Computing and Informatics, Vol. 29, 2010, 947–973

MAINTAINING FUNCTIONAL INTEGRITY IN MULTI-AGENT SYSTEMS FOR RESOURCE ALLOCATION

Krzysztof Cetnarowicz, Rafał Dreżewski

Department of Computer Science
AGH University of Science and Technology, Kraków, Poland
e-mail: {cetnar, drezew}@agh.edu.pl

Manuscript received 10 March 2008; revised 4 December 2009
Communicated by Ivana Budinská

Abstract. The resource allocation problem is the problem of assigning resources needed to accomplish some tasks. Such problems are present in many practical domains and are generally very hard to solve. In the paper two multi-agent systems for resource allocation are presented. Each of them uses different mechanisms (based on the ideas of *"free agents"* and *"life energy"*) for maintaining functional integrity related with the number of agents in the population. The preliminary results of simulation experiments are also presented and the mechanisms of maintaining functional integrity are analyzed.

Keywords: Multi-agent systems, resource allocation, functional integrity, limiting the number of agents mechanisms

Mathematics Subject Classification 2000: 68T42: Agent technology

1 INTRODUCTION

The resource allocation problem is the problem of assigning resources needed to accomplish some tasks. Such problems are present in many practical domains such as logistics, networking, manufacturing, parallel computations and grid computations. Generally such problems are very hard to solve and uncertainties in the times when resources are needed and released introduce additional complexities.

In the case of grid computations paradigm application is composed of sub-tasks. Each sub-task demands different types of services and resources. Resources in such computations are processors, memory, disk storage, service providers, and network links. The application of fast and robust technique of resource allocation, which can also deal with crisis situations, is crucial to the correct functioning of grid technology.

Multi-agent paradigm, which comes from artificial intelligence, is now regarded as the new approach to the construction of complex, distributed and decentralized systems [3, 6, 7, 4]. The application of multi-agent technologies leads to the construction of robust, flexible and crisis situations tolerant systems. It seems that systems based on multi-agent paradigm can be used for solving resource allocation problems, especially in the situations where we have to deal with distributed systems, dynamic environments, and uncertainties in the times when resources are needed and released (like in the grid computations).

Many problems related to functioning of multi-agent systems are still unsolved. Functional integrity of the systems belongs to such problems. In general, functional integrity of a multi-agent system may be defined as preservation of basic functions of the system during its functioning. Functional integrity may be analyzed from the point of view of different functions of the system (the functions that should be preserved) and also from the point of view of various factors that may influence the loss or preservation of functional integrity of the system.

In the paper two systems for resource allocation are presented. Each of them uses different mechanisms for maintaining functional integrity related with the number of agents in the population. These mechanisms are based on the ideas of so-called *"free agents"* and *"life energy"*. The *free agents* play a role of "unemployed" agents looking for a job. The "life energy" is the special type of resource used to prevent the excessive growth of the population of agents. The preliminary results of simulation experiments are also presented and the mechanisms of maintaining functional integrity are analyzed.

2 MULTI-AGENT SYSTEMS INTEGRITY: THE PROBLEM OF CONTROLLING THE NUMBER OF AGENTS

Functional integrity of a multi-agent system may be defined as preservation of basic functions of the system during its functioning. In a system, there are various factors that may influence the loss or preservation of functional integrity of a system.

The basic factors that may influence multi-agent system's functional integrity are as follows:

- The global number of agents in a multiagent system. Excess of agents may cause that system resources (hardware and software) become exhausted, what is followed by the breakdown of the whole multiagent system.
- The number of agents of a given type in a multiagent system. There are agents of different types, realizing different tasks. Ability to perform tasks of a system

is determined by an appropriate number of agents of each type: their absolute number as well as a relative one (relatively to numbers of agents in other groups).

- The proper utilization of a given agent or group of agents. The agent may be charged with a task for which it is not created. So, it is impossible to achieve the goal by the agent and then by the whole system.

As mentioned above, one of the basic factors that may influence multi-agent system functional integrity is the number of agents in a system.

Thus it is possible to single out such parameters as:

- global number of all agents:
  - excess of the global number causes a general blocking of the whole multi-agent system;
  - a global lack of agents causes gradual disappearance of system functioning;
- the number of agents of certain types;
  - an excess of agents of a certain type causes, that other groups of agents have limited access to system resources and perform their tasks worse;
  - a lack of agents of a certain type causes ceasing of system's ability to perform certain actions.

Thus a procedure of generation and liquidation of certain agents is important and may be considered as follows:

- A certain agent may (if it has such abilities) generate a new agent of the same type or of the other, determined type.
- A certain agent may be liquidated:
  - on its own initiative as a result of a decision made or as a result of its status,
  - as a result of initiative of the other agent (it is liquidated by the other agent).

Therefore a role of a mechanism that controls, depending on system needs, generation of new agents and liquidation of useless or even disturbing ones, is fundamental. It is very important that the mechanism mentioned above does not introduce the explicit or implicit centralization of the multi-agent system.

An agent has limited possibilities of perception of an environment, limited in principle to its own environment and it does not have a direct access to some global information in the system. As a result, in many cases, a certain agent making decisions on generation or liquidation of the other agent cannot take all factors essential from the point of view of functional integrity of the whole system into account.

The task of control of a number of agents is a key task for efficient system functioning. On the other hand, it is difficult to be performed if we additionally assume that the regulation mechanism should function automatically without a necessity to

centralize the multi-agent system that is decentralized from its nature and has to be decentralized.

There are some works where the out of control increase of the number of agents has been observed (for example [5]), but we did not find any discussion of methods that enable to control the number of agents in the system.

3 MULTI-AGENT SYSTEMS FOR RESOURCE ALLOCATION WITH UNEMPLOYMENT MECHANISM

Practical examination of MAS functional integrity problem connected with the number of agents in the system and verification of mechanisms of their number regulation were carried out with the use of a multi-agent system of resources distribution in decentralized environment of a graph form.

The graph is made by multiprocessor structure composed of processors as nodes and connections between them as edges. Tasks to be computed by processors make a resource to be balanced in the considered system.

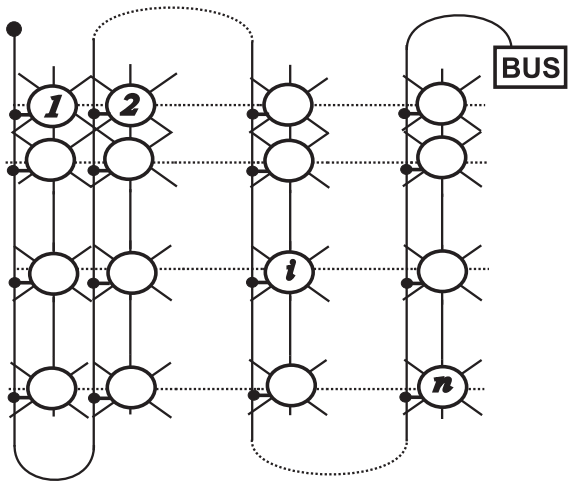


Fig. 1. Scheme of the multiprocessor structure

In the studied multiprocessor structure (Figures 1, 2) each of processors has its own memory and it can communicate with eight determined other processors by means of independent channels (ports) as well as with any other processor of the structure by means of a bus.

