to start with" [35]. A more elaborated definition is that "a system exhibits emergence when there are coherent emergents at the macro-level that dynamically arise from the interactions between the parts at the micro-level. Such emergents are novel with respect to the individual parts of the system" [8]. An "emergent" is a notion that can represent a property, a structure or a behaviour that results from emergence.

The essence of emergence is not actually the novelty or the unexpectedness of the emergent – as these will fade at later experiments although the emergents will stay the same – but the difference between the description of the individual and the description of the emergent, from the point of view of an observer [35,28]. If the minimal description of the individual is taken, it cannot be used to describe the emergents resulting from the system, therefore the emergent is considered as novel and, potentially, unexpected.

There are a few important features of emergence [8, 13, 18]:

- emergence occurs out of the interactions between the parts of a complex system;
- emergence is defined in relation with two levels – it is manifested at the higher level, arising from interactions at the lower level;
- the representation of the emergents cannot be reduced to the specification of the individuals;

- emergents only arise after a certain time in which the system has evolved;
- once they occurred, emergents will maintain identity over time;
- emergents arise without any centralized or exterior control;
- the emergent phenomenon is robust and flexible, i.e. it is not influenced by damage on the system (even if the emergent is temporarily not observable, it will arise again as the system evolves).

In the field of multi-agents systems, most examples that demonstrate emergent properties use reactive agents [34]. This is because they are easier to implement and control, and they are suitable for small devices with low computational power. But, more than anything, systems of reactive agents are easier to predict and to design so that they will manifest self-organization or other emergent properties. Notable examples of emergents in reactive agent systems relate to the formation of a certain geometrical or geometry-related structure or behaviour. Self-organization is reached by mechanisms that are usually inspired by nature and the behaviour of animal societies (generally insects) [21].

For example, "smart dust" uses attraction and repulsion forces and simple leader election algorithms to dispose individuals in a circular shape, or in a lobed shape, that is resilient to structural damage [3, 20, 37]; "spider" agents roam through the pixels of an image and use stigmergy to mark the different areas of the image [5]; local behaviour can be used for the gathering of resources in a single area [29], for the foraging of food or the transportation of loads [36]; in a very interesting example of dynamic self-organized behaviour, agents establish, by means of emergent coordination, traffic directions through narrow corridors for the transportation of resources between two areas [24].

Although reactive agent systems may be very useful, there are many advantages that a cognitive agent has over a reactive agent. First, it is proactive. Even if there are no signals, perceptions or stimuli from the environment, a cognitive agent may act by itself, taking action according to its objectives. Second, a cognitive agent is aware of its situation and may reason about it. It is aware of what it is supposed to fulfill as final goal and is capable of making plans and taking action towards the realization of its goal. The cognitive agent can use its experience and information about the environment and the consequences of past actions to develop a better plan every time a similar situation occurs.

Ricci et al. support the concept of *cognitive stigmergy* [31], that is inspired from natural, reactive, self-organization (stigmergy, introduced by Grassé [14]) but is using cognitive agents and a more complex environment. The environment is composed of *artifacts* that agents are using and in relation to which they make annotations. The usage of shared resources in the environment by one agent driven by local goals influences the other agents that have access to the resource, leading to feedback and eventually to organization. The research shows encouraging results in applying the concepts from self-organizing reactive architectures to cognitive agent systems.

Other studies of emergent behaviour in cognitive agents exist. Emergence of norms in a social game is possible through *social learning* [15, 33]. Gleizes et al show that local cooperative behaviour of cognitive agents can lead, by means of

emergence, to better system-wide results [12]. The study is continued, using the AMAS (adaptive multi-agent system) technology as a framework that facilitates emergent functions based on cooperation [7].

However, the use of cognitive agents in self-organizing systems is still very reduced. While, in the present, numerous real-life applications of self-organization exist [26], they are based on reactive behaviour of the individual entities. It is true that these systems are easier to verify and control, and it is also true that even in reactive agent systems it is hard to find a good balance between explicitly designed behaviour and implicit, emergent behaviour [25], but the features of reactive systems are limited by the absence of reasoning in agents.

Besides taking inspiration from biological systems, there is also another approach to the design of self-organization, that is used mostly in wireless sensor networks. WSNs use simple individual units and, more importantly, need to consume little power. This approach is also many times validated formally, not only by simulation. It consists of different algorithms used for routing, resource allocation, structure formation, synchronization, power conservation and resilience [10, 23]. Bernadas et al. demonstrate communication services based on this type of self-organizing algorithms, with individual agents having a more complex internal behaviour and manifesting features like proactivity and a certain amount of reasoning [2]. A model for a self-organizing system for communication systems is proposed by Marinescu et al., using cognitive agents that have local goals and specific available actions. The model is verified formally and relevant results are shown [22]. However, these models address particular concerns in the functionality of a system and are not very flexible. It is a considerable problem to build a model that will address more than one concern at a time, and in a generic manner [22, 23].

