Available online at www.sciencedirect.com

ScienceDirect

AASRI Procedia 6 (2014) 49 – 58

AASRI Procedia

www.elsevier.com/locate/procedia

2013 2nd AASRI Conference on Computational Intelligence and Bioinformatics

# Adaptive Multiagent Organization of the Distributed Computations

Dr. Anatoly Kalyaev[a]*, Dr. Iakov Korovin[a]

*aScientific Research Institute of Multiprocessor Computing Systems of Southern Federal University, Chekov st., 2, 347900, Taganrog, Russia*

**Abstract**

In this paper, authors offer the new method of the organization of the distributed computations. The main benefit of proposed method is possibility of using resources of many different private computers for solving coherent tasks. Proposed method allow correcting the computational process when computing nodes parameters change. Computing system achieves this ability because it consists of equal proactive agents, which interact to create communities to solve incoming tasks. While creating the community agents optimize it composition to solve user task in time and to minimize needed computing resources.


Peer-review under responsibility of Scientific Committee of American Applied Science Research Institute

*Keywords: distributed computing systems; collective decition-making; decentralization; communities; adaptability; multiagent systems*

## 1. Introduction

The idea of using corporative and personal computing resources for solving computing tasks appears more than 30 years ago, but only about 10 years ago due to progress in global and local networking different organizations begin to use such systems. This type of systems were called distributed computing systems

* Corresponding author. Tel.:+7-952-5666699 .
*E-mail address:* anatoly@kalyaev.net .


Peer-review under responsibility of Scientific Committee of American Applied Science Research Institute
doi:10.1016/j.aasri.2014.05.008

(DCS). The first widely spread distributed computing systems were GRIDs proposed by Ian Foster and Carl Kesselman [1] [2]. Today, the relevance of such systems grows, powered by a progress of both networks and personal computers.

While analyzing some widely spread DCS [3] [4] [5] [6] [7] we noticed that majority of them could only be used in private networks based on a set of similar computing nodes (CNs). The computations in such DCS networks are usually controlled by computing nodes that controls distribution of incoming tasks in computing nodes network [8].

Deeper analysis of DCS systems shows that dedicated servers perform almost all control functions in such systems. This type of organization of the DCS allows providing load balancing between CNs.

However, modern local and global networks interconnects many private and corporative computers, and great part of such computers is not fully loaded. While performing analysis authors made decision that, task of using these idle resources for creation of DCS is very important.

The main problem of using such CNs in DCS is probability of changing of their parameters (such as computing power, network bandwidth and so on) any moment if their owners perform some actions (for example run some application or begin downloading some file). Another problem is the variety of CNs of different owners. Within one DCS may coexist many different CNs, and this makes it much more difficult to solve the task of effective distributing of tasks.

The project SETI@home (setiathome.berkeley.edu) can be example of such attempt. This project uses the CNs of volunteers to perform filtering of the signals from radio telescopes to look for extraterrestrial intelligence. However, analysis show that SETI@home allow to use private computing resources because it is created to solve only easily decomposing tasks, that can be divided into non-coherent pieces. Therefore, non-coherent tasks simplifies implementation of load balancing and time to solve task is not critical, that means that characteristics of the individual CNs can be dynamically varying. That makes systems like SETI@home very limited in usage. That is why it is urgent to develop new methods and algorithms to allow using personal CNs in DCS.

Creation of universal DCS based on private CNs collaboration is considerably more difficult because it should allow solving coherent tasks consisting of coherent subtasks. The problem appears because the set of CNs in DCS is mixed: some CNs have a wider channel, some CNs have higher computational performance. In addition, there are dynamic changes of parameters of CNs depending on their owner's behavior. Today effective loading of a DCS while solving coherent tasks is a challenge. [9] The task of solving such tasks while parameters of CNs dynamically changes is much harder. However, the development of new methods for performing adaptive computations in DCS, which can let the system to adjust computations to variations of CNs parameters, can solve it [10].