At a given moment of time only one pair of processors can use a bus. Every processor of the structure is equipped with the operating system that provides execution of tasks and task transfer by channels and by the bus. The goal of the multiprocessor system is to compute a group of tasks. The order of computing of

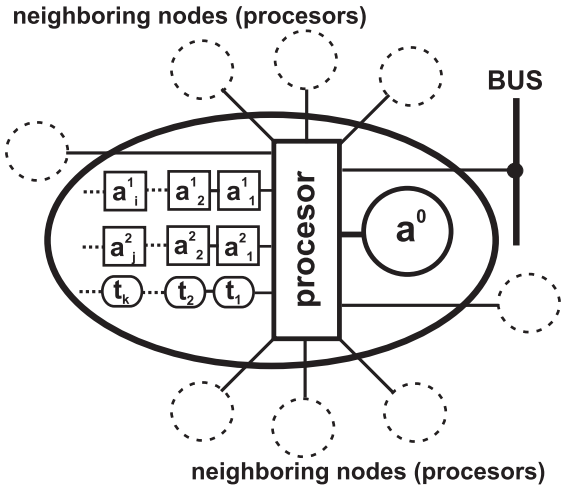


Fig. 2. Scheme of the node (processor) in the multiprocessor structure

tasks is determined at random during the course of computation (that is a typical situation, for example, for simulation of discrete processes).

The main function of the operating system is to ensure such a load balancing for processors of the structure that execution of all tasks could be completed in the shortest time possible.

Distribution of tasks in the structure by connections between a given processor and its neighbors can be provided by an operating system of the processor with the use of channels. Distribution of tasks without the use of a bus does not guarantee high execution efficiency. When a bus is applied, for each transmission two distant processors in the structure have to be chosen, and a sender as well as a receiver have to be determined.

For the purpose the above-mentioned multi-agent system may be applied.

The resource distributed around the multi-processor structure represents the tasks to be executed. The resource consumption corresponds to execution of tasks and resource generation – to creation of tasks.

In the realized multi-agent system agent a^0 (of type 0) that resides at a given processor which, for instance, needs tasks (or has excess of tasks) looks for tasks to exchange with neighboring nodes via direct connections.

However, if it is impossible (the neighbor nodes are in the same situation, i.e. they have a lack or an excess of tasks), then agent a^0 can create an agent a^1 of type 1 (or an agent a^2 of type 2). The created agent has a mission to find some tasks or a processor that has a lack of tasks. The agent a^1 (a^2) navigates through the whole multiprocessor structure and looks for a processor which has an excess (or a lack) of tasks. When it finds such a processor, it initiates a transmission process with the use of the bus.

An agent acquires the information necessary to realize its algorithm by the observation (what is typical for the M-agent architecture [1]). Mobile agent is able to observe resources (parameters) found in the node where it remains. The result of the observation gives input data to the performed algorithm of the agent. Mobile agents do not need to go back to their source node, but in certain type of experiments they in fact returned to the home node. Whether an agent has to return to the home node depends on the task being realized – sometimes it can be desirable to return to the home node and pick up the next task for realization and sometimes the agent can destroy itself after sending the obtained information to the home node. At its destruction the mobile agent can send information (about the destruction) to the agent of type a^0 of its source node to enable it to generate a new mobile agents.

Effectiveness of computing of the multi-agent system may be represented by:

- efficiency index E_f:
$$E_f = \frac{T_c}{n * T_r} * 100\,\% \tag{1}$$
where T_c is a time of execution of all the tasks in a single processor computer, T_r is a real time of execution of all the tasks in a multiprocessor structure, n is a number of processors in the multiprocessor structure,
- index of transient uniformity measure of tasks distribution in the structure W_q:
$$W_q = \frac{n * \max_{1 \le i \le n}(Nt_i)}{\sum_{i=1}^{i=n}(Nt_i)} \tag{2}$$
where Nt_i is a number of tasks in a node (processor) t_i, n is a number of processors in the structure.

3.1 Control of a Number of Agents in a Multi-Agent System

As mentioned before, an existence and a number of agents of a certain types belong to factors that have a basic influence upon preservation of functional integrity of the system.

One of very essential cases of functional integrity of the system is prevention of excessive (even unlimited) increase of the number of some groups of agents, and thus – the global number of agents.

The described problem may be observed on examination of the presented system of distribution of tasks. For instance, a decrease of the global number of tasks in the system leads to the increase of the number of agents looking for jobs (a^1).

This makes realization of basic functions of the system by other groups of agents more difficult, which causes the loss of functional integrity of the system (Figure 3).

At the time when a number of new tasks appearing in the system stops to increase (stabilizes or decreases), an increase (practically unlimited) of the total number of agents occurs.

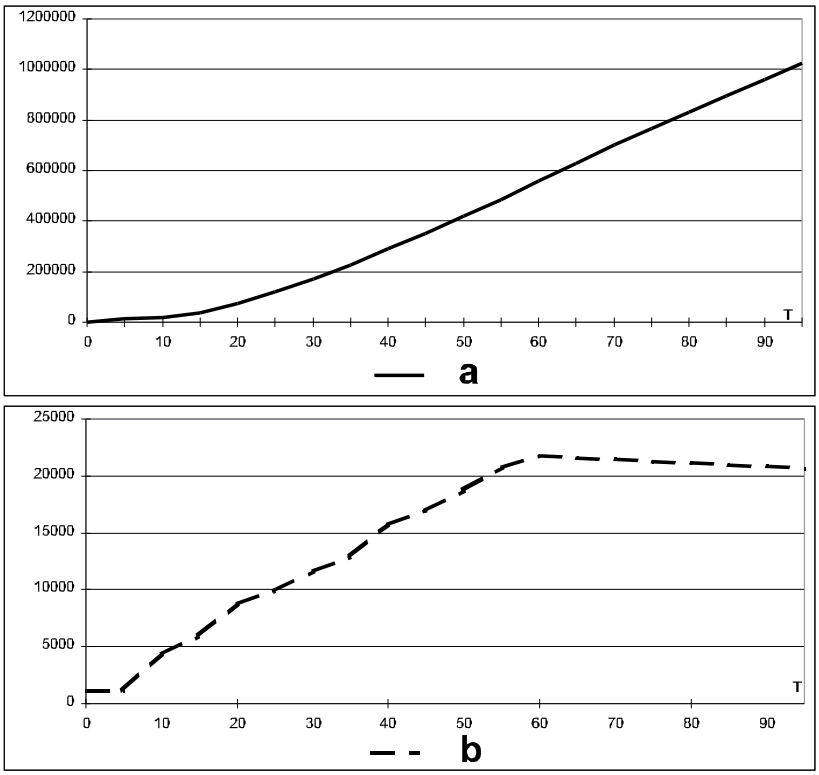


Fig. 3. Total number of agents a) and total number of tasks b) appearing in the multiprocessor system (T – time)

When the number of agents exceeds a permissible number for a certain system (in the examined system – approx. one million), blocking of the system functioning occurs.

3.2 Multi-Agent System with a Limited Number of Agents

To prevent unlimited increase of the number of agents in the system, it is possible to equip the system with mechanisms that enable to limit the number of agents existing and acting in the system at the moment.

In an example system, the constraint should concern the number of agents of type a^1 and a^2, while the number of agents of type a^0 is constant, in agreement with the shown definition (one agent in one node of the structure). Agents a^1 and a^2 are generated by agent a^0, therefore the limiting mechanism should be built in the algorithm of an agent of type a^0. It is realized with the use of the so-called *limitation mechanism*.

Each agent of type a^0 has maximal numbers of agents of type a^1 and a^2 assigned which may be generated at the given moment (in the experiments the number of generated agents is limited to 6 agents). It has counters that enable to trace a number of generated agents of type a^1 and a^2 – the agents that may be generated by a certain agent of type a^0.