It is worth mentioning in this section the related work on holonic systems. Holonic systems are based on entities that are, at the same time, wholes in themselves and parts of larger wholes [19]. Self-organization is also present in holonic multi-agent systems. Moreover, in holonic systems organization may be present on multiple levels [32]. However, there is little relation between this work and holonic systems. At present, our goal is to study emergence in a system that works on a single level, but a closer look to holonic systems will be considered for future work.

In this paper we present a multi-agent system that uses cognitive agents to obtain emergent properties. In the design of the system we used techniques from self-organizing reactive agent systems, but considering the cognitive features of the agents. This resembles the work on cognitive stigmergy [31], but uses direct interaction instead of stigmergic mechanisms. Different from other works on emergence in cognitive agent systems, our work, while trying to remain generic, is focused on the exchange and distribution of information through a multi-agent system. Our goal is not a function that needs to be calculated, nor a certain well-defined result that must be obtained, but a behaviour that needs to be achieved. And this by means of cognitive agents, because of the additional features they offer, which will not only favour a more adaptive behaviour

and the possibility to work at the semantic level, but will also reduce the quantity of needed communication by having knowledge about and reasoning on their environment.

3. A Cognitive Multi-Agent System for the Distribution of Data

An application has been developed and experiments have been performed in order to demonstrate how emergents may arise in a system formed of cognitive agents. The system that was designed models a network for the storage, replication and distribution of data. The purpose is to store pieces of data inserted into the system and distribute them, as well as provide the user with the data that is requested.

The requirement that led the design of the system was to use local interaction and local knowledge about the system in order to obtain global emergent properties. The implementation was to be generic, without referring to a certain domain of application, this way establishing a more general methodology for building a system with emergent properties.

The presented experiments were performed with a set of agents placed on a rectangular grid. Each agent can communicate only with its 8 neighbours. Each agent has a limited capacity for the storage of data. The agents must always have some capacity left, ready for data that might be injected from the environment. However, agents must try to store as much data as possible, in order to not waste the capacity of the system. The information stored in the system (the information that is useful to the users) is represented as generic pieces of data, or, in short, *Data*.

The desired emergent properties of the system are the replication of data, uniform distribution and availability. This means that, after letting the system evolve for some time, the following properties should exist: each piece of data should be held by more than one agent; any (reasonably small) area of the grid should contain copies of all pieces of data present in the system; if requested to, any agent should be able to get hold of any piece of data, i.e. copies of all pieces of data should exist in the proximity of any agent.

While the experimental setup is simple, the purpose of this research was to show how a system should be designed in order to obtain emergent properties like the ones specified. Even if the system is generic (not having requirements related to a specific setup), there are still many challenges that arise from the full distribution of the system and from the use of local knowledge and interactions: the agents must not become overwhelmed with messages or with beliefs, nor must they become unresponsive or lose all knowledge about them and their surroundings. All that while keeping the behaviour basic enough to be efficient on devices with limited capabilities.

To design agents that will lead, by means of local interaction, to global properties, we have followed ideas from previous work on systems with emergent properties [3, 4, 9, 20, 21, 37]. We apply these ideas to a system of cognitive

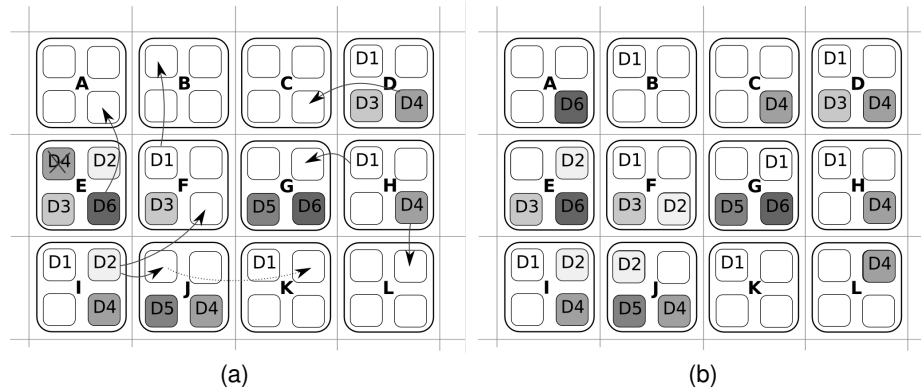


Fig. 1. Group of agents (named **A** to **L**), before (a) and after (b) the data transfers indicated by the arrows. There are 6 data pieces in the system: D1 to D6.

agents (as opposed to reactive agents), envisaging a behaviour that relates to information, rather than more simple concepts. First, the emergent properties cannot be of different nature than the local capacities of the agent. For example, if the agent can only move around, what we can expect as a global behaviour of the system would be for the agents to form structures of a certain shape [37]. Local behaviour should reflect the features of the global desired behaviour. There should also be feedback, that will lead to the stabilization of formed structures and to the dynamical equilibrium of the system [18].

