Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 857512, 19 pages
doi:10.1155/2012/857512

Research Article

Emergent Behaviors in Social Networks of Adaptive Agents

Florin Leon

Faculty of Automatic Control and Computer Engineering, "Gheorghe Asachi" Technical University of Iasi, Boulevard Mangeron 27, 700050 Iasi, Romania

Correspondence should be addressed to Florin Leon, florinleon@gmail.com

Received 24 June 2011; Accepted 13 July 2011

Academic Editor: Carlo Cattani

Designing multiagent systems that can exhibit coherent group behavior based on a small number of simple rules is a very challenging problem. The evolution of mobile computing environments has created a need for adaptive, robust systems, whose components should be able to cooperate in order to solve the tasks continuously received from users or other software agents. In this paper, an interaction protocol for a task allocation system is proposed, which can reveal the formation of social networks as an emergent property. The agents can improve their solving ability by learning and can collaborate with their peers to deal with more difficult tasks. The experiments show that the evolution of the social networks is similar under a great variety of settings and depends only on the dynamism of the environment. The average number of connections and resources of the agents follows a power law distribution. Different configurations are studied in order to find the optimal set of parameters that leads to the maximum overall efficiency of the multiagent system.

1. Introduction

Complexity theory is a relatively recent research field which mainly studies the way in which critically interacting components self-organize to form potentially evolving structures exhibiting a hierarchy of emergent system properties [1]. Thus, complexity theory includes the new domain of self-organizing systems, which searches for general rules about the forms and evolution of system structures, and the methods that could predict the potential organization resulting from changes to the underlying components [2]. It is hypothesized that a dynamic system, independent of its type or composition, always tends to evolve towards a state of equilibrium, that is, an attractor. This process decreases the uncertainty about the system state and therefore the system's entropy, leading to self-organization [3].

Four necessary requirements have been identified for systems to exhibit a self-organizing behavior under external pressure [4].

(i) *Mutual causality:* at least two components of the system have a circular relationship, each influencing the other.

(ii) *Autocatalysis:* at least one of the components is causally influenced by another component, resulting in its own increase.

(iii) *Far from equilibrium condition:* the system imports a large amount of energy from outside the system, uses it to renew its own structures (autopoiesis), and dissipates rather than accumulates the increasing entropy back into the environment.

(iv) *Morphogenetic changes:* at least one of the components of the system must be open to external random variations from outside the system, and the components of the system are changed [5].

Currently, many applications of self-organizing systems have been proposed [6], among which we could mention the self-organization of broker agent communities for information exchange [7], solutions to constraint satisfaction problems such as the automated timetabling problem [8], flood forecasting [9], land use allocation based on principles of eco-problem solving [10], and traffic simulation using holons [11]. An interesting approach is the use of the emergent structures given by Conway's game of life [12] to define functional components such as adders, sliding block-based memories, and even the pattern of a Turing machine [13].

1.1. Emergent Properties

The emergent properties are characteristic of complex systems, which usually are formed of a relatively small variety of fairly homogenous agents acting together according to simple local rules of interaction and communication, with no global or centralized control. Despite their homogeneity, complex systems are irreducible, because emergent global properties result from agents operating in a self-organizing manner [14]. Emergent properties are often associated with properties such as fault tolerance, robustness, and adaptability.

Surprise or novelty is characteristic of emergence. It can be stated that the language of low-level design L and the language of high-level observation H are distinct, and the causal link between the elementary interactions programmed in L and the behaviors observed in H is nonobvious, and therefore surprising, to the observer [15]. Thus, emergence can be considered a process that leads to the appearance of structure not directly described by the defining constraints and forces that control the system, and over time something new appears at scales not directly specified by the local rules of interaction [16].

Some researchers studied the possibility of emergence engineering [17], that is, designing the local rules such that a desired high-level behavior should be produced. They concluded that different methodologies than the usual software development should be employed in order to get close to this desideratum, because the main idea is to implement or generate the system without actually knowing how it works. They suggest three approaches in this respect:

(i) by imitating phenomena usually considered as emergent;

(ii) by using an incremental design process and incrementally adapting the system to make it produce a behavior that will be closer to the behavior one ultimately expects;

(iii) by developing self-adaptive systems.

1.2. Complex Networks

Many complex phenomena are related to a network-based organization. Complex networks are often encountered in the real world, for example, technological networks such as the power grid, biological networks such as the protein interaction networks or the neural network of the roundworm *Caenorhabditis elegans*, and social networks such as scientific collaboration of human communication networks [18]. There are also many other types of physical or chemical systems that demonstrate nonlinear behavior [19–21].

Many real complex networks share distinctive features that make them different from regular (lattice) and random networks. One such feature is the "small world" property, which means that the average shortest path length between the vertices in the network is small; it usually scales logarithmically with the size of the network, and the network exhibits a high clustering effect. This model proposed by Watts and Strogatz [22] was designed as the simplest possible model that accounts for clustering while retaining the short average path lengths of the Erdős and Rényi model [23].

A well-known example is the so-called "six degrees of separation" in social networks [24]. Another is the scale-free property of many such networks, that is, the probability distribution of the number of links per node $P(k)$ satisfies a power law $P(k) \sim k^{-\gamma}$ with the degree exponent γ in the range $2 < \gamma < 3$ [25]. The World Wide Web network has been shown to be a scale-free graph [26].

The generation of scale-free graphs can be performed based on two basic rules [27].

(i) *Growth:* at each time step, a new node is added to the graph.

(ii) *Preferential attachment:* when a new node is added to the graph, it attaches preferentially to nodes with high degrees.

Complex networks can also reveal self-similar properties, and thus their fractal dimension could be computed using methods such as "box counting" and "cluster growing" [28].

1.3. Agent-Based Simulation of Complex Phenomena

Software agents can be naturally used to play the role of autonomous entities capable of self-organization. Agents are appropriate to simulate such systems, in order to better understand models or create new ones [29]. However, agents can be increasingly useful for the development of distributed systems whose components can self-organize and work in a decentralized manner for the realization of the global functionality [30].