Maximal permissible numbers of generated agents are defined arbitrarily at the moment of generation of each agent of type a^0. If a given agent of type a^1 or a^2 performs an assigned task and is liquidated, it sends information about it to an agent of type a^0, that generated it. The agent a^0 may now (if it is necessary) generate a new agent of a certain type.

As a result, each agent of type a^0, and thus each processor with which a certain agent is connected, may have at every moment of system functioning, the number of generated agents of type a^1 or a^2 from zero to the value defined by an appropriate parameter.

The parameters are fixed arbitrarily for each agent of type a^0 as system configuration parameters, selected in experimental way.

Simulation examination of a multi-agent system equipped with the described limitation mechanism consisting in arbitrary limiting of agents' number, confirm effectiveness of the proposed solution.

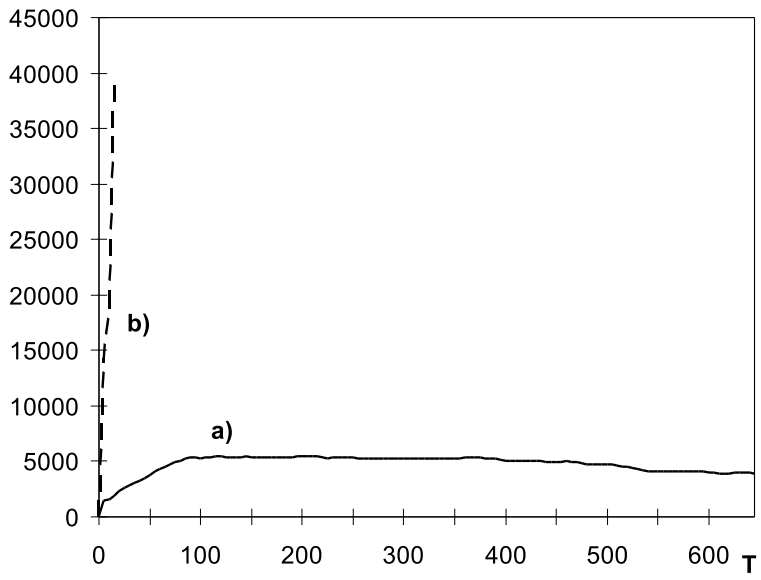


Fig. 4. Chart showing the total number of agents in time (T), variant with a limited number of agents a), variant without a limit of agents' number b)

The total number of agents shown in Figure 4 stays at a relatively low level and does not block the system work.

The presented mechanism of limitation of the number of agents is effective, however it has imperfections that have influence upon optimal work of a system. One of basic disadvantages is the fact that an arbitrary definition of limits of number of agents of particular types is difficult (or even impossible), both in a theoretical as well as in a practical way. It is even difficult to define a reasonable premise that enables in general case an estimation of optimal number of agents of particular types.

3.3 Limitation of the Number of Agents with the Use of Mechanism of Excessive Agents Liquidation

To regulate the number of agents, a mechanism of excessive agents liquidation may be used. The mechanism is based on accurate identification of results that may be caused by an excessive number of agents in the system. By way of analysis of the mentioned results, the decisions concerning the type and number of agents to be liquidated are made.

Analyzing the considered multi-agent system it is possible to notice that an agent of type a^0 connected with a certain processor is responsible for processor's work.

The agent, among others, performs tasks processing and services for agents being at the moment in a given node.

If the number of agents in a certain node (including agents passing through a given node) increases, then just an agent of type a^0 has to devote more time to operations for the sake of the mentioned agents instead of computing tasks.

Thus, if the number of agents (of type a^1 and a^2) in a certain node increases (above a given level), an agent of type a^0 residing in a given node liquidates some number of the agents (of type a^1 and a^2). As a result, a number of agents of type a^1 and a^2 is brought to some level as far as it is possible to the optimal one.

Results of application of excessive agents' liquidation mechanism are shown in Figure 5.

3.4 Limitation of the Number of Agents with the Use of Mechanism Making a Decision of New Agent Generation Dependent Upon the Number of Agents of the Same Type in Neighboring Environment

Analyzing a problem of too great a number of agents occurring in the system, it is possible to reach the source of the problem. To limit the number of agents of a given type it is necessary to check and control the number of generated agents. It is necessary to introduce a mechanism of control of agents' generation in a system.

An agent responsible for generation of all types of agents is of type a^0. Such a mechanism of control of generated number of agents should be built into an algorithm of an agent of this type.

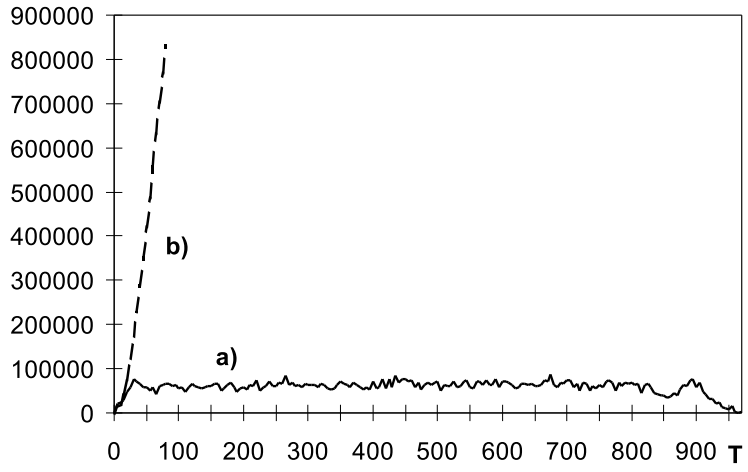


Fig. 5. Chart showing the number of agents in time (T), variant with a mechanism of liquidation of excessive agents a), variant without limitation of agents' number b)

Let us assume that an agent of type a^0 is to generate a new agent (of type a^1) searching for tasks. A schema of procedure of the mentioned agent of type a^0 that takes into account control of generated agents number is as follows.

The mentioned agent of type a^0 observes the closest environment i.e. the node where it is, particularly agents of type a^1 currently existing in the node during their passing through the node.

Each of the passing agents (of type a^1) has some resource of energy that is determined on generation and is being decreased (of fixed energy portion) during the transfer of the agent between nodes.

Many agents of low energy level in a given node make evidence that searching for tasks in a structure by other nodes (precisely, their agents of type a^0), is fruitless. This allows assuming with a high probability that there is a lack of searching resources (free tasks) in the environment (multiprocessor structure).

The mentioned resident agent of type a^0 counts the number of agents of type a^1, which have energy level less than some arbitrary determined level.

If a number of agents of type a^1 with the energy level less than a given level is greater than some arbitrary assigned number, then a given agent of type a^0 abstains from generation of a new agent of type a^1 (i.e. looking for tasks) on the assumption that such generation is pointless as there is a lack of a resource in the environment (free tasks).

Result of practical examination of influence – of the described mechanism of refraining from generation of new agents – on a general number of agents in the system is shown in Figure 6.