In our design, a lot of inspiration has been taken from the human behaviour (as it is with the design of software agents in general), providing the agents with capabilities that will keep their knowledge bases, goal lists and message queues at reasonable loads. In order to obtain data replication and distribution, an agent is "curious" and, if capacity is available, it will be interested in and it will request a copy of a piece of data it does not hold. In order to obtain variation and uniformity in the data distribution pattern, an agent will "lose interest" in pieces of data that are already held by most of the neighbour agents and may "forget" them. In order for an agent to be able to get hold of pieces of data not held by itself or any of the neighbours, agents will be able to collaborate for the common goal of finding a certain piece of data that, since data is well distributed, cannot be very far away. More on the implementation of these features is presented in the description of the agent behaviour.

Figure 1 gives an example of a step in the functioning of a group of agents in the system. In the presented state, there are still many agents that don't have any data. Consider, however, every agent has sufficient knowledge about his vicinity. For example, agent **I** has the following beliefs about surrounding data:

$\langle I \text{ has } D1 \rangle \langle I \text{ has } D2 \rangle \langle I \text{ has } D4 \rangle$
 $\langle J \text{ has } D4 \rangle \langle J \text{ has } D5 \rangle$
 $\langle E \text{ has } D4 \rangle \langle E \text{ has } D2 \rangle \langle E \text{ has } D3 \rangle \langle E \text{ has } D6 \rangle$
 $\langle F \text{ has } D1 \rangle \langle F \text{ has } D3 \rangle$

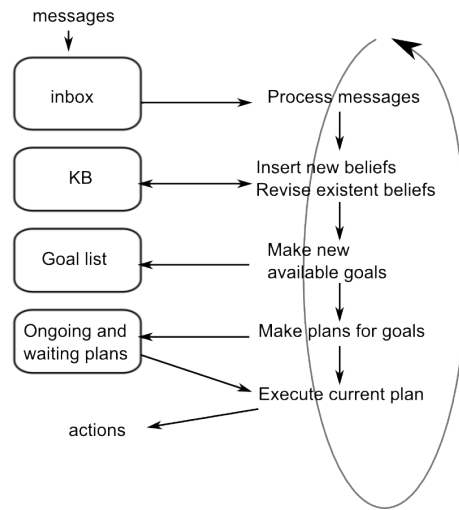


Fig. 2. The basic execution cycle of an agent. The main structural components involved are the knowledge base (KB) and the list of available goals (*Goal list*)

$\langle G \text{ has } D5 \rangle$ (knows that from F)

Each agent that has less than 75% of its capacity full will ask for some piece of data from one of its neighbours. Agent **E** will discard data $D4$, as both **I** and **J** also have it (it has to discard something, so it will discard the data that is most present in its vicinity). Supposing that agent **K** needs $D2$ (for example because there is an external request for it) it will communicate its intentions to its neighbours and **J** will collaborate with it and retrieve $D2$.

The system is using cognitive agents and the implementation is based on the Beliefs-Desires-Intentions model [30]. There are several reasons for using cognitive agents. First, considering our long-term Aml-related goal, there is an obvious necessity for the agents to be cognitive, so that they are capable to aggregate and reason on the information that they hold. But, more important than that, cognitive agents are more capable and can act better in many situations [31]. If their behaviour is flexible, they can even run on resource-constrained devices. Although the behaviour of a cognitive agent system may be harder to control and to predict, the use of cognitive agents can bring many advantages in the future, and the study of emergence in such a system is vital for future research.

The structure of an agent is presented in Figure 2. The working cycle of an agent is straightforward: the agent processes the messages that it receives (there are no other types of perceptions – new data is also inserted into the system as messages); revises its beliefs and its current available goals; chooses a goal and makes a plan for it; executes the actions in the current plan. Details about these steps and the structures involved are given in the sections below.

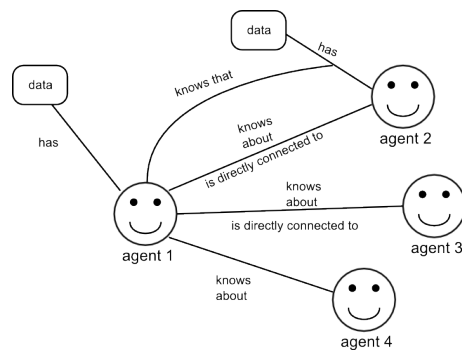


Fig. 3. Example content for an agent's knowledge base (here, the knowledge base of the agent called agent-1). Several relationships are present: *knows about*, *is connected to*, *has*, *knows*. This is only an example and actual agent knowledge bases rise to a greater number of facts.