The ability of a multiagent system to dynamically reorganize its structure and operation at run-time is highly valuable for many application domains. Therefore, allocating tasks to agents in multi-agent systems is a significant research issue. There are two main directions of study in this respect. On the one hand, the centralized methods for task allocation assume that there is a central controller to assign tasks to agents [31]. The centralized approach can make the allocation process efficient and effective in a small network since the central planner has a global view of the system and it knows which agents are good at which tasks. On the other hand, the decentralized style is more scalable and robust but the communication overhead increases. Some proposed mechanisms form groups of agents before allocating tasks which may result in the increase of computation and communication cost [32]. Other distributed protocols only allow agents to request help for tasks from their directly linked neighbors which may increase the possibility of task allocation failure because of limited resources available from the agents' neighbors [33]. Another technique is to address the task allocation problem in a loosely coupled peer-to-peer system [34]. Other researchers aim at minimizing the team cost subject to the constraint that each task must be executed simultaneously by a given number of cooperative agents which form coalitions [35]. Task allocation can also be observed in the biological world, for example, ants of different sizes take on tasks of different difficulties: larger ants carry larger food, while smaller ones perform more manageable tasks [36].

2. Model Description

The proposed multiagent system is composed of a set of agents A, which are "physically" distributed over a square grid. However, this localization does not prevent agents from forming relations with any other agents, based on the common interest of solving tasks, as it will be shown in Section 3.

The tasks are considered to be defined by a series of p attributes. A task has specific values of these attributes, considered to be *complexity levels*, each within a certain domain. Let T be the set of tasks $T = \{t_i\}$ and F the set of attributes or features $F = \{c_j\}$. Then:

$$t_i = \{c_1, \ldots, c_p\}, \tag{2.1}$$

with $c_j \in D_j$, for all $j \in \{1, \ldots, p\}$ and $p = |F|$.

Each agent has *competence levels* l_a^j associated with each attribute j, which describe how much knowledge an agent has regarding a particular feature of a task.

For example, in a software industry environment, attributes can relate to the use of databases, developing graphical user interfaces, or working with specific algorithms. A programming task can contain such features in varying degrees, with different complexity levels. An agent specialized on databases has a high competence level for the first attribute, although these tasks may also contain issues related to the other attributes, possibly with lower complexity levels. However, if other types of tasks should be addressed, the agent can gradually specialize in other development areas.

The tasks are generated in the following way. First, the number of nonnull attributes $p_{nn}^i \in \mathbb{N}$, $1 \le p_{nn}^i \le p$, is determined, by using a power law or a uniform distribution:

$$P\left(p_{nn}^i\right) \approx \frac{1}{\left(p_{nn}^i\right)^k} \tag{2.2}$$

or

$$P\left(p_{nn}^i\right) \approx U(1, p). \tag{2.3}$$

When using the power law distribution, most of the tasks will have one or two non-null attributes, and therefore they will have a large degree of specialization. Thus, agents can specialize in performing some type of tasks. When using a uniform distribution, there will be a larger number of "interdisciplinary" tasks; therefore the agents will have to interact more in order to solve them in a cooperative way.

Based on the distribution of non-null attributes, tasks are continuously generated during the simulation, and the corresponding values of the non-null attributes are generated from a uniform distribution:

$$c_j \sim U(1,\ L_{\max}), \tag{2.4}$$

where $L_{\max}$ is the maximum complexity level allowed in a certain simulation epoch. In Section 4, we will analyze the situations where $L_{\max}$ is constant throughout the whole simulation, or gradually increases.

When an agent receives a task, it first verifies whether the complexity levels of the attributes are less or equal to its respective competence levels: $c_j \leq l_a^j$, for all $j = 1, \ldots, p$.

In this case, the agent can solve the task by itself and receives a monetary payment of:

$$M_a^i = \sum_{j=1}^{p} c_j^2. \tag{2.5}$$

If the complexity level of a task attribute is greater by 1 than the corresponding competence level, then the agent can learn in order to reach the proper level. There are two prerequisites. First, each agent has an individual "willingness to learn" $W_a^l \sim U(W_{\min}, W_{\max})$. The probability to pass from a competence level to another is given by the following probability:

$$P_a^l = \left(W_a{}^l\right)^{l_a^j}. \tag{2.6}$$

Accordingly, the probability to go from a low level to the next is much higher than the probability to go from a high level to the next. For example, if $W_a^l = 0.95$, the probability to go from level 1 to level 2 is 0.95, while the probability to go from level 9 to level 10 is 0.63.

Secondly, a payment is required of the agent for the learning step:

$$M_a^l = c_j^2. \tag{2.7}$$

For this process to work, each agent has an equal initial amount of "money" M^0.

If the agent is fit to solve the whole task following a possible learning phase, the agent solves it and it receives the same payment as that described in (2.5).

If the agent is unable to solve a task by itself, it seeks other agents to solve the parts of the task it cannot handle by itself.

3. Interaction Protocol and Social Network Formation

The environment randomly distributes a number of tasks fewer than the number of agents in the system: $|T| < |A|$. Since some agents will be able to solve their tasks, either individually or cooperatively, while others will not, in the subsequent epochs, tasks will be given preferentially to agents that previously succeeded more. Thus, each agent has a probability to solve tasks defined as

$$P_a^s = \frac{T_a^s}{T_a^r}, \tag{3.1}$$

where T_a^s is the number of tasks solved by agent a and T_a^r is the total number of tasks received by agent a.

In this setting, the agents are sorted by their probabilities to solve tasks, P_a^s, and only the first $|T|$ receives new tasks. Ties between agents with the same P_a^s are broken randomly. The initial values are $T_a^s = 1$, $T_a^r = 2$, for all $a \in A$, and therefore the initial probability to solve is $P_a^s = 0.5$, for all $a \in A$.

However, the agents who fail to solve tasks at first become disadvantaged because they no longer receive new tasks and therefore they no longer have the direct incentives to improve their competence levels. For this reason we introduced a perturbation rate R, such that, when $R = 0$, the tasks are distributed deterministically, as presented above, and, when $R = 1$, tasks are randomly distributed.

Each agent has a set of connections to other agents, that is, "friends," initially void. When an agent cannot solve a task by itself, it begins a breadth-first search on the connection graph. As agents are "physically" situated on a lattice, the immediate neighbors are always added to the search queue after the immediate friends. Search is performed without allowing agents to be visited twice in order to avoid infinite loops.

Every time an agent is not able to solve the originator's task either directly or by learning, its own friends are added to the search queue. In this way, the entire graph is ultimately explored, and the closest agents are always selected to solve the tasks, since breadth-first search is complete and optimal. We ignore the exponential memory requirements of the algorithm because we consider that this aspect is irrelevant to our study; the graph itself has quite a small depth, similar to the networks that exhibit the "small world" phenomenon.

Once an agent is found which can solve one or more attributes of a task, and only when the whole originator's task can be solved, the agents on the path from the originator to the solver are sequentially connected. Finally, the originator and the solver are directly connected.