## 2. Principles of Organization and Functioning of the Decentralized DCS

In the proposed method of DCS organization, we try to avoid dedicated servers and to make a decentralized organization of the DCS [11]. This new method is based on multiagent systems and collective decision-making principles. Every CN of DCS have a proactive agent software installed and interaction between these agents realize dispatching of incoming tasks. Every time DCS can consist of different set of agents that is why there appears problem of interaction between user and system because user do not know where to send his task. To solve this problem, we use service nodes that are passive and serve as "bulletin boards" (BB) [12] a structure chart of proposed decentralized DCS is shown on figure 1.

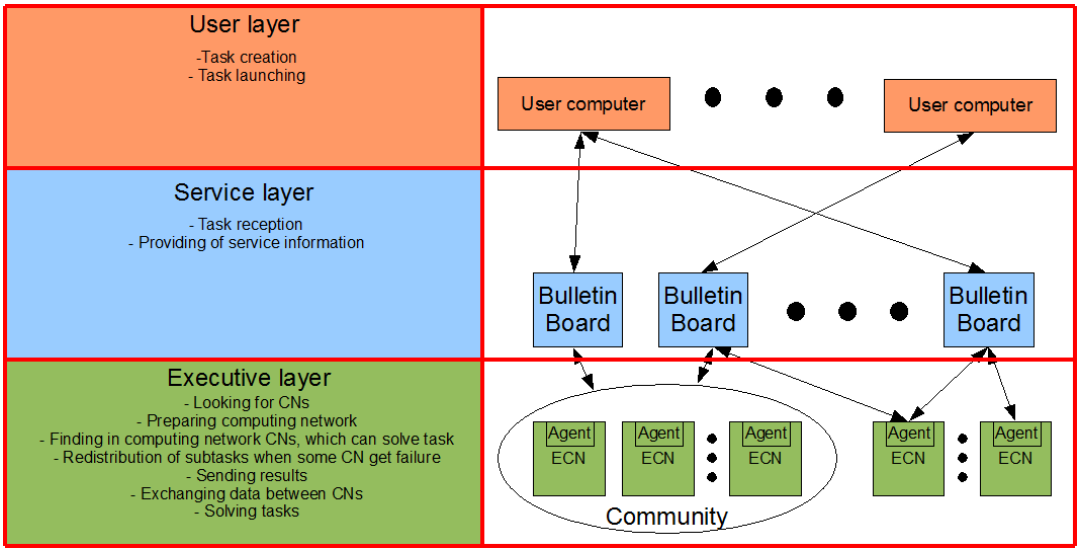


Fig. 1. a structure chart of proposed DCS

As you can see, proactive agents perform almost all service tasks in such DCS [13]. We proposed to implement each agent on basis of real executive CNs (ECN on the figure 1). Agents dispatch the process of solving user tasks in DCS. In order to solve coherent tasks they need to create unique communities to solve all such tasks [5]. The target of the community is to solve incoming user task during a time, specified by user. Set of the CNs in community have to vary if parameters of CNs in this community change. The system may experience failures of CNs, variation of CNs computing power, reducing of bandwidth, and all that can cause increasing of the task solution time. In case of all this situations community of agents have to change its composition in order to solve the task in time, specified by user.

In general, method of solving of incoming task in decentralized DCS can be described as follows. User of the system forms task he want to solve, sets requirement for the time to solve the task and determines a virtual payment for solving, which is represented by some virtual points (it can be money if we use private computers or just priority of task if we use corporative CNs). After that user sends all the data to a BB. Proactive agents of the DCS are monitoring BBs and look for new tasks. If some agent finds new tasks, it use task's computational complexity and the number of virtual points to evaluate profit of the performing of each task. When agent finds the most "beneficial" task, it joins the community to solve it: it takes some part of computations required to solve the task. Every time new agent joins community, it evaluated time of task solving, and at some point of time, the community becomes powerful enough to solve the task in time and the community stops its expansion. When new agent joins the community, it take some subtask and begin to solve it. While solving the subtasks agents are monitoring their CNs' parameters. If variations of parameters cause exceeding of user-specified time of task solving, the community begin accepting new agents. When community solves the task, the agent, that produce the result, send it to the user, and then make the community to dissolve itself [14].