3.1. Agent Beliefs

The information in the agent's knowledge base (KB) is stored in Facts. In general, a Fact can be either a basic, atomic notion (like *Agent* or *Data*) or a relationship between two other facts. The resulting structure is a concept map, formed of basic notions and relationships between them. At this point in the development of the system, a simplified approach is used, and a Fact may have one of these forms:

- $\langle Agent, knows\ about, Agent \rangle$
- $\langle Agent, is\ directly\ connected\ to, Agent \rangle$
- $\langle Agent, has, Data \rangle$
- $\langle Agent, has, Goal \rangle$
- $\langle Agent, knows, Fact \rangle$

Note that the last definition is recursive, and the recursion must end with one of the other four forms. Most Facts are of the last three types. Considering the structure of Goals (described in the next section) and the agent behaviour, most facts will refer, in the end, to a piece of *Data*.

An example is presented in Figure 3. We consider that the displayed facts are in the knowledge base of the agent named *agent-1*. Relationships of the type "knows" link the agent to all facts that it has: the agents that it knows about, the facts that it knows and what it knows about the knowledge of other agents. Actually, the "knows" relationship between the agent and a fact is equivalent to the fact belonging to the agent's knowledge base, so there is no need to represent it explicitly.

Agents may also know about other agent's goals, so that it is possible for them to cooperate for the realization of goals. *Data* are pieces of information that are not further represented as facts, and that are considered as atomic.

This concept-map-like structure allows agents to represent partial information on the system. It is also possible for agents to easily attach new information

(also represented as a concept map) to their knowledge base, or to send only parts of their knowledge base to other agents. This representation was chosen because it is flexible but also does not consume much space. Even if the presented structure allows very long chains and graphs of $\langle Agent, knows, Fact \rangle$ relationships, the behaviour of the agent has been designed so that information about farther agents will be discarded sooner. Moreover, forgetting may be flexible according to the agent's KB capacity.

3.2. Agent Goals

Following the ideas presented above, and taking inspiration from existing studies of emergent properties in reactive agent systems, the goals of the agents have been designed so that they will be local but will, however, reflect the spirit of the desired global emergents. According to this policy, goals of an agent are of the following types (in parenthesis – the name of the type in the internal representation):

- Get interesting data that the agent knows about and fulfill external requests, if any (*GET*).
- Maintain capacity free, ready for potential data injected from the exterior (*FREE*).
- Inform neighbour agents about new data (*INFORM*).

Apart from that, the agent will also respond to requests for data from neighbour agents, if it has the requested data. This is done before treating any other types of messages, as it does not require any reasoning and is a reactive behaviour.

Goals are represented as pairs of the form $\langle Goal_type, Object \rangle$, where the element *Object* may be the description of a piece of data (its identifier), in the case of *GET*, or a fact about a piece of data, in the case of *INFORM*.

The goal to free capacity has no *Object* and is activated when the free capacity reaches a percentage of the total capacity below a certain threshold. In the experiments presented in this paper, this threshold was 25%. As the pieces of data had equal size, and an agent had a maximum capacity of 4 pieces of data, 25% was chosen as a tradeoff between appropriate use of an agent's capacity and always having room for one more piece of data.

When the used capacity of an agent is under 75% (free capacity is over 25%), it will choose to get a piece of data that is held by a neighbour but not by itself. In order to support the replication of data, it will choose the least well represented in the vicinity (i.e. present in the least number of copies in neighbour agents).

3.3. Agent Behaviour

The types of goals that are presented above will be instantiated according to the beliefs of the agent. For example, the first type of goal (called *GET*) may be

instantiated as $\langle GET, D3 \rangle$, meaning that the agent has the goal to get hold of data $D3$.

Instantiated goals are generated permanently, based on the information that the agent receives, and they are put in the list of *available goals*. This list is sorted according to the *importance* of the goals. Importance is primarily computed depending on the type of the goal. *FREE* is the most important type. Then *INFORM* and *GET* follow. This way, facts about Data will spread first, and then the Data itself will spread. Between different *INFORM* and *GET* goals, more important are the goals referring to Data that is closer (i.e. the chain of *Agent knows Agent knows...Agent has Data* is shorter). From the Goal list the most important goal will be chosen and a plan will be built with actions that the agent expects would lead to the desired result.

In order to fulfill its goals, an agent has the following available actions:

- Send a request for data to a neighbour.
- Discard a piece of data.
- Send a message to a neighbour, containing relevant information (for example that the sending agent has a new piece of data), in the form of facts.
- Send a message to a neighbour, containing a goal of the sending agent, also in the form of a fact.

Say, for example, that agent **A** needs to get data $D3$. If the agent knows that any of the neighbours have the data, i.e. it has a belief $\langle Agent, has, D3 \rangle$, where *Agent* is a neighbour, then it will send a direct request for data $D3$. If *Agent* is not a neighbour, but **A** knows that *Agent* is a neighbour of **B**, and **B** is a neighbour of **A**, it will plan to first send a message containing information about its goal $\langle GET, D3 \rangle$ to **B**, and then put the plan in wait. Eventually it will be informed that **B** has acquired data $D3$. If that doesn't happen in a certain amount of time, **A** will contact **B** and other neighbours and send them its goal. Finally, in the case when **A** does not know of any agent holding data $D3$, it will send the message containing its goal to all the neighbours.