Several aspects must be underlined: there is no limitation to the number of tasks an agent can solve in one round. It may have only one individually assigned task, but several task attributes received as subcontracting. The originator agent u receives a commission rate α for the subcontracted task attributes, and the solver agent v receives the rest:

$$\begin{aligned} M_u' &= M_u + \alpha \cdot \sum_{j\in\Delta} c_j^2, \\ M_v' &= M_v + (1-\alpha) \cdot \sum_{j\in\Delta} c_j^2, \end{aligned} \tag{3.2}$$

where Δ is the set of task attributes transferred to agent v for solving.

There is no payment for the intermediary agents, and this is one of the main differences between this approach and the classical contract net protocol [37]. In this paper, we consider

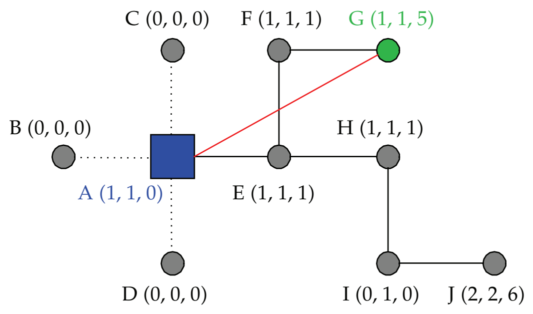

Figure 1: Illustration of the interaction protocol and formation of new connections.

that the most important feature of the social network is to *transfer information*, not actual solutions for tasks accompanied by monetary payments.

Once a task is solved, it is considered to be solved by the originator agent u, whose P_u^s increases. Every connection, once established, is active a limited number of epochs θ_T after which it disappears. If an agent does not succeed in solving a task during one epoch and does not receive another one in the next epoch, it can keep its task for a maximum number of epochs θ_C. If an agent cannot eventually solve a task, its P_u^s consequently decreases.

Figure 1 presents a small example that can illustrate the interaction protocol. Let us assume that agent A receives a task with complexity levels $\{2,\ 1,\ 5\}$. Its own competence levels are $\{1,\ 1,\ 0\}$, therefore

(i) it can learn to solve attribute 1, If the learning process is successful, its competence levels will become $\{2,\ 1,\ 0\}$,

(ii) it can directly solve attribute 2,

(iii) it must search for another agent to solve attribute 3.

Finding a capable agent for attribute 3 involves a breadth-first search (BFS) process. The agents connected with agent A are automatically added to the search queue, in this case agent E. The other neighbors are also added to the queue, that is, B, C, and D. The neighbor links are not considered social network connections, but, in case an agent is isolated at a certain point, its physical neighbors may help it to find other agents. Therefore, its start queue is $\{E,\ B,\ C,\ D\}$. As search nodes are expanded, the “friends” of the agents are added to the end of the queue. In the second stage, the queue becomes $\{B,\ C,\ D,\ F,\ H\}$. The nodes corresponding to the agents already visited are not expanded again. As B, C, and D have no new connection, the node corresponding to the F agent is expanded and the queue becomes $\{H,\ G\}$. So far, no agent expanded is capable of solving attribute 3. Then, H is expanded, and the queue is $\{G,\ I,\ J\}$. Next, G is expanded and it is capable of solving attribute 3. Therefore, A and G become directly connected, and A pays G a fraction $1 - \alpha$ of the price for solving attribute 3.

Although agent J is also capable of solving attribute 3, the optimality of the BFS algorithm ensures that the nearest capable agent (in the social network, not necessarily in the physical environment) is always selected.

4. Parameter Influence on System Behavior: Case Studies

In this section, the results of some simulations are presented, which illustrate the behavior of the multiagent system. We focus on the common patterns that emerge from the social network created by the system in execution, while varying some of the parameters of the simulation.

For all the case studies, the following parameters remain unchanged. The initial amount of "money" available to the agents $M^0 = 100$. The limits of the "willingness to learn" are $W_{\min} = 0.9$ and $W_{\max} = 1$. The number of epochs while a connection is active after being created is $\theta_T = 5$. The maximum number of epochs while an agent can keep a task without being able to solve it is $\theta_C = 3$. The total number of tasks distributed during every epoch is $|T| = |A|/2$.

Figure 2 shows the evolution of the social network after 10, 100, 350, and 500 epochs, from left to right, top to bottom, respectively. The maximum complexity level $L_{\max} = 10$ remains constant throughout the simulation. As it can be seen, the number of agents is 400, arranged on a grid of size 20×20. The perturbation rate $R = 0$, the commission $\alpha = 0.1$, and the non-null attributes follow a power law distribution (2.2) with exponent $k = 2$. One can notice that at the beginning local connections form on short physical distances on the grid. Gradually, these small clusters connect and global networks emerge. Depending on the initial conditions (the actual values of task complexity levels, agent competence levels, and the first distribution of tasks), which are randomly generated and can vary from a simulation to another, one global network or two/few networks can appear. Their number also greatly depends on the number of agents (reflected on the size of the grid). After a great increase, the number of connections gradually decreases, because the agents tend to learn and adapt to the unchanging environment, depicted by a constant L_{max}. Thus, as agents increase their individual competence levels, they no longer need other agents to solve their tasks. Finally, when most of the agents become independent, the number of active connections approaches 0.

Figure 3 shows the status of the multiagent system after 100 epochs, with the same parameters, when the number of agents varies (100, 256 and 2500), in order to demonstrate the scaling behavior of the system. One can see that the formation of global social network(s) is similar. Also, the evolution of the system as the number of epochs increases is the same: the agents enhance their competence and no longer need their peers, and therefore the connections go through a gradual dissolution process.

This tendency of the system is altered when the maximum complexity level of the tasks is no longer constant but grows as the simulation proceeds. Figure 4 shows the status of the multiagent system after 500 epochs, when the environment is dynamic and $L_{\max}$ constantly increases: $L_{\max} = 10 + N_{\text{epochs}}/20$. Because the competence levels of the agents no longer become saturated, the shape of the social network is maintained throughout the simulation and its dissolution no longer takes place.

This situation is more similar to the real-world problem of task allocation, where the environment continuously changes and individuals must change as well in order to stay adapted and solve the increasingly competitive tasks they are faced with. It is this individual change and adaptation itself that causes the dynamism of the environment, due to the interconnectedness of the individuals.