To make it easier to understand proposed method we decided to divide it into 3 stages:

- The formal representation of the task;
- Collaboration in a community;
- Distribution of task in created community.

Let us describe stages in more details.

## 3. Formal Representation of Task

At first, we need to present the task in a parallelized form to let the community of agents perform the solution. In our work, we decided to represent task in the system in form of information graph (IG).

IG of the task is a graph where vertices represent subtasks of the user's task and arcs represent data transfers between subtasks. In order to form the IG of the task, user have to create the task in a form that allows logically decomposing it into interconnected parts.

In the IG G (Q, X) of the task every vertex $q_i \in Q$ represents i-th subtask. For very subtask user specifies the estimated computational complexity $Y_i$. Every arc $x(q_i, q_j) \in X$ is specifies to the amount of data $W_{ij}$ in transfer from subtasks $q_i$ to subtask $q_j$ [15].

For convenience, we decided to present the IG in the multi-layer parallel form (MPF) to simplify the further parallelization of the task graph. To do that all the vertices of the graph should be divided into subsets $V_i$ according to the rule:

If the arc $x(q_m, q_n)$ connects the layer j to the layer k, that means that $j < k$.

Figure 2 shows MPF for the task IG, which contains 9 vertices.

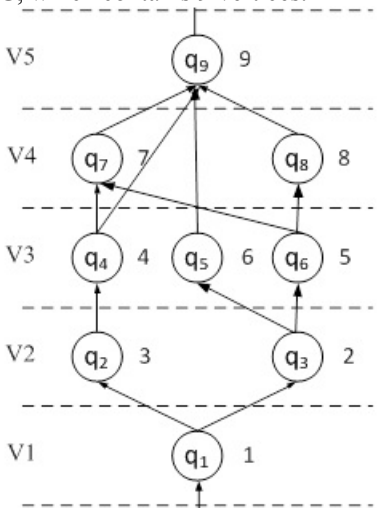


Fig. 2. the MPF of the IG of the task

When task IG is represented in the MPF, we can easily evaluate the number of parallel computations of the task and the maximum number of agents in community for solving this task as the maximum quantity of vertices in layer of the MPF IG.

At the next step the user have to specify maximum time to solving his task and priority (quantity of virtual points).

After all user sends the task to one of the BBs of DCS.

## 4. Collaboration in a Community

When the user task is sent to the bulletin board of the DCS, nodes of the DCS start to collaborate in a community which will allow to solve the task in user specified time. [16]

While creating community of agents decentralized DCS use the following algorithm:

1. Bulletin board marks the incoming task as incomplete one;
2. Agent Aj look for tasks that are incomplete at bulletin boards of DCS. For each found task, agent receives:
   (a) A IG G of the found task;
   (b) The priority P of the task;
   (c) Subtasks $q_i$ (i = 1, K) statuses ("ready" / "working" / "new");
   (d) Maximum time to solve task $T_{max}$;
   (e) Maximum number of agents afforded by the task M;
3. Agent Aj evaluates the profit of solving all found incomplete tasks:
   (a) Agent $A_j$ ranks tasks their profitability value (P / Y) starting with the most profitable;
   (b) The agent $A_j$ looks for the next task from the list. If the list is empty, then go to step 4;
   (c) Agent $A_j$ receives parameters of agents in community and number of agents N;
   (d) If N = M, the agent go to step 3b;
   (e) Agent figures out if community have enough CNs to solve task in user-specified time using the following formula

$$T_s = \frac{\sum_{i=1}^{L} Y_i}{\sum_{m=1}^{D} S_m} \qquad (1)$$

where Yi is complexity of the one of the unsolved subtasks $q_i$, L is the quantity of unsolved subtasks, $S_j$ is the computing power of CN j, D is a quantity of CNs, presented in the community.