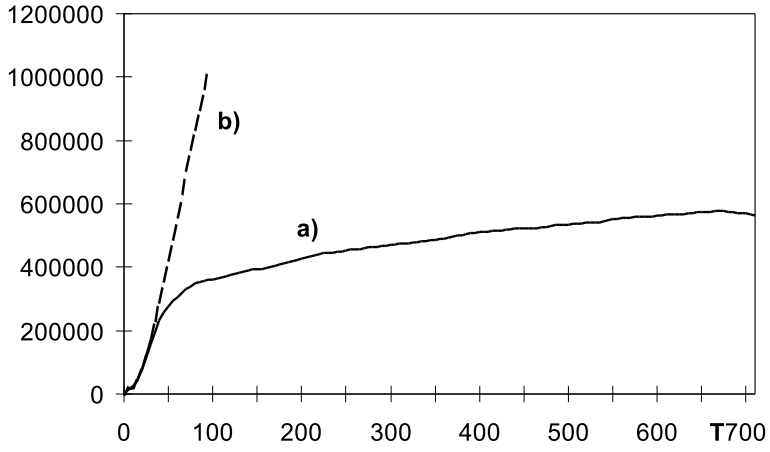


Fig. 6. Chart showing the number of agents in time (T). Variant with limitation of the number of generated agents (decision on refraining from generation) a) and variant without limitation of agents' generation b).

Practical usefulness of the described mechanism of regulation of a number of agents in multi-agent environments is hereby confirmed.

3.5 A Concept of Free Agents and Their Application to Stabilization of the Number of Agents in a Certain Population

The notion of a free agent enables effective and a relatively simple limiting of the number of agents in a certain population. Control of the total number of agents in a system is possible by way of increasing a number of agents of one type at the cost of the number of agents of the other type – according to the needs connected with tasks performed by the system.

Moreover, owing to migration of free agents in the system, effective transfer of agents from one part of environment to the other – where agents of a certain type are needed, is possible.

A concept of a free agent applies the principle that each agent may have the ability to generate the other agent of the same type or of the other type.

A free agent is such a sort of an agent that does not perform any assigned task but only moves around in the system and looks for tasks for itself. Thus it is a sort of "unemployed" agent searching for a job. In particular, a procedure of a free agent may be presented in the following points:

- A certain agent of a determined type undertakes actions to perform assigned tasks.

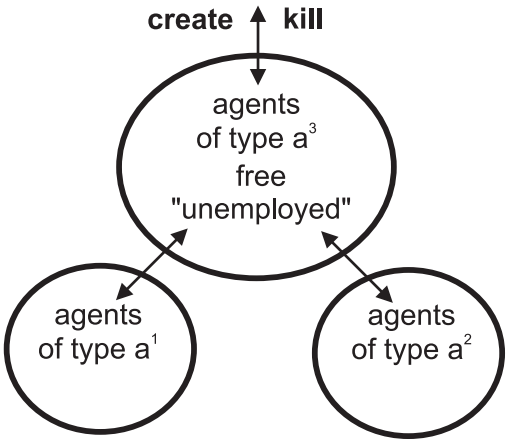


Fig. 7. Scheme of evolution of population of agents with the concept of a free agent

- A situation occurs when the mentioned agent cannot continue its activity and is to be liquidated. It may happen in these two cases:
  - an agent concludes that its further activity is pointless because a task was realized and it liquidates itself,
  - an agent is not able to realize the assigned task, thus further action of the agent is impossible and the agent undergoes liquidation – e.g. it ceases to exist because of lack of life energy.
- Before its liquidation, an agent generates a free agent – thus it transforms itself into an "unemployed" agent.
- A free agent, moving in the environment checks if, on the basis of a status of environment, there is a need for an agent of a type available in the system.
- If a free agent finds out that there is such a need for an agent of a determined type (a^1 or a^2), it generates such an agent and it undergoes liquidation (effectively it transforms into an agent realizing a task).

The concept of a free agent consists in creation of a new type of an agent a^3, that as an "unemployed" one occupies itself with a search for a job.

As a result, evolution of the population of agents consists in the following: agents of types a^1 and a^2 may transform into agents of a type a^3 (free, "unemployed") and vice versa. The transformation process is composed of two operations: agent generation and agent destruction. For example the agent of type a^3 generates an agent of type a^1 or type a^2 and destroys (deactivates) itself.

If we want to change the global number of agents in a system then we generate (create) or liquidate (kill) only free agents – of type a^3 (Figure 7).

3.6 Results of Simulation Examination of a Multi-Agent System with Free Agents Applied to Distribution Tasks

The presented concept of a free agent application was examined with the use of an exemplary multi-agent system for tasks distribution in the multi-processor structures.

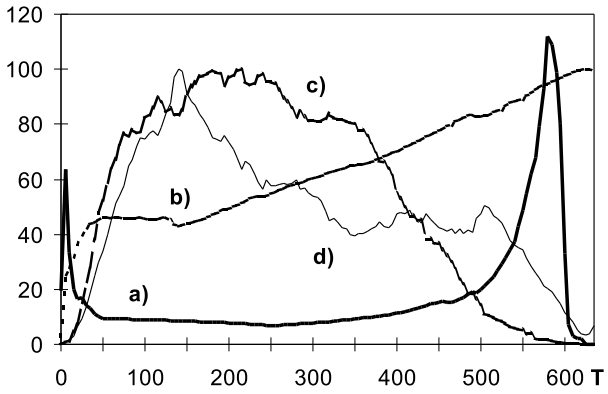


Fig. 8. Chart showing the number of agents in time (T). Variant with free agents ("unemployed") chart of a coefficient of non-uniformity of tasks' distribution a), relative number of agents: of type a^1 b), of type a^2 c), of type a^3 – free agents d).

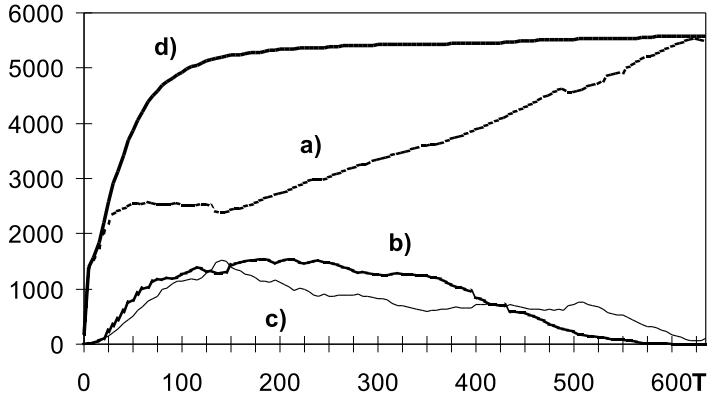


Fig. 9. Chart showing the number of agents in time (T). Variant with free agents ("unemployed") the number of agents of type a^1 a), the number of agents of type a^2 b), the number of agents of type a^3 – free agents c), chart of the total number of agents of all types (a^1, a^2, a^3) – d).

Results in the form of a chart are shown in Figures 8 and 9.

The chart in Figure 9 shows changes of the numbers of each type of agents during the system functioning and tasks computation. It is possible to observe that the global number of agents initially increases up to a predetermined value and stays practically constant during the computation.

Numbers of agents of certain types change depending on the needs.

The fact may also be observed in Figure 8, where the number of agents of each type was compared with the coefficient of non-uniformity of tasks distribution (W_q) during computations.

variant	Efficiency index E_f
mechanism of arbitrary limiting of a number of agents in node	72 %
mechanism of limiting of a number of generated agents (decision on generation)	73 %
mechanism with the use of a free agent (a constant global number of agents)	73 %

Table 1. Results of simulation examination of a multi-agent system of tasks distribution. Comparison of efficiency of the system for different mechanisms of the number of agents limiting

The results shown in Table 1 make evidence that efficiency of the system with the presented mechanisms (with arbitrary limited number of agents in the node, with a mechanism of abstaining from generation in the case of a large number of agents in surroundings and with a mechanism based on a concept of free agents) is close for each mechanism.