Figure 5 shows the status of the social network after 100 epochs, when the perturbation rate $R = 0.5$. When the perturbation rate was 0, the agents who solved the tasks were preferred when assigning new tasks. In this case, the evolution of the system greatly depended on the initial assignment of tasks. While unsuccessful agents were eliminated from receiving new tasks, it is likely that some competent agents remained in the system without being assigned in the first phases of the simulation, and this also decreased their chances to being assigned later on. With a 50% randomization, as all the agents have a chance to receive tasks, the social

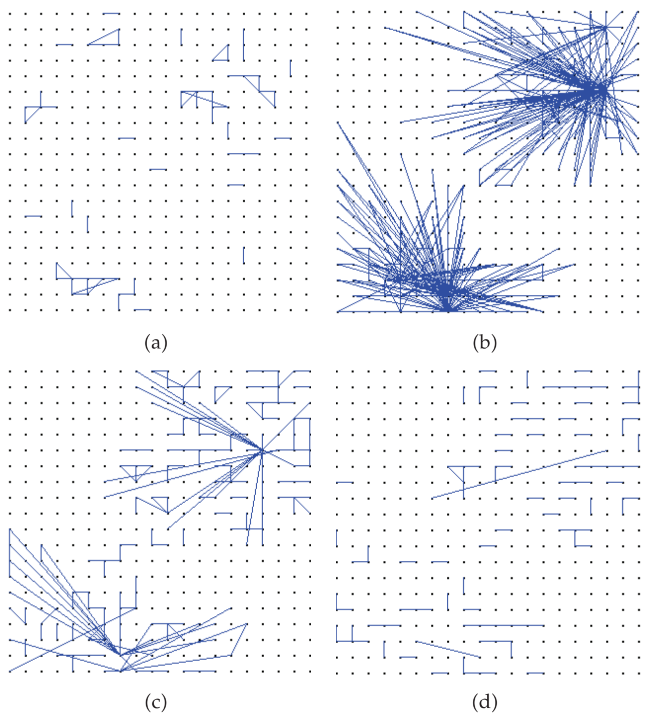

Figure 2: The evolution of the social network after 10, 100, 350, and 500 epochs when the environment is static.

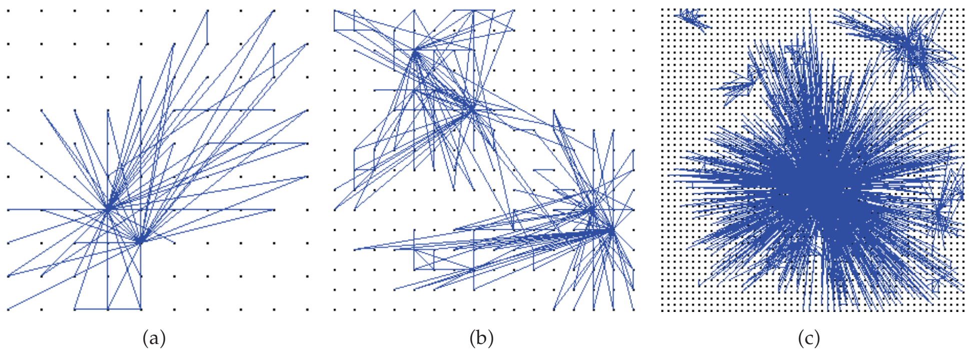

Figure 3: The status of the social network after 100 epochs for different numbers of agents: 100, 256, and 2500.

network is greatly extended. Because the system is evolving, with the same linear increase of $L_{\max}$ in the number of epochs passed, the shape of the system remains similar later in the simulation.

Figure 6 shows the social network of the agents after 100, 350, and 500 epochs, respectively, when the environment is not dynamic ($L_{\max} = 10$ and remains constant) and the perturbation rate is $R = 0.5$. The resulting behavior is an intermediate one between those displayed in Figures 2 and 5. Although more connections are formed, the agents still tend to saturate their competence levels. However, the random perturbation involves more agents in

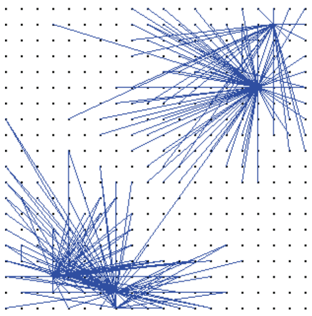

Figure 4: The status of the social network after 500 epochs when the environment is dynamic.

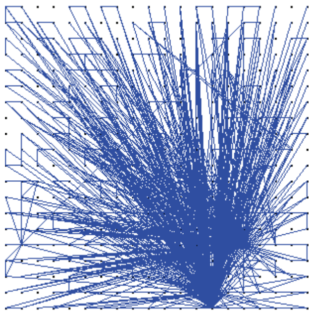

Figure 5: The status of the social network after 100 epochs when the perturbation rate $R = 0.5$.

the task allocation process, and, therefore, connections still remain active after 500 epochs, but in a different, mostly neighbor-based configuration, because it becomes increasingly likely that a competent agent will be found in the physical neighborhood of an agent.

Figure 7 shows the social network after 100, 350, and 500 epochs, respectively, when the environment is dynamic ($L_{\max} = 10 + N_{\text{epochs}}/20$), and the perturbation rate is $R = 1$. In this case, the social network covers almost all the agents, and this configuration remains relatively stable until the end of the simulation.

The overall behavior of the multiagent system is similar when the task attributes are generated following a uniform distribution (2.3). When the environment is dynamic (Figure 8), the network configuration is preserved. When it is static (Figure 9), the connections eventually dissolve. However, compared to the power law distribution case, here the evolution of the system is faster, because the tasks involve more cooperation on the part of the individual agents, and therefore the agents specialize more quickly on the different types of attributes. After only 350 epochs, almost all the connections disappear.

In the following paragraphs, we will analyze some of the statistical properties of the multiagent system, taking into account the way in which the average number of connections, the average wealth, and the average competence levels of the agents vary during 500 epochs of the simulation.

In Figure 10, the evolution of the average number of connections is displayed, when the type of environment and the type of task attribute generation differ: evolving (dynamic) or nonevolving (static) for the former and power law or uniform distribution for the latter. The shape of the function is similar in all four cases, with a sharp increase toward the beginning of the simulation. As it was shown above, the uniform distribution delays the

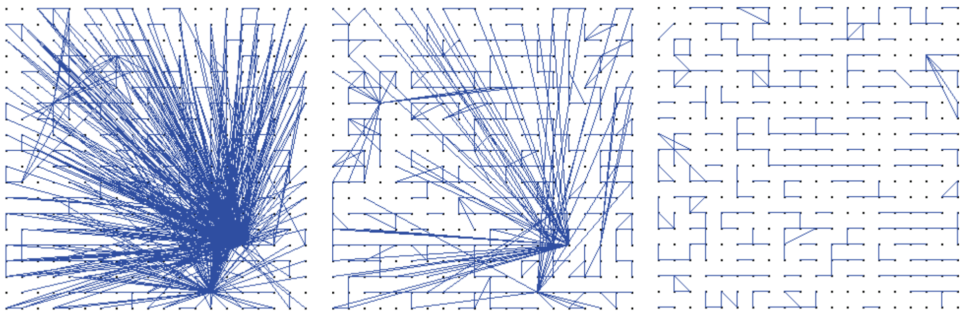
(a) (b) (c)

Figure 6: The evolution of the social network after 100, 350, and 500 epochs when the environment is static and the perturbation rate $R = 0.5$.