If $T_{max} < T_s$ then the agent removes a this task from a list of incomplete tasks and proceed to step 3b;

   (f) Agent $A_j$ joins the community;
   (g) If the performance of agents (found by formula (1)) is enough to perform users task in specified time, then agent marks the task as complete on the bulletin board;
4. If the agent Aj have successfully joined the community, then proceed to step 5, else agent goes into standby mode for time interval Twaiting and go to step 2;
5. If task is marked as complete, the community is successfully created and agents start solving the user task.

Steps above let the agents of DCS form a communities to solve incoming coherent tasks in time. The set of agents in the community depends on their actual performance. This ability make the community to adapt computing process to the dynamical variations of parameters of CNs.

After all these steps agents have to effectively distribute subtasks in community to make task be solved in user specified time.

## 5. Distribution of Task in Created Community

Once agents of DCS successfully create the community, they have to perform the distribution of subtasks between community nodes [15, 16]:

1. If there is free agent in community (does not have subtask to solve), then this agent sends messages to all agents in community about the launch of free subtasks allocation procedure.
2. To participate in this procedure that each agent of community follow steps:
   (a) Agent $A_j$ look for new subtask $q_R$;
   (b) For the found subtask $q_R$ agent $A_j$ calculates the time of its solving $T_{sol}^{R_j}$ on $CN_j$;
3. Agent Aj sends the resulting value $T_{sol}^{R_j}$ to all agents of the community;

4. When all agents in community receive all the values, they choose the lowest execution time $T_{sol}^{R_j}$. The agent As, who can afford this time take subtask qR, mark it as "working" and start its solution;
5. Agent As, requests the data required for the solution of task qR from all agents, which are owners of subset of vertices connected with vertex qR by incoming arc;
6. If one of agents finishes decision of some subtask, it marks this subtask as "ready";
7. After solving every subtask agent use its computational complexity YR and time of decision Tsol to determine the current performance of its CNs Sj, as Sj = YR / Tsol and sends the result values to all other agents and to bulletin board;
8. Agent estimate characteristics of the community to find out if the task can be solved in specified time. If community can not solve the task then agent mark it as "incomplete" and the creation of community restarts;
9. If there are any unsolved subtasks then go to step 1;
10. If the agent finishes solving of the last subtask in the user task it send the resulting data to the bulletin board and then to the user.

The main benefit of proposed algorithm is adaptive process of computation.

## 6. The Experiments and the Results

Verifying the efficiency of proposed algorithms for solving coherent task in DCS based on private CNs resources is very hard task because there is great amount of parameters in such system. To do that we create the program model of DCS, which allows modelling the DCS containing of up to 1000 CNs solving up to 250 tasks. Figure 3 shows the interface of created program model [17].