The behaviour of an agent, i.e. the stages that an agent goes through during a step of the system's evolution, is described below:

- **Receive data from and send data to** neighbour agents. Data is transmitted only as a response to previous requests. These operations are simple and need no reasoning, and that is why they are handled separately and before any other decision is made.
- **Revise beliefs.** This is done based on information received from the other agents. Duplicate facts are found and removed. Also, circular facts are identified and discarded.
- **Check waiting plans** for completion (was the goal achieved?). In the case of completion the plan is discarded. Most plans are discarded immediately after all actions are completed. Plans waiting longer than that exist only for goals of type *GET*.

- **Make plans.** Take the goal with the highest importance. If there is no plan for it, make a plan composed of the actions needed to be taken and put it in the list of ongoing plans.
- **Execute plans.** Take the plan associated with the most important goal and execute its next action. If there are no more actions to be performed, move the plan to the list of waiting plans. It will be checked for completion in the next cycle.
- **Fade memory.** Discard old facts and facts that refer to data that is further away (calculated as described above). To prevent the Goal list from overloading, unimportant goals are also discarded.

In order to keep the system in constant change (away from static equilibrium), which is one of the factors that leads to self-organization [18], when there are no more available goals the agent will "lose interest" in a piece of data, the one that is present in a greater number of copies in the neighbours (at least according to the agent's beliefs) and it will discard it. Note that even if an agent will lose interest in data, he is able to regain interest in that data when the data will become more rare in its vicinity. This, of course, leads to a constant movement of data through the system, but is also a necessity for dynamic equilibrium, which in turn helps maintain a robust decentralized system.

Being a complex system (experiments were performed on a system composed of 400 agents), and all agents acting locally, it is clear that the system as a whole will behave, many times, unpredictably. That is why there are several tweaks that had to be performed, across many experiments, in order for the system to function as expected, so that agents do not become overwhelmed with messages, or with available goals, or with useless facts in their KB. These tweaks were implemented in the agents' policies.

First, there must exist a good balance between how fast the agent can learn new knowledge and revise its beliefs and how often an agent broadcasts its own knowledge and intentions to its neighbours. If an agent broadcasts all its knowledge each time it learns something new, all agents will be flooded with messages they don't have time to process. If the agents broadcast their knowledge too rarely, this might result in the agents having outdated beliefs about their neighbours.

Another element that needs good balance is the rate at which an agent forgets. If an agent forgets too quickly, it might have already forgotten some important information at the time it is supposed to be using it. If an agent forgets too slowly, it will have a lot of information that is outdated and of no use anymore.

It is arguable that these tweaks will always depend on the exact application the system is used for, but it is true that in order to achieve self-organization and coherent emergent properties, any complex system must be in a delicate balance between fixed, static equilibrium and chaotic behaviour, so this sort of tuning will still be necessary (the system is not self-tuning).

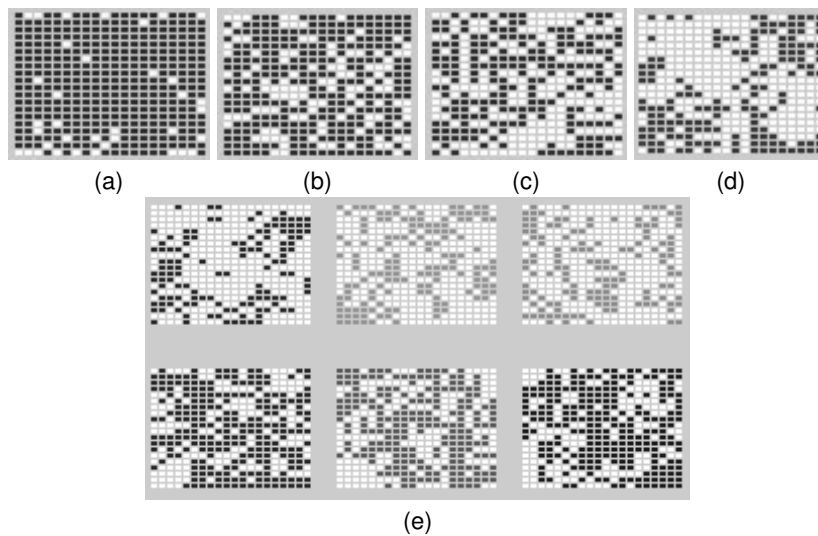


Fig. 4. The distribution of one piece of data after 150 steps, in a system with agents of capacity 4 in the context of a total number of pieces of data of: (a) 3, (b) 4, (c) 6, (d) 8. The simultaneous distributions of 6 pieces of data after 150 steps (e).