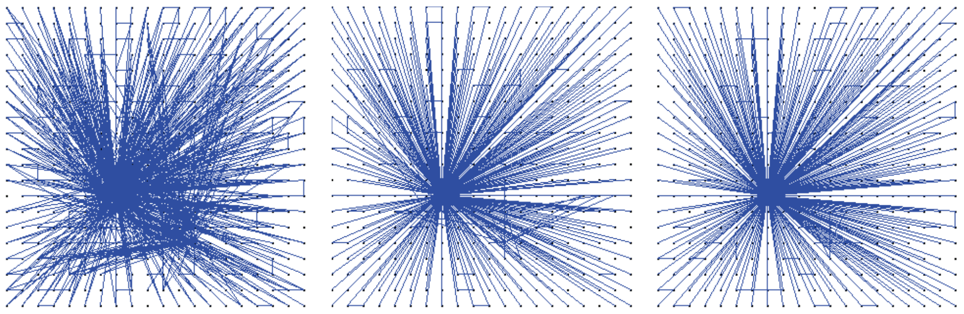
(a) (b) (c)

Figure 7: The evolution of the social network after 100, 350, and 500 epochs when the environment is dynamic and the perturbation rate $R = 1$.

formation of the maximum sized network however, when the network is formed, it has a greater number of connections. As the system converges, the average number of connections in the evolving setting is greater than that for the power law distribution. Also, for the non-evolving case, the average number of connections decreases more quickly than in the power law distribution scenario.

These average values show that only a few agents have a great number of connections, while most of the agents have one or no active connections at all. In fact, the histogram of the number of agents having a certain number of connections reveals a power law distribution, as shown in Figure 11. The ordinate axis (the number of agents) has a logarithmic scale. The abscise axis shows the five equal size intervals ranging from 0 to the maximum number of connections.

Regarding the evolution of the average competence levels, Figure 12 displays a typical situation. It is important to note that the shape of this function is approximately the same in all scenarios.

The evolution of the maximum competence level (Figure 13) has a similar profile at the beginning of the simulation, but it grows fairly linear, in small increments, unlike the average value, which resembles a logarithmic expansion. Therefore, it is clear that the difference between the top agents and the rest will become increasingly larger over time, and this will amplify the disparity concerning the resource allocation in the system.

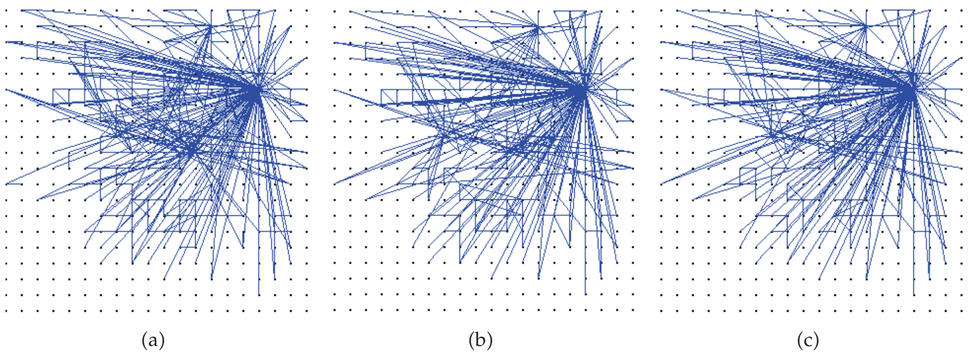

Figure 8: The evolution of the social network after 100, 350, and 500 epochs when the task attributes are generated from a uniform distribution and the environment is dynamic.

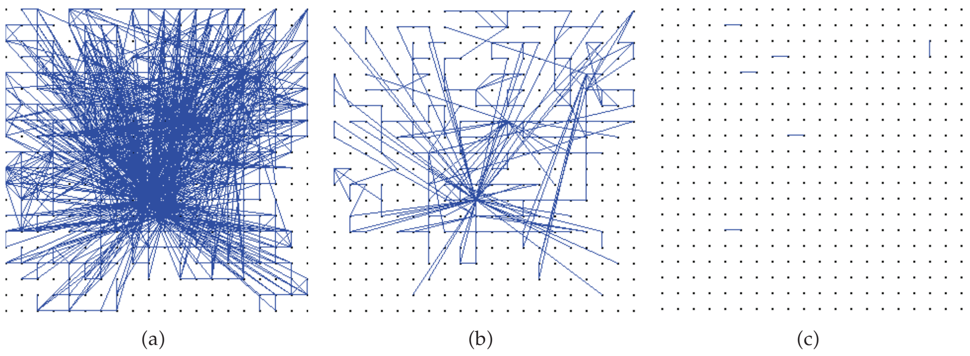

Figure 9: The evolution of the social network after 35, 100, and 350 epochs when the task attributes are generated from a uniform distribution and the environment is static.

An interesting fact was observed, that the "willingness to learn" W_a^l is not crucial to become a top agent. There are "best agents" with the greatest competence levels and the most money, which do not have $W_a^l = 1$, but even a value close to $W_{\min}$. It seems that the most important fact is the number of opportunities to learn (by receiving tasks directly or indirectly, from other agents in its social network), which forces the agent to eventually learn and become the most competent.

Regarding the average wealth of the agents, which results from the monetary payments for solving tasks, a similar distribution is encountered: there is one (or very few agents) on top, with a large amount of money, and most agents are on the bottom of the scale. Since all the agents start with an equal wealth, at some point in the simulation the ratio reverses, a phenomenon displayed in Figures 14 and 15. The figures show the histogram with five intervals, from 0 to the maximum amount of money, and the bars show on the ordinate the number of corresponding agents on a logarithmic scale. At the beginning (time frame 1, but not epoch 1), most agents are in internal *I5*, close to the maximum. During only a few epochs, the majority of the agents lose their relative wealth compared to the top agents and move into interval *I1*, close to the minimum. The phenomenon takes place more quickly when the commission $\alpha = 0.1$ (Figure 14) than when the commission $\alpha = 0.95$ (Figure 15). Therefore, a higher commission leads to a more egalitarian distribution of the wealth in the first phase of the simulation although, in the end, the results are similar.