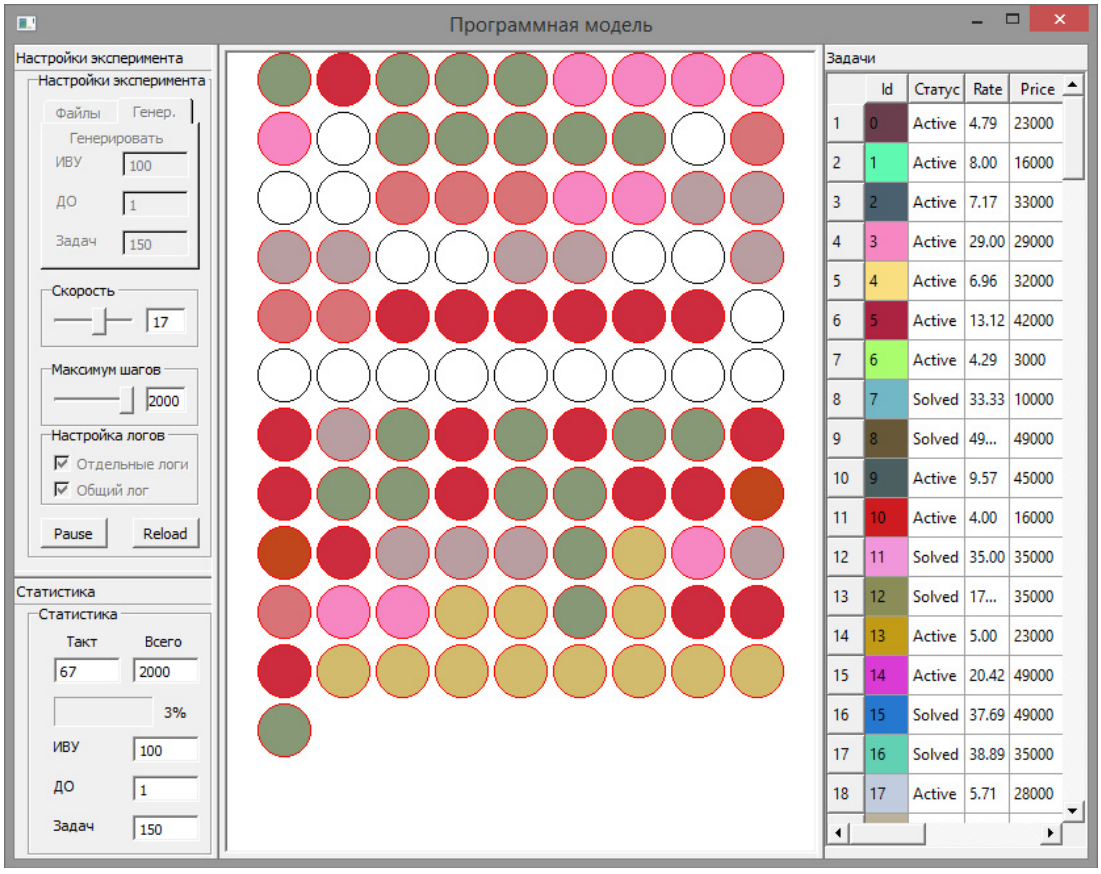


Fig. 3. interface of program model of DCS

The form in the right shows a set of tasks of the DCS, their statuses ("new" / "solving" / "ready") and its parameters (the maximum time for solution, complexity, number of virtual points for solution, etc.). Each task in this table have different color. A visual representation of DCS CNs can be found in the center (lot of circles). Each circle represent modelled CN and its color matches the color of the task that is solving at CN.

The main target of research is the practical proof of efficiency of proposed algorithms, but also we wanted to evaluate of the relative loss of processor time on CNs due to dispatching of the decentralized computing. That is why we had to carry out several series of experiments with different parameters of CNs and computer networks. While researching on the proposed algorithms, we used some types of real tasks we work with in Southern Federal University [18].

We decided to use percent of actual load at CNs, which can show loss of processor time due to organizational purposes and percent of tasks, solved in time, which show efficiency of system for end-user [19]. There are many important parameters in DCS that is why we decided to use planning of experiments: we divided parameters to three groups: Primary (system parameters) (P), Secondary (parameters of specific tasks and CNs) (S), and Tertiary (user-conditional parameters) (T). In the end, we decided to explore influence of following parameters of DCS:

- number of CNs in DCS (P1);
- frequency of incoming tasks (P2);

- computing power of CN (S1);
- bandwidth of network (S2);
- computational complexity of task (S3);
- volume of data to send between subtasks of the task (S4);
- priority (T1);
- number of subtasks (T2);
- time for solving the task (T3).

For primary parameters we decided to use absolute values, for secondary parameters relative values and for tertiary parameters limited random values. To make result more precise due to random values we carried out every experiment three times. Volume of this paper gives us no opportunity to show all the experimental results that is why below you can see tables (Table 1, Table 2) with parts of them.