4. Results and Discussion

Many experiments were carried out throughout the development of the system, first for assuring that the agents were not overwhelmed by messages and actions to perform, second for the actual study of the evolution of the system. The presented experiments involved a square grid of 400 agents, the pieces of data in the system were of equal size, and each agent had a capacity of 4 pieces of data. As said before, data is generic, and is characterized only by an identifier. The resulting pseudo-stable distributions of data were monitored, but also the dynamic evolution of the system. The purpose of the experiments was to obtain a system that distributes data uniformly throughout the system, while using only local knowledge and interactions.

The system was implemented as a Java application. The execution of the agents is synchronous – at any moment of time all agents are executing the n^{th} step of their evolution. Besides the objectives stated above, one other objective was to make the system run as fast as possible, making the behaviour of individual agents more efficient. This also lead to lower simulation times and the possibility to perform more experiments. A typical experiment lasted 200-250 steps in the multi-agent system's time and about 15 seconds in real time on a machine with an Intel Core 2 Duo 3.33GHz CPU, with 2GB of RAM memory.

Figure 4 (a-d) presents, in several cases, the resulting distribution of a certain piece of data inserted into the system. What changes between the four distributions is the number of other pieces of data that also exist in the system.

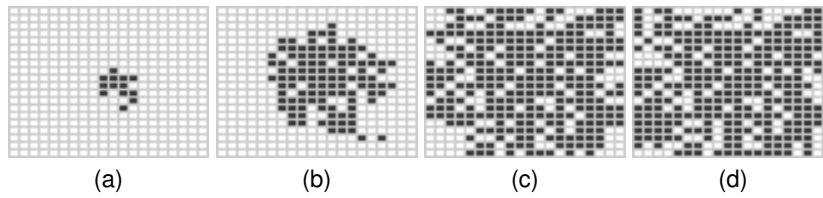


Fig. 5. The distribution of a certain piece of data over several steps of the system's evolution, in a system already holding 4 other pieces of data.

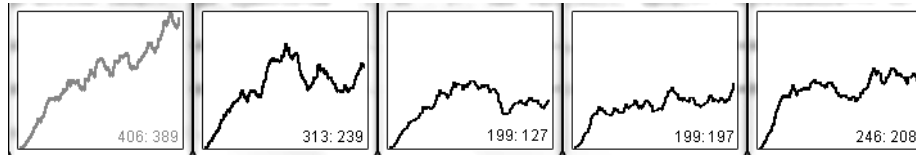


Fig. 6. Each graph corresponds to a piece of data in the system, and represents the evolution of the number of facts about that piece of data (over the whole system), since the beginning of the system's evolution to the current step (step 200). Every image also contains a quantitative indication of the maximum value of the graph as well as the current value.

In the first case (Figure 4(a)) there are only 2 other pieces of data in the system, of equal size. The system is at 75% capacity, so normally all agents should have all three pieces of data. The image reflects that, however, because the system is intentionally unstable, and agents have the ability to forget, there are some agents that do not have the piece of data that was monitored.

In the following two cases (Figure 4(b, c)), featuring other 3, respectively 5 pieces of data in the system, the resulting distribution becomes more scarce, but nevertheless uniform. In the last case (Figure 4(d)) the distribution is less uniform (there is a total of 8 pieces of data in the system), but still the data is in a range of 3 agents from any agent.

It is also interesting to follow how the distribution of one piece of data evolves as the system runs. Figure 5 shows the distribution of a piece of data injected in a system already holding other pieces of data. The distribution grows fairly slow, and no solid distributions are found.

Figure 6 presents the evolution over time of the number of facts (over the whole system) about the 5 pieces of data in the system. This number is slightly larger than the number of agents actually having that data. On these graphs one can easily follow the way the agents become interested in data and then lose interest and forget many of the facts regarding it. Observe that facts about the different pieces of data do not exist in equal numbers throughout the system. That is due to the fact that this is a complex system. Feedback and non-deterministic behaviour result in outcomes that are not fully predictable. However, the numbers are well in the same order of magnitude and considering the little amount of knowledge that each agent has, the distributions are very uniform.

It is relevant to note that the distributions that are displayed in the figures are never stable. The "holes" in the distributions – the agents that do not hold the data – change all the time, as they lose interest, forget, and become interested again. This instability is also shown in Figure 6, where the curves are not smooth, but present many small spikes.

Although all the examples presented here involve the same number of agents, this number can be changed easily and the system scales with no problem, as every agent interacts only with and has knowledge only about its close vicinity.

What is important to point out is that very good distributions are obtained although none of the agents has that as an objective. Moreover, agents are not capable of measuring in any way the distribution of the data in the system, nor do they have any perception on the global state of the system. Their objectives are local and mostly selfish, but a coherent global result is obtained, which is an emergent property. As shown in the previous section, the system and the agents have been specifically designed to produce the emergents, by expressing the desired global result at the local scale of the individual.