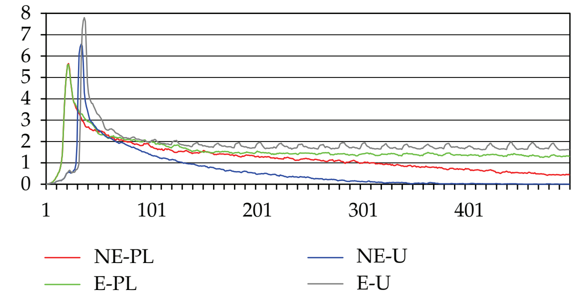

Figure 10: The evolution of the average number of connections.

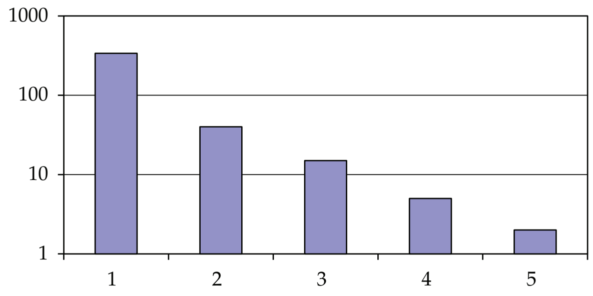

Figure 11: Power law distribution of the number of connections.

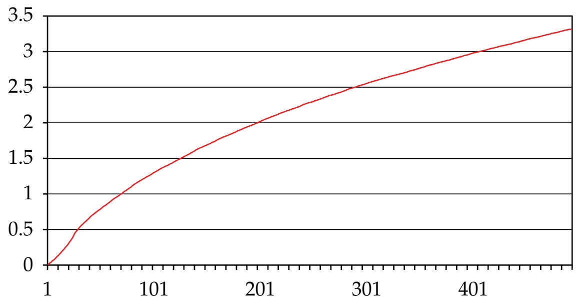

Figure 12: The evolution of average competence levels.

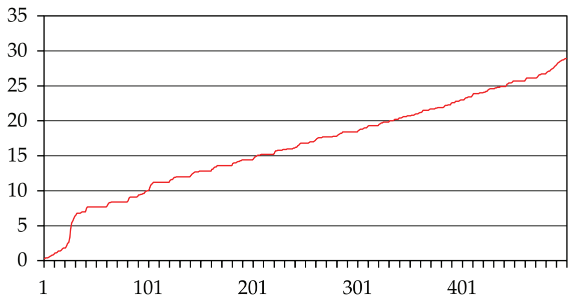

Figure 13: The evolution of maximum competence levels.

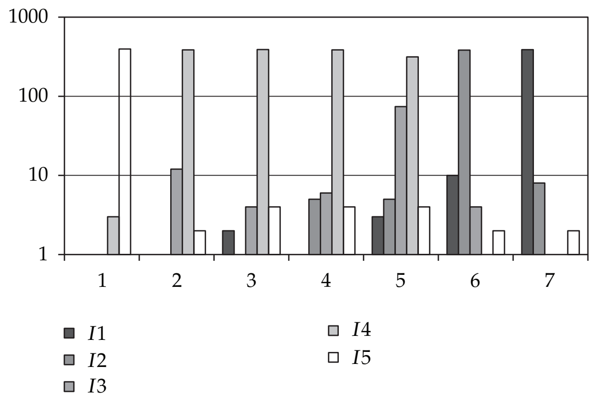

Figure 14: The change histogram of the average wealth with a commission rate $\alpha = 0.1$.

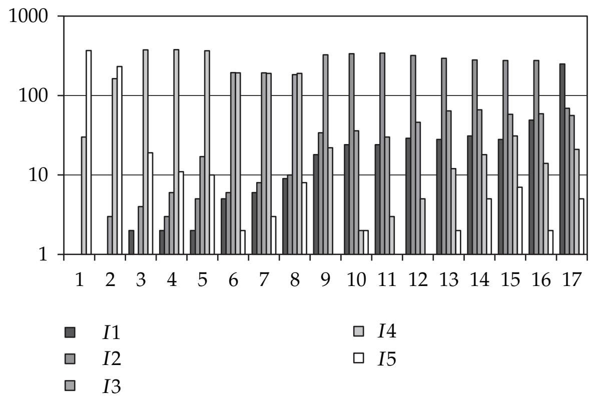

Figure 15: The change histogram of the average wealth with a commission rate $\alpha = 0.95$.

Also, it was observed that when the number of tasks decreases (e.g., from 50% to 20% of the number of agents), the resulting wealth distribution is more balanced.

It is important to note that these figures show only relative values. The absolute value of the money actually increases for all the agents involved.

Next, we address the reciprocal relations between the statistical quantities under study.

In Figure 16, the average wealth of the agents is displayed as a function of the average competence level. It can be seen that the amount of money increases exponentially with knowledge, because an agent with higher competence can receive and solve tasks from all the agents in its social network.

Figure 17 presents the average number of connections as a function of the average competence level. At the beginning, the number of connections must increase if the agents have low competence levels because they must find other agents to help them solve their tasks. As the average knowledge increases, the number of connection decreases and stabilizes because the agents become more independent and can solve the tasks on their own.

The shapes of the functions in Figures 16 and 17 are the same irrespective of the parameters used in the simulation.

Finally, we analyze the efficiency of the overall task allocation system, defined as the average probability to solve tasks P_a^s of all the agents in the system.

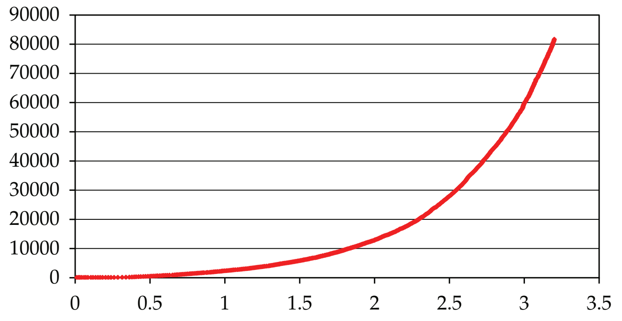

Figure 16: The average amount of money as a function of the average competence level.

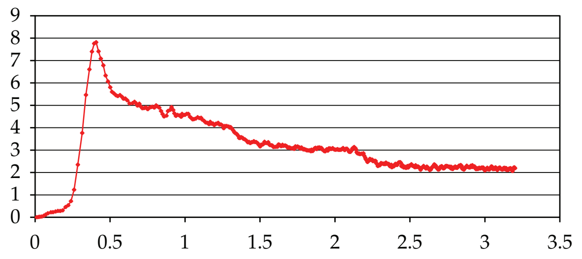

Figure 17: The average number of connections as a function of the average competence level.