Table 1. Results of experimental researches: actual load at CNs.

| № | Parameters | | | | Percent of actual load at CNs, % | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S4 | P1=500 P2=10 | P1=500 P2=50 | P1=500 P2=100 | P1=1000 P2=10 | P1=1000 P2=50 | P1=1000 P2=100 |
| 1 | - | - | - | - | 80 | 89 | 88 | 71 | 76 | 74 |
| 2 | - | - | - | + | 79 | 87 | 87 | 69 | 76 | 73 |
| 3 | - | - | + | - | 81 | 89 | 88 | 71 | 77 | 75 |
| 4 | - | - | + | + | 83 | 90 | 88 | 72 | 78 | 74 |
| 5 | - | + | - | - | 82 | 89 | 87 | 72 | 76 | 76 |
| 6 | - | + | - | + | 81 | 88 | 86 | 70 | 75 | 73 |
| 7 | - | + | + | - | 84 | 89 | 88 | 75 | 78 | 78 |
| 8 | - | + | + | + | 83 | 88 | 86 | 73 | 77 | 76 |
| 9 | + | - | - | - | 78 | 86 | 86 | 72 | 78 | 76 |
| 10 | + | - | - | + | 77 | 86 | 85 | 72 | 78 | 77 |
| 11 | + | - | + | - | 81 | 88 | 88 | 74 | 79 | 78 |
| 12 | + | - | + | + | 80 | 87 | 86 | 74 | 77 | 77 |
| 13 | + | + | - | - | 77 | 86 | 86 | 71 | 75 | 73 |
| 14 | + | + | - | + | 77 | 85 | 85 | 72 | 77 | 75 |
| 15 | + | + | + | - | 81 | 87 | 86 | 73 | 77 | 74 |
| 16 | + | + | + | + | 83 | 89 | 87 | 74 | 80 | 78 |
| Average | | | | | 80,43 | 87,68 | 86,68 | 72,18 | 77,12 | 75,43 |

Table 2. Results of experimental researches: tasks solved in time.

| № | Secondary parameters | | | | Tasks solved in time, % | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S4 | P1=500 P2=10 | P1=500 P2=50 | P1=500 P2=100 | P1=1000 P2=10 | P1=1000 P2=50 | P1=1000 P2=100 |
| 1 | - | - | - | - | 99 | 98 | 97 | 99 | 98 | 98 |
| 2 | - | - | - | + | 98 | 98 | 97 | 99 | 98 | 98 |
| 3 | - | - | + | - | 98 | 97 | 97 | 99 | 98 | 98 |
| 4 | - | - | + | + | 98 | 96 | 97 | 98 | 98 | 97 |
| 5 | - | + | - | - | 99 | 98 | 98 | 99 | 99 | 99 |
| 6 | - | + | - | + | 99 | 97 | 97 | 99 | 98 | 98 |
| 7 | - | + | + | - | 98 | 97 | 97 | 98 | 98 | 97 |
| 8 | - | + | + | + | 98 | 96 | 96 | 98 | 98 | 97 |
| 9 | + | - | - | - | 99 | 99 | 98 | 99 | 99 | 98 |
| 10 | + | - | - | + | 99 | 97 | 97 | 99 | 98 | 97 |
| 11 | + | - | + | - | 99 | 98 | 98 | 99 | 98 | 97 |
| 12 | + | - | + | + | 99 | 97 | 97 | 99 | 98 | 98 |
| 13 | + | + | - | - | 99 | 99 | 99 | 99 | 99 | 99 |
| 14 | + | + | - | + | 99 | 97 | 97 | 99 | 98 | 98 |
| 15 | + | + | + | - | 99 | 97 | 96 | 99 | 98 | 98 |
| 16 | + | + | + | + | 99 | 98 | 98 | 99 | 99 | 98 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Average | 98,68 | 97,43 | 97,25 | 98,81 | 98,25 | 97,81 |

The result table 1 shows that the percent of actual load at CNs was good: from 72% to 92%. That means that time loss due to organization ranged from 8% to 28% of the total time. Modern researches in the same directions [8][9] show that average effectiveness of solving coherent tasks in DCS is about 40%, which means that proposed methods are effective. You also can see from table 2 almost all the tasks in modelled DCS have been successfully solved in time

## Conclusion

In this paper we present the new method of adaptive multiagent organization of the distributed computations. The main benefit of presented method is possibility of using resources of many different private computers for solving coherent tasks.