However (as mentioned before), the facility is different in application of each method, easiness of optimal parameters selection and universality of the methods.

4 MULTI-AGENT SYSTEM FOR RESOURCE ALLOCATION WITH THE MECHANISM OF LIMITING OF THE NUMBER OF AGENTS BASED ON THE "LIFE ENERGY"

The multi-agent system for resource allocation with the mechanism of limiting of the number of agents based on the "life energy"is presented in Figure 10. The main goal of the system is to maintain the amount of resource in all nodes at the optimal level.

The environment of the system has the graph-like structure and is composed of nodes with each node connected with its four neighbors. Each node can be located

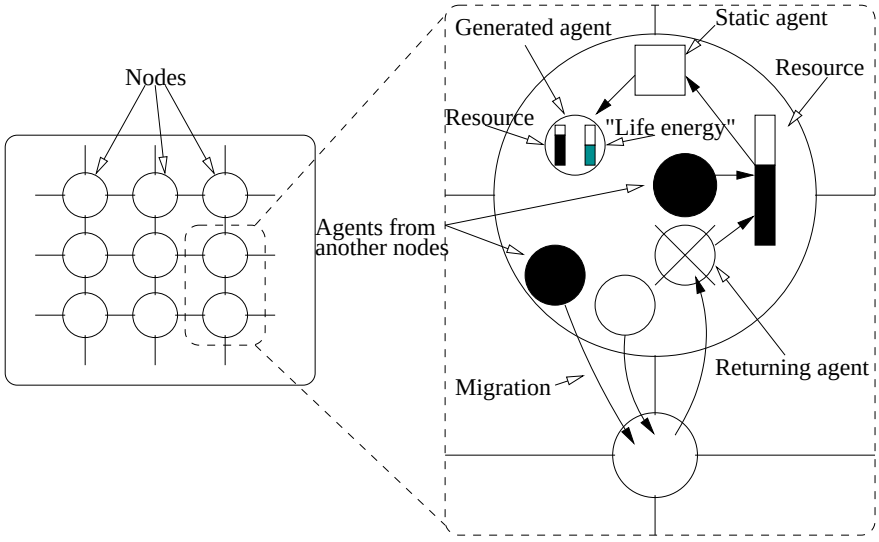


Fig. 10. Multi-agent system for resource allocation with the mechanism of limiting of the number of agents based on the "life energy"

on different host. In each node there is some amount of resource. The amount of resource in a particular node is changing during the simulation in stochastic manner.

There are two kinds of agents in the system: static and mobile. One static agent is present at each node. The main goal of the static agent is to make decisions whether it is necessary in the given time to generate a mobile agent. This decision is based on the current amount of the resource in the given node.

When the actual level of resource in the node is not equal to the optimal value then the static agent generates the mobile agent. When there is too much resource in the given node, the static agent gives some resource to the mobile agent.

In the opposite case, the mobile agent is sent in order to collect some resource from another nodes and "transport" it to the node of its origin. The mobile agent can gain some resource from another node only when the priority of its home node is higher than that of the node in which it is currently located. The priority of the given node is proportional to the deviation of the current amount of its resource from the optimal level.

In order to realize their goal, the mobile agents migrate from node to node. In the process of making the decision to which node the agent should migrate the nodes which were already visited are taken into account – the agent tries to avoid already visited nodes. When the agent realizes its goal (collecting or distributing the given amount of resource) it returns to its home node, returns the resource which it possesses, and is removed from the system.

The system is implemented in Java. All the agents are implemented as threads and the communication between nodes is based on the socket mechanism, so it is possible to run the nodes at the same host or at different hosts.

4.1 The Number of Agents Control Mechanism Based on "Life Energy"

In the systems described in the previous sections the agents' "energy" was used as the "second level" mechanism for maintaining system integrity. In the system described in this section the so called "life energy" mechanism is used as the main technique of preventing the excessive growth of the number of mobile agents.

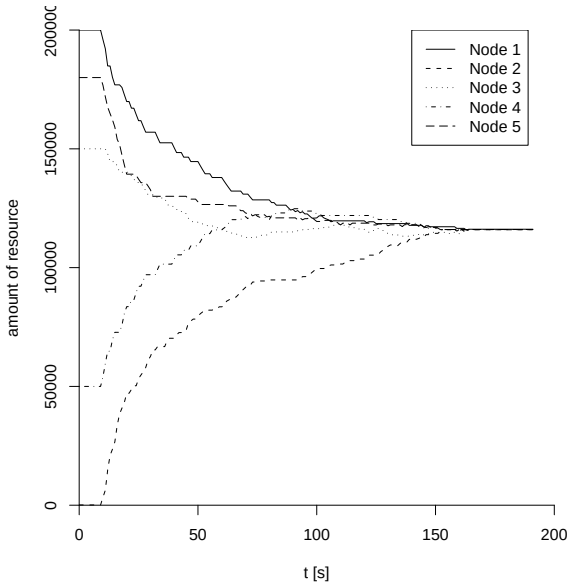
During its creation each mobile agent is given some amount of special resource (called "life energy"). Each migration between nodes costs some "energy". When the agent loses all its "energy" it returns to home node, despite of the state of realization of its goal. When the goal was not fully realized the agent returns all undistributed (or collected) resource to its home node after the return.

Such mechanism, in connection with the restriction of the number of mobile agents generated in each node, prevents excessive growth of the number of agents resulting from mobile agents, which can not realize their goals and do not return to their home nodes. Without such mechanism such agents do not return to its home nodes and nodes generate additional mobile agents, what can result in excessive growth of the population and in poor performance of the whole system.

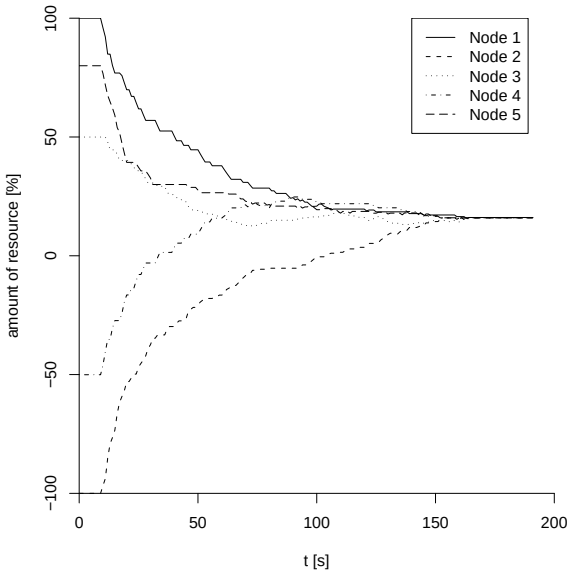
4.2 Experimental Results

In this section the results of experiments with the described system are presented. The main goal of the experiments was to investigate whether the presented system for resource allocation works properly during various crisis situations. Also the influence of the maximal number of mobile agents generated by each node on the performance of the system was investigated. Other parameters were constant during the experiments – they were set to optimal values obtained during preliminary experiments.

The results of the experiments, whose goal was to show whether the system works properly with different maximal numbers of mobile agents, are presented in Figures 11–15. The best results were obtained when the maximal 15 mobile agents for each node were used. In this case it took about 15 seconds to allocate resources in such a way that their levels was equal in all 5 nodes starting from different levels in each node. In the case of less agents used, the time needed to allocate the resources was longer and in the case of 20 agents the time was longer and the resources were not allocated quite properly – there were differences in the resource levels in the nodes resulting from the fact that many agents were generated and each of them took some resources in order to distribute them among the other nodes. In the case when the differences in the amounts of the resource between nodes are small and there is generally too much resource in the system, mobile agents can not find nodes where they may leave their resources.