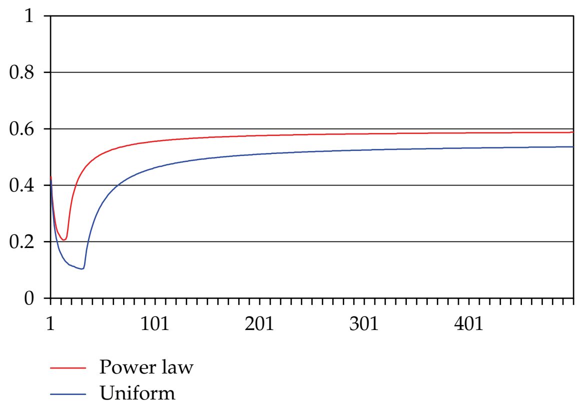

Figure 18: The evolution of the overall system efficiency with different distributions for task attribute generation.

In Figure 18, one can notice that the efficiency is slightly larger when the task attributes are generated using a power law, compared to the uniform distribution. This is because tasks generated in this way are "easier," as they require less collaboration or learning for solving them.

Figure 19 displays the evolution of the overall efficiency for different values of the perturbation rate R. Although giving tasks preferentially to previously successful agents could seem like a useful heuristic, in fact it appears that the inclusion of more agents in the task solving process leads to better global results. It is especially important to keep the perturbation rate positive $R > 0$ because the difference between the cases when $R = 0.5$ and

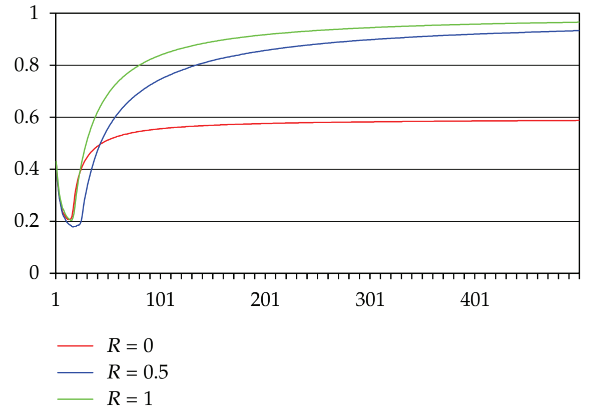

Figure 19: The evolution of the overall system efficiency for different perturbation rates R.

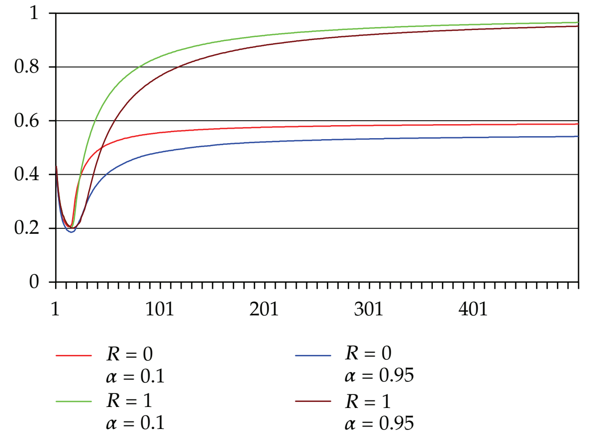

Figure 20: The evolution of the overall system efficiency for different perturbation rates R and commission rates α.

$R = 1$ is smaller than the difference between the cases when $R = 0$ and $R = 0.5$. A small change in the preferential allocation mechanism can lead to a greater change in the overall efficiency. Therefore, this is another nonlinear effect of the model.

When the commission is considered, it can be seen in Figure 20 that the efficiency is a little larger when the commission is smaller. This great change in the way in which money is distributed between the originator and the solver agent could be expected to have a great effect in the overall wealth distribution and system efficiency (i.e., the most money could belong to mediocre agents). However, it was found that, even when the commission is high, the competent agents eventually have the greatest wealth. Therefore, it seems that the proposed protocol is very robust to speculations.

5. Conclusions

According to the proposed model, the agents can gradually enhance their knowledge by learning when presented with tasks to be solved. They also manifest a collaborative behavior for solving more complex problems. The social network acts as a means to transfer information, unlike the classic contract net protocol which transfers actual tasks and "money" for subcontracting. The model also includes monetary payments for agents which successfully accomplish their goals. Under different configurations of the environment, it is shown that the agents form evolving social networks and the system exhibits several types of emergent properties:

(i) a power law distribution of the number of connections and agent "wealth";

(ii) a dynamic or evolving environment is required to maintain the social networks;

(iii) a non-deterministic task distribution increases the overall system efficiency;

(iv) small changes in task distribution cause great changes in the resulting system efficiency;

(v) robustness to speculations (for different commission rates).

A future development of the research would be to apply the generic model proposed in this paper to a real-world task allocation problem.

Acknowledgment

This work was supported by CNCSIS-UEFISCSU, project no. PNII-IDEI 316/2008, *Behavioural Patterns Library for Intelligent Agents Used in Engineering and Management*.

References

[1] C. Lucas, "Self-organizing systems: frequently asked questions," Version 3, September 2008, http://www.calresco.org/sos/sosfaq.htm.

[2] J. M. E. Gabbai, H. Yin, W. A. Wright, and N. M. Allinson, "Self-organization, emergence and multi-agent systems," in *Proceedings of the IEEE International Conference on Neural Networks and Brain Proceedings (ICNN&B '05)*, pp. 1858–1863, Beijing, China, October 2005.

[3] F. Heylighen, "Knowledge Management, Organizational Intelligence and Learning, and Complexity," in *The Encyclopedia of Life Support Systems*, L. D. Kiel, Ed., Eolss Publishers, Oxford, 2001.

[4] P. Glansdorff and I. Prigogine, *Thermodynamic Study of Structure, Stability and Fluctuations*, Wiley, 1971.

[5] N. S. Contractor and D. R. Seibold, "Theoretical frameworks for the study of structuring processes in group decision support system—adaptive structuration theory and self-organising systems theory," *Human Communication Research*, vol. 19, no. 4, pp. 528–563, 1993.

[6] C. Bernon, V. Chevrier, V. Hilaire, and P. Marrow, "Applications of self-organising multi-agent systems: an initial framework for comparison," *Informatica*, vol. 30, no. 1, pp. 73–82, 2006.

[7] F. Wang, "Self-organising communities formed by middle agents," in *Proceedings of the 1st International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS '02)*, pp. 1333–1339, Bologna, Italy, July 2002.