The next step of researching of proposed algorithms planned by authors is creation of functional prototype of DCS, which will allow creating computing network on basis of computers of science laboratory.

## References

[1] Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001.
[2] Foster, I., Kesselman, C., Nick, J. and Tuecke, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Global Grid Forum.
[3] Vinogradov N. Analysis of modern state of GRID projects worldwide //Jet Info online http://www.jetinfo.ru/stati/?nid=0161d223cb9ed29f66470cbf7931ae22, 2007.
[4] About the Globus Toolkit, http://www.globus.org/toolkit/about.html.
[5] How SETI@home works, http://seticlassic.ssl.berkeley.edu/about_seti/about_seti_at_home_1.html.
[6] Thain, D., Tannenbaum, T., and Livny, M, Distributed Computing in Practice: The Condor Experience Concurrency and Computation: Practice and Experience, Vol. 17, No. 2-4, 323-356, 2005.
[7] Berman, F. Wolski, R. , Adaptive computing on the Grid using AppLeS, IEEE Transactions on Parallel and Distributed Systems, Vol. 14, 369 – 382, 2003.
[8] Poleshaev P.N. Experimental research of algorithms of task planning for GRID-systems with simulator / Material of international conference “High Performance Computations” HPC-UA’2012 (Ukraine, Kiev, 8-10 October 2012).
[9] Ivashko E.E., Golovin A.S. Computation effectiveness of BOINC-GRID / Material of international conference “High Performance Computations” HPC-UA’2012 (Ukraine, Kiev, 8-10 October 2012) P187-193.
[10] Michael Wooldridge, An Introduction to MultiAgent Systems, John Wiley & Sons Ltd, 2002, 366 pages.
[11] Kalyaev. A.I. Decentralized organization manager for GRID community-based agents. - Proceedings of the SFU. Technical sciences, № 8, 2011, 230-238.
[12] Kalyaev A.I , Melnik E.V. About one way of the decentralized organization of calculations in grid / Extreme robototechnics. // materials of international conference with elements of young scientists school – Saint-Petersburg, 2010. – P. 73-75.
[13] Tarasov V.B. From multiagent systems to intelligent organizations, 2002.
[14] Kalyaev A.I Multiagent Approach for Building Distributed Adaptive Computing System / Procedia Computer Science, Volume 18, 2013, Pages 2193-2202.
[15] Kalyaev A.I. Method and algorithms of the adaptive organization of the distributed computations in a decentralized GRID / International Young Scientists Conference High Performance Computing and Simulation 2012. Book of abstracts. - C.55-56.

[16] Kalyaev A.I. Method and algorithms of the adaptive organization of the distributed computations in a decentralized GRID / Herald of computer and information tecnoligies №4, 2012г.– Moscow– P. 28-33.
[17] Kalyaev A.I , Melnik E.V. One way of decentralized organization of GRID-computations / Infocommunications and computational tecnologies: materials of 3rd international conference – Ulan-Ude, 2010. - P.160-162.
[18] Hisamutdinov M.V., Korovin I.S., Kalyaev A.I. The Application of Evolutionary Algorithms in the Artificial Neural Network Training Process for the Oilfield Equipment Malfunctions' Forecasting / 2013 2nd International Symposium on Computer, Communication, Control and Automation (3CA 2013) Proceedings. – Atlantis press. Pages 253-257.
[19] Gorelova G.V., Radchenko S.A., Melnik E.V., Kalyaev A.I. Planning of experiment while researching new methods and algorithms of organization of distributed computations / Herald of computer and information tecnoligies №10, 2007г.– Moscow– P. 49-56.