The developed system makes a good platform for the study of emergent properties in a relatively simple cognitive agent system. Over the experiments, many monitoring tools have been developed, as following in detail the behaviour and evolution of a system formed by so many agents is not an easy task.

Although the designed system is still very simple, there are important differences between the implemented agents and reactive agents. It must be pointed out that the agents reason permanently about what data they should get and what they should keep. A simple form of collaboration has been implemented – agents are able to share goals. Reactive agents would not have been able to have beliefs about their neighbours and would not have been able to reason about what action would be best in the context.

5. Conclusion

Emergence is an essential notion in the field of multi-agent systems. It provides the possibility of obtaining a more complex outcome from a system formed of individuals of lower complexity. Although there are many recent studies concerning emergence, there is yet no clear methodology on how to specifically design a system so that it would manifest emergence. Moreover, most implementations use reactive agents, that have limited or no cognitive or planning abilities.

The paper presents a cognitive multi-agent system in which the agents' interactions allow the emergence of specific properties required for solving the problem. The system has been designed with the purpose of manifesting emergent properties: the agents have been given local goals that naturally lead to the global, desired, goal of the system.

As future work, the system will be improved, so that the emergents will be more nuanced. Data and facts will have certain context measures relating to

their domains of interest and to their importance. Agents will also be able to have interests in particular domains. New tools for monitoring the system will be developed and the configuration of the network will be able to change – the agents that an agent communicates with will change over time, in order to simulate agent mobility and changes in the environment. Closer to the requirements in Ambient Intelligence, different agents will be able to have different capabilities, different storage capacity and different computing power.

Acknowledgements

This work was supported by CNCSIS - UEFISCSU, project number PNII - IDEI 1315/2008 and Grant POSDRU 5159.

References

1. Amaral, L., Ottino, J.: Complex networks: Augmenting the framework for the study of complex systems. *The European Physical Journal B-Condensed Matter* 38(2), 147–162 (2004)
2. Bernadas, A., Alfano, R., Manzalini, A., Solé-Pareta, J., Spadaro, S.: Demonstrating communication services based on autonomic self-organization. *Proceedings of the 2008 International Conference on Complex, Intelligent and Software Intensive Systems-Volume 00* pp. 101–107 (2008)
3. Beurier, G., Simonin, O., Ferber, J.: Model and simulation of multi-level emergence. *Proceedings of IEEE ISSPIT* pp. 231–236 (2002)
4. Boschetti, F., Prokopenko, M., Macreadie, I., Grisogono, A.: Defining and detecting emergence in complex networks. *Lecture notes in computer science* 3684, 573–580 (2005)
5. Bourjot, C., Chevrier, V., Thomas, V.: A new swarm mechanism based on social spiders colonies: From web weaving to region detection. *Web Intelligence and Agent Systems* 1(1), 47–64 (2003)
6. Cabri, G., Ferrari, L., Leonardi, L., Zambonelli, F.: The LAICA project: Supporting ambient intelligence via agents and ad-hoc middleware. *Proceedings of WETICE 2005, 14th IEEE International Workshops on Enabling Technologies, 13-15 June 2005, Linköping, Sweden* pp. 39–46 (2005)
7. Capera, D., Georgé, J., Gleizes, M., Glize, P.: Emergence of organisations, emergence of functions. In: *AISB03 symposium on Adaptive Agents and Multi-Agent Systems* (2003)
8. De Wolf, T., Holvoet, T.: Emergence versus self-organisation: Different concepts but promising when combined. *Engineering Self Organising Systems: Methodologies and Applications* 3464, 1–15 (2005)
9. De Wolf, T., Holvoet, T.: Design patterns for decentralised coordination in self-organising emergent systems. *Lecture Notes in Computer Science* 4335, 28–49 (2007)
10. Dolev, S., Tzachar, N.: Empire of colonies: Self-stabilizing and self-organizing distributed algorithm. *Theoretical Computer Science* 410(6-7), 514–532 (2009)
11. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.: Scenarios for ambient intelligence in 2010. *Tech. rep., Office for Official Publications of the European Communities* (February 2001)