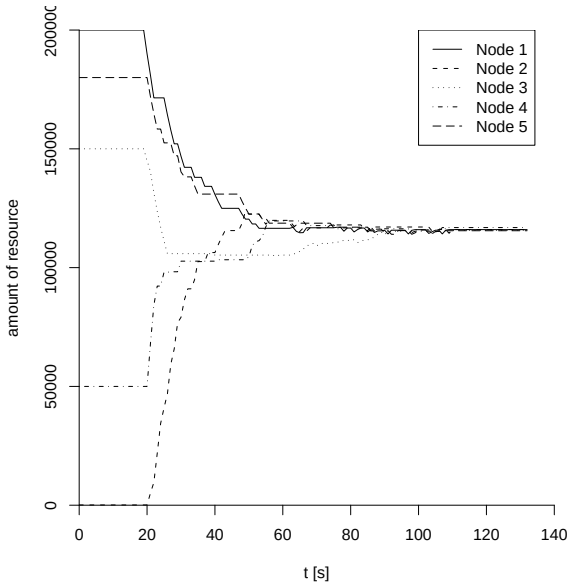


a)

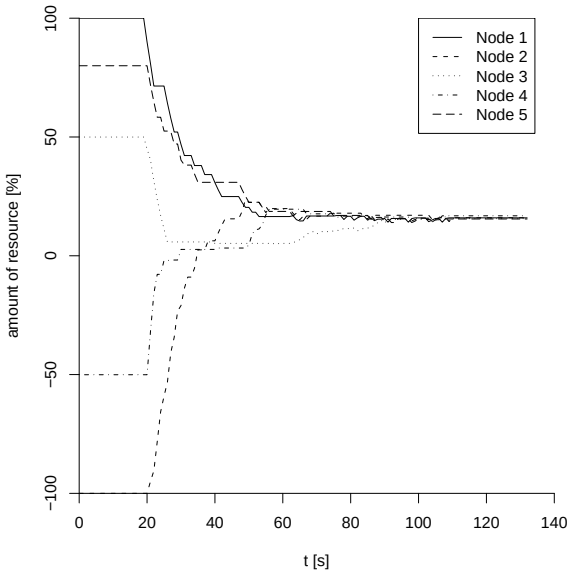


b)

Fig. 11. The amount of resource a) and the percentage of deviation from the optimal level of resource b) in the nodes during the typical experiment. Maximum 2 mobile agents generated in each node.

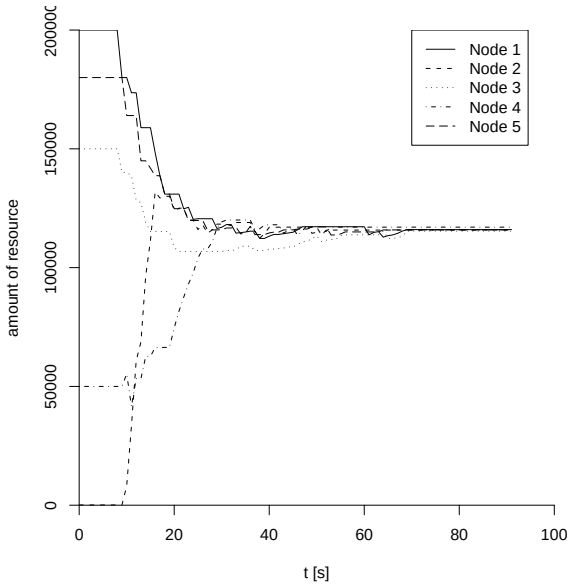


a)

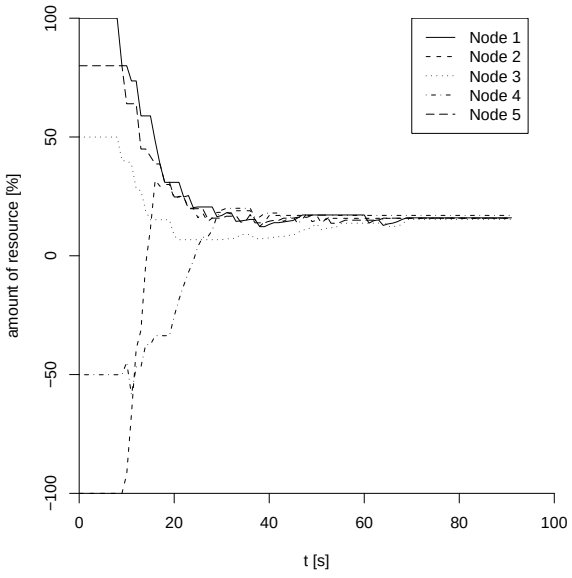


b)

Fig. 12. The amount of resource a) and the percentage of deviation from the optimal level of resource b) in the nodes during the typical experiment. Maximum 5 mobile agents generated in each node.

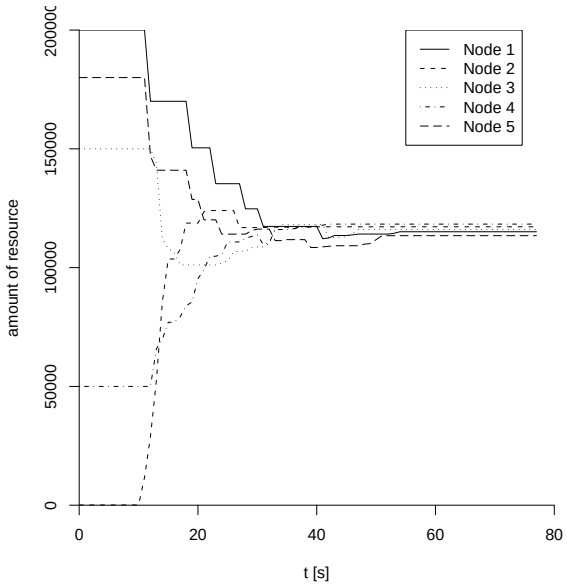


a)

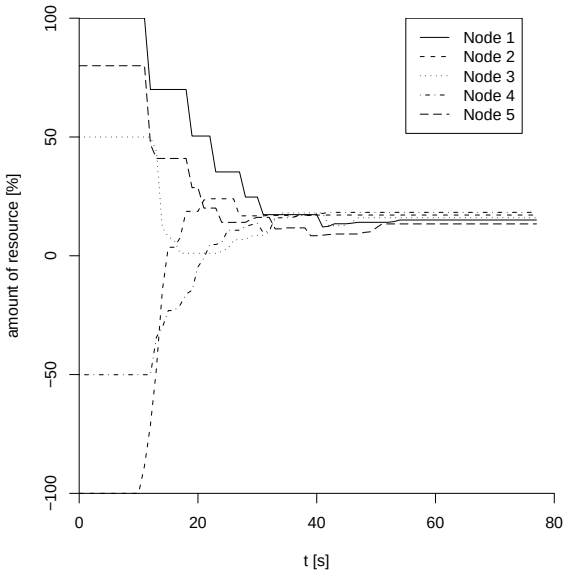


b)

Fig. 13. The amount of resource a) and the percentage of deviation from the optimal level of resource b) in the nodes during the typical experiment. Maximum 10 mobile agents generated in each node.