[8] G. Picard, C. Bernon, and M. P. Gleizes, "ETTO: emergent timetabling by cooperative self-organisation," in *Proceedings of the 3rd International Workshop on Engineering Self-Organising Applications (ESOA '05)*, pp. 31–45, Utrecht, The Netherlands, 2005.

[9] J. P. Georgé, M. P. Gleizes, P. Glize, and C. Régis, "Real-time simulation for flood forecast: an adaptive multi-agent system STAFF," in *Proceedings of the Symposium on Adaptive Agents and Multi-Agent Systems (AISB '03)*, pp. 7–11, University of Wales, Aberystwyth, UK, 2003.

[10] A. Dury, F. Le Ber, and V. Chevrier, "A reactive approach for solving constraint satisfaction problems: assigning land use to farming territories," in *Intelligent Agents V Agents Theories, Architectures and Languages*, J. P. Muller, M. P. Singh, and A. S. Rao, Eds., vol. 1555 of *Lecture Notes in Artificial Intelligence*, pp. 397–412, Springer, 1998.

[11] S. Rodriguez, V. Hilaire, and A. Koukam, “Holonic modeling of environments for situated multi-agent systems,” in *Proceedings of the 2nd International Workshop on Environments for Multi-Agent Systems (E4MAS ’05)*, Selected Revised and Invited Papers, pp. 18–31, Utrecht, The Netherlands, 2006.

[12] M. Gardner, *On Cellular Automata, Self-Reproduction, and the Game “Life”*, Scientific American, 1971.

[13] P. Rendell, *Turing Universality of the Game of Life*, Collision-Based Computing, Springer, London, UK, 2002.

[14] S. Stepney, F. A. C. Polack, and H. R. Turner, “Engineering emergence,” in *Proceedings of the 11th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS ’06)*, pp. 89–97, Stanford, Calif, USA, August 2006.

[15] E. M. A. Ronald, M. Sipper, and M. S. Capcarrère, “Testing for emergence in artificial life,” in *Advances in Artificial Life: 5th European Conference*, D. Floreano, J. D. Nicoud, and F. Mondada, Eds., vol. 1674 of *Lecture Notes in Artificial Intelligence*, pp. 13–20, Springer, Heidelberg, Germany, 1999.

[16] J. P. Crutchfield, “The calculi of emergence: computation, dynamics, and induction,” *Physica D*, vol. 75, no. 1–3, pp. 11–54, 1994.

[17] J. Deguet, L. Magnin, and Y. Demazeau, “Emergence and software development based on a survey of emergence definitions,” in *Emergent Intelligence of Networked Agents*, A. Namatame, S. Kurihara, and H. Nakashima, Eds., vol. 56 of *Studies in Computational Intelligence*, pp. 13–21, Springer, 2007.

[18] L. Guo and X. Cai, “The fractal dimensions of complex networks,” *Chinese Physics Letters*, vol. 26, no. 8, Article ID 088901, 2009.

[19] E. G. Bakhoum and C. Toma, “Specific mathematical aspects of dynamics generated by coherence functions,” *Mathematical Problems in Engineering*, vol. 2011, Article ID 436198, 10 pages, 2011.

[20] E. G. Bakhoum and C. Toma, “Dynamical aspects of macroscopic and quantum transitions due to coherence function and time series events,” *Mathematical Problems in Engineering*, vol. 2010, Article ID 428903, 13 pages, 2010.

[21] F. Leon, S. Curteanu, C. Lisa, and N. Hurduc, “Machine learning methods used to predict the liquid-cristalline behavior of some copolyethers,” *Molecular Crystals & Liquid Crystals*, vol. 469, pp. 1–22, 2007.

[22] D. J. Watts and S. H. Strogatz, “Collective dynamics of “small-world” networks,” *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[23] P. Erdős and A. Rényi, “On random graphs: I,” *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.

[24] S. Milgram, “The small world problem,” *Psychology Today*, vol. 2, pp. 60–67, 1967.

[25] R. Albert, H. Jeong, and A. L. Barabási, “Diameter of the world-wide web,” *Nature*, vol. 401, no. 6749, pp. 130–131, 1999.

[26] R. Albert and A. L. Barabási, “Statistical mechanics of complex networks,” *Reviews of Modern Physics*, vol. 99, no. 3, pp. 7314–7316, 2002.

[27] M. E. Gaston and M. desJardins, “Social networks and multi-agent organizational performance,” in *Proceedings of the 18th International Florida Artificial Intelligence Research Society Conference (FLAIRS ’05)*, Special Track on AI for Social Networks, Social Networks for AI, Clearwater, Fla, USA, May 2005.

[28] C. Song, S. Havlin, and H. A. Makse, “Self-similarity of complex networks,” *Nature*, vol. 433, no. 7024, pp. 392–395, 2005.

[29] F. Leon, *Intelligent Agents with Cognitive Capabilities*, Tehnopress, Iasi, Romania, 2006.

[30] G. Di Marzo Serugendo, M. P. Gleizes, and A. Karageorgos, “Self-organisation and emergence in MAS: an overview,” *Informatica*, vol. 30, no. 1, pp. 45–54, 2006.

[31] X. Zheng and S. Koenig, “Reaction functions for task allocation to cooperative agents,” in *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, pp. 559–566, Estoril, Portugal, 2008.

[32] K. Lerman and O. Shehory, “Coalition formation for large-scale electronic markets,” in *Proceedings of the 4th International Conference on Multi-Agent Systems*, pp. 167–174, Boston, Mass, USA, 2000.

[33] M. D. Weerdt, Y. Zhang, and T. Klos, “Distributed task allocation in social networks,” in *Proceedings of the 6th Automous Agents and Multiagent Systems*, pp. 500–507, Honolulu, Hawaii, USA, 2007.

[34] D. Ye, Q. Bai, M. Zhang, K. T. Win, and Z. Shen, “An efficient task allocation protocol for P2P multi-agent systems,” in *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA ’09)*, pp. 11–18, Chengdu, China, August 2009.

[35] X. Zheng and S. Koenig, “Greedy approaches for solving task-allocation problems with coalitions,” in *Proceedings of the Workshop on Formal Models and Methods for Multi-Robot Systems (AAMAS ’08)*, pp. 35–40, Estoril, Portugal, 2008.

[36] A. Campbell and A. S. Wu, “Multi-agent role allocation: issues, approaches, and multiple perspectives,” *Autonomous Agents and Multi-Agent Systems*, vol. 22, no. 2, pp. 317–355, 2010.

[37] R. G. Smith, “The contract net protocol: high-level communication and control in a distributed problem solver,” *IEEE Transactions on Computers*, vol. 29, no. 12, pp. 1104–1113, 1980.