12. Gleizes, M., Camps, V., Glize, P.: A theory of emergent computation based on cooperative self-organization for adaptive artificial systems. In: Fourth European Congress of Systems Science (1999)
13. Goldstein, J.: Emergence as a construct: History and issues. *Emergence* 1(1), 49–72 (1999)
14. Grasse, P.: La reconstruction du nid et les interactions inter-individuelles chez les *bellicositermes natalensis* et *cubitermes* sp. la thorie de la stigmergie: essai d'interprétation des termites constructeurs. *Insectes Sociaux* 6, 41–83 (1959)
15. Hales, D., Edmonds, B.: Evolving social rationality for MAS using "tags". Proceedings of the second international joint conference on Autonomous agents and multi-agent systems pp. 497–503 (2003)
16. Hellenschmidt, M., Kirste, T.: A generic topology for ambient intelligence. *Lecture notes in computer science* pp. 112–123 (2004)
17. Hellenschmidt, M.: Distributed implementation of a self-organizing appliance middleware. In: Davies, N., Kirste, T., Schumann, H. (eds.) *Mobile Computing and Ambient Intelligence. Dagstuhl Seminar Proceedings*, vol. 05181, pp. 201–206. ACM, IBFI, Schloss Dagstuhl, Germany (2005)
18. Heylighen, F.: The science of self-organization and adaptivity. *The Encyclopedia of Life Support Systems* pp. 1–26 (2002)
19. Koestler, A.: *The Ghost in the Machine*. Macmillan (1968)
20. Mamei, M., Vasirani, M., Zambonelli, F.: Self-organizing spatial shapes in mobile particles: The TOTA approach. In: Brueckner, S., Serugendo, G.D.M., Karageorgos, A., Nagpal, R. (eds.) *Engineering Self-Organising Systems, Methodologies and Applications (after ESOA 2004 workshop)*. *Lecture Notes in Computer Science*, vol. 3464, pp. 138–153. Springer (2004)
21. Mano, J.P., Bourjot, C., Lopardo, G.A., Glize, P.: Bio-inspired mechanisms for artificial self-organised systems. *Informatica (Slovenia): Special Issue: Hot Topics in European Agent Research II Guest Editors: Andrea Omicini* 30(1), 55–62 (2006)
22. Marinescu, D., Morrison, J., Yu, C., Norvik, C., Siegel, H.: A self-organization model for complex computing and communication systems. *Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems* pp. 149–158 (2008)
23. Mills, K.L.: A brief survey of self-organization in wireless sensor networks. *Wireless Communications and Mobile Computing* 7(1), 823–834 (2007)
24. Picard, G.: Cooperative agent model instantiation to collective robotics. In: *Engineering Societies in the Agents World V: 5th International Workshop, ESAW 2004, Toulouse, France, October 20-22, 2004: Revised Selected and Invited Papers*. Springer (2005)
25. Prokopenko, M.: Design vs self-organization. *Advances in applied self-organizing systems* pp. 3–17 (2007)
26. Prokopenko, M. (ed.): *Advances in Applied Self-Organizing Systems*. Springer-Verlag, London, UK (2008)
27. Ramos, C., Augusto, J.C., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. *IEEE Intelligent Systems* 23(2), 15–18 (2008)
28. Randles, M., Taleb-Bendiab, A., Miseldine, P.: Harnessing complexity: A logical approach to engineering and controlling self-organizing systems. *International Transactions on Systems Science and Applications* 2(1), 11–20 (2006)
29. Randles, M., Zhu, H., Taleb-Bendiab, A.: A formal approach to the engineering of emergence and its recurrence. *Proc. of EEDAS-ICAC* pp. 1–10 (2007)

Andrei Olaru, Cristian Gratie, and Adina Magda Florea

30. Rao, A.S., Georgeff, M.P.: BDI agents: From theory to practice. In: Lesser, V.R., Gasser, L. (eds.) Proceedings of the ICMAS-95, First International Conference on Multiagent Systems, June 12-14, San Francisco, California, USA. pp. 312–319. San Francisco, CA, The MIT Press (1995)
31. Ricci, A., Omicini, A., Viroli, M., Gardelli, L., Oliva, E.: Cognitive stigmergy: Towards a framework based on agents and artifacts. *Lecture Notes in Computer Science* 4389, 124–135 (2007)
32. Rodriguez, S., Gaud, N., Hilaire, V., Galland, S., Koukam, A.: An analysis and design concept for self-organization in holonic multi-agent systems. *Lecture Notes in Computer Science* 4335, 15–27 (2007)
33. Sen, S., Airiau, S.: Emergence of norms through social learning. Proceedings of the Twentieth International Joint Conference on Artificial Intelligence pp. 1507–1512 (2007)
34. Serugendo, G.D.M., Gleizes, M.P., Karageorgos., A.: Self-organization and emergence in MAS: An overview. *Informatica* 30(1), 45–54 (2006)
35. Standish, R.: On complexity and emergence. Arxiv preprint nlin.AO/0101006 pp. 1–6 (2001)
36. Unsal, C., Bay, J.: Spatial self-organization in large populations of mobile robots. Proceedings of the 1994 IEEE International Symposium on Intelligent Control pp. 249–254 (1994)
37. Zambonelli, F., Gleizes, M., Mamei, M., Tolksdorf, R.: Spray computers: Frontiers of self-organization for pervasive computing. Proceedings of the 13th IEEE Int'l Workshops on Enabling Technologies, WETICE pp. 403–408 (2004)