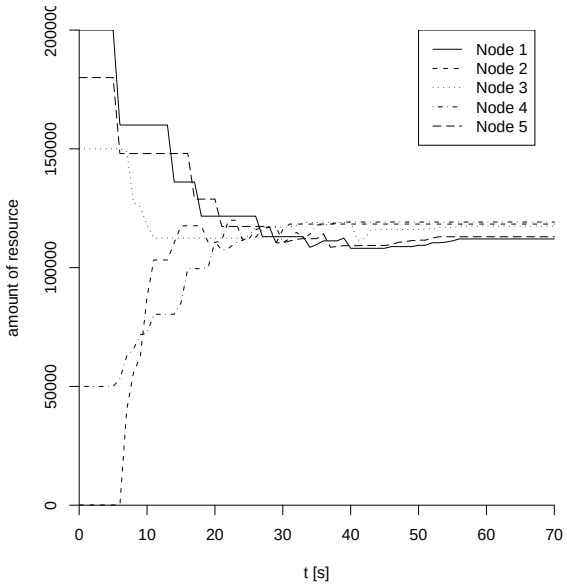


a)

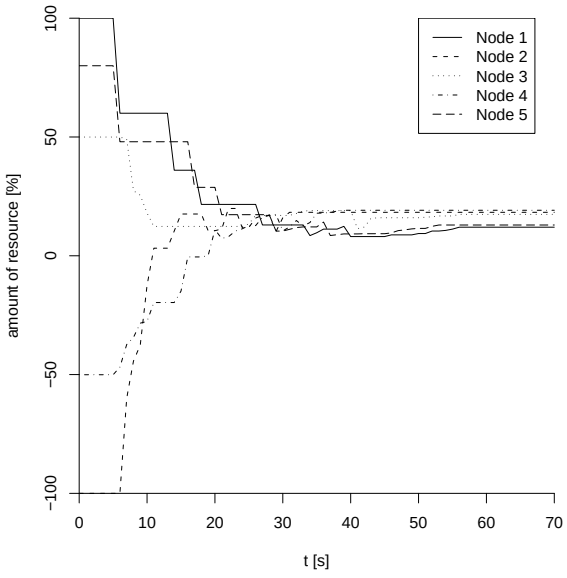


b)

Fig. 14. The amount of resource a) and the percentage of deviation from the optimal level of resource b) in the nodes during the typical experiment. Maximum 15 mobile agents generated in each node.

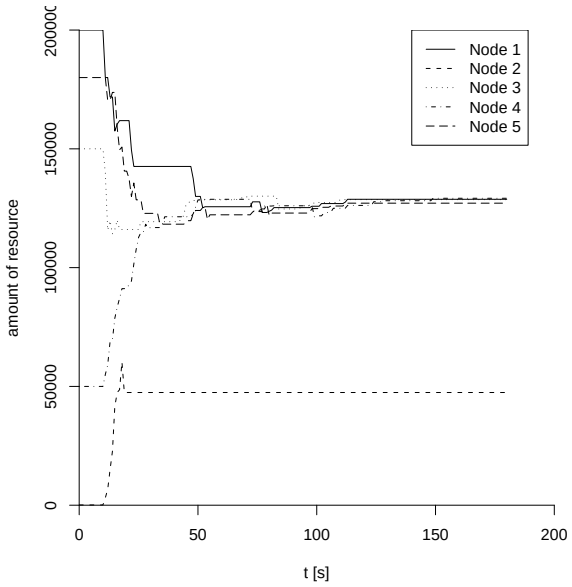


a)

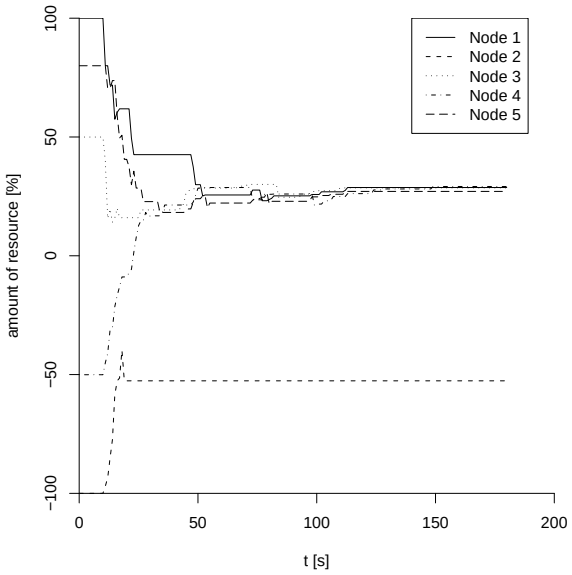


b)

Fig. 15. The amount of resource a) and the percentage of deviation from the optimal level of resource b) in the nodes during the typical experiment. Maximum 20 mobile agents generated in each node.

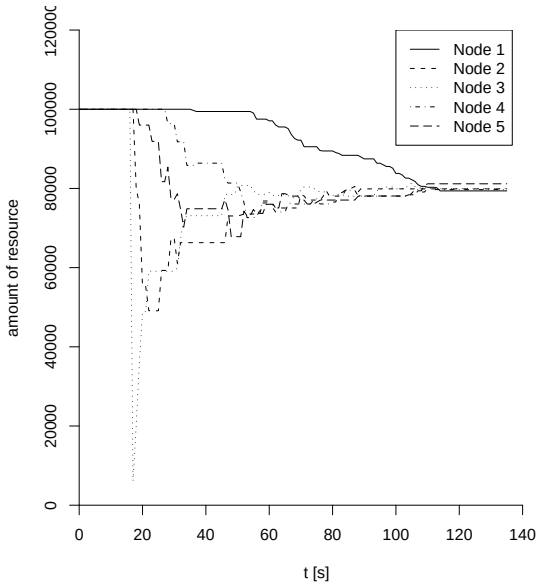


a)

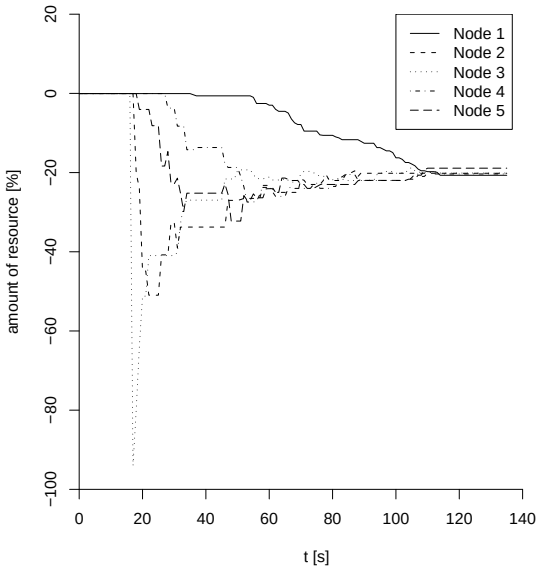


b)

Fig. 16. The amount of resource a) and the percentage of deviation from the optimal level of resource b) in the nodes during the experiment with one node down. Maximum 10 mobile agents generated in each node.

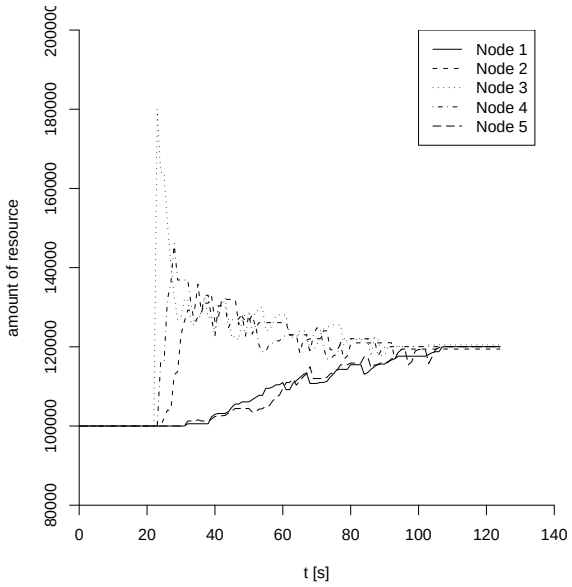


a)

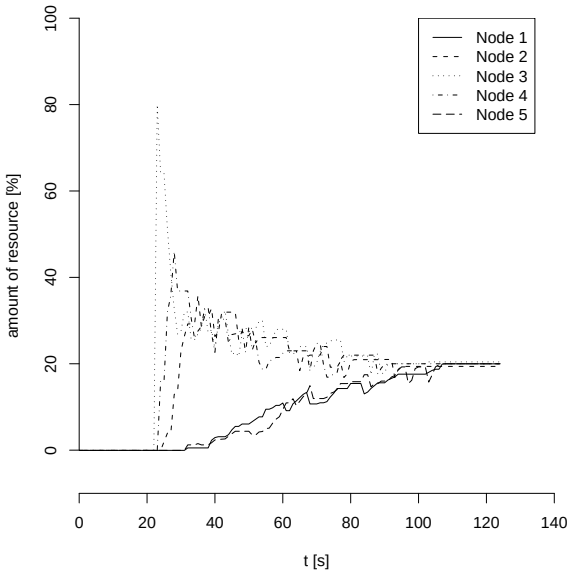


b)

Fig. 17. The amount of resource a) and the percentage of deviation from the optimal level of resource b) in the nodes during the experiment with the rapid decrease of the resource in one node. Maximum 10 mobile agents generated in each node.



a)



b)

Fig. 18. The amount of resource a) and the percentage of deviation from the optimal level of resource b) in the nodes during the experiment with the rapid increase of the resource in one node. Maximum 10 mobile agents generated in each node.

Results of the experiments with different kinds of crisis situations are presented in Figures 16–18. The results of experiments with one node down during the process of distributing the resource are presented in Figure 16. It can be seen that the system was able to redistribute resources despite of the malfunction of one of the nodes. In the case of the rapid decrease of the resource in one of the nodes the system was also able to properly redistribute resources, what can be observed in Figure 17. In Figure 18 the results of the experiment with the rapid increase of the resource in one of the nodes are presented. Also in this case the system was able to properly redistribute the resource between the nodes.

5 CONCLUSIONS

In the paper the application of the multi-agent paradigm to the construction of the systems for resource allocation was presented. Two such systems were presented in the course of this paper. The basic difference between them was the mechanism of maintaining their functional integrity – controlling the number of agents in the system. First of the presented systems used the mechanism of which the most important part was the idea of "unemployed agents" and the second one the mechanism based solely on the "life energy".

As the presented preliminary results clearly show, multi-agent systems can be used in the case of the resource allocation problems. Systems based on the multi-agent paradigm are robust, crisis situations resistant, and can be applied to the resource allocation problems in dynamic and distributed environments.

Both mechanisms of maintaining functional integrity of the presented systems worked well. The experiments showed that the problem of excessive number of generated agents cannot be solved directly by agents responsible for the generation, on their own. The reason is they usually do not have the global information, which would enable making appropriate decisions. However, a possibility exists that the agents may obtain the information indirectly by the observation of some groups of the agents of other types, within the whole population of agents. It is even possible to suggest a solution consisting in the following: in the system there are some types of agents, whose only task is global information supply to the other agents (also by their own specific behavior in the system).

Stabilization of the number of agents with the application of the "free agents" concept seems to be very promissing. The concept may be considered as a special approach to the the multi-agent system, where agents on their own look for the tasks, since they are not arbitrary granted assignments to the jobs. Moreover, a group of the free agents ensures a way of agents' transfer from a group of a certain type to a group of the other type what causes self-control of a number of agents within the groups of certain types depending on the system needs.

The idea of "life energy" – introducing special kind of resource needed for performing every action of an agent – also proved to work well in the task of maintaining functional integrity of the multi-agent system. Such mechanism was already

applied with success in evolutionary multi-agent systems, in which the evolutionary processes are realized in the multi-agent system [1, 2]. In such systems agents may reproduce and die, thus some decentralized mechanisms of controlling of the number of agents is needed. The mechanism of "life energy" also plays the role of selection mechanism in such systems – the agents compete for limited resources and the better fitted ones survive and reproduce with greater probability.

Future research will include the application of the multi-agent systems for resource allocation in dynamic environments and in domains with high uncertainties in the times when the resources are needed and released, like logistics, manufacturing, and grid computations. The comparison to other techniques for resource allocation is also included in the future plans.

REFERENCES

[1] Cetnarowicz, K.—Kisiel-Dorohinicki, M.—Nawarecki, E.: The Application of Evolution Process in Multi-Agent World to the Prediction System. In M. Tokoro (Ed.), Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS 1996), Menlo Park, CA, 1996. AAAI Press.

[2] Dreżewski, R.: Co-Evolutionary Multi-Agent System With Speciation and Resource Sharing Mechanisms. Computing and Informatics, Vol. 25, 2006, No. 4, pp. 305–331.

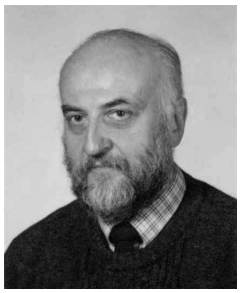
[3] Ferber, J: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley, 1999.

[4] Russell, S.—Norvig, P.: Artificial Intelligence: A Modern Approach (2nd Edition). Prentice-Hall, 2002.

[5] Uhurski, P.—Grochowski, M.—Schaefer, R.: A Two-Layer Agent-Based System For Large-Scale Distributed Computation. Computational Intelligence, Vol. 24, 2008, No. 3.

[6] Weiss, G. (Ed.): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press, 1999.

[7] Wooldridge, M.—Ciancarini, P.: Agent-Oriented Software Engineering: The State of the Art. In P. Ciancarini and M. Wooldridge (Eds.), Agent-Oriented Software Engineering, Vol. 1957 of LNCS. Springer-Verlag, 2001.



Krzysztof Cetnarowicz is Associate Professor at the Department of Computer Science, AGH University of Science and Technology in Kraków, Poland. He obtained Ph. D. degree in 1977 and habilitation in 1999 at the AGH University of Science and Technology. He is the author of patents and more than 100 papers in artificial intelligence, multi-agent systems, evolutionary algorithms, image processing, and simulation. He was the member of programme committees of the DIMAS 1995, ICMAS 1998, MAAMAW 2001, CEEMAS 2001, IACS 2004, CEEMAS 2005, the workshop Intelligent Agents in Computing Systems organized within the ICCS 2007, and the workshop Intelligent Agents and Evolvable Systems organized within the ICCS 2008 conference.

Rafał Dreżewski works at the Department of Computer Science, AGH University of Science and Technology in Kraków, Poland. He obtained Ph. D. degree in 2005. His research interests include artificial intelligence techniques and artificial life simulations of complex and emergent systems. He is the author of more than 40 papers mainly in the area of artificial intelligence, evolutionary algorithms, and multi-agent systems. He was the member of programme committee of the Workshop on Evolutionary Computation in Finance and Economics (EvoFIN) organized within EvoStar 2008, EvoStar 2009, and EvoStar 2010 conferences, the member of programme committee of PPSN 2010, and the member of the technical committee of Congress on Evolutionary Computation 2008.