ARTÍCULO

Influence of Inter-Agent Communication Cost to Diffusion Scheduling in Irregular Parallel Computations

Marek Grochowski, Ewa Tuska, Piotr Uhruski

Institute of Computer Science
Jagiellonian University
Kraków
{grochows,tuska,uhruski}@ii.uj.edu.pl

Abstract

The paper briefly presents the architecture of a multi-agent computing system with an emphasis on diffusion scheduling. Two binding energy formulas used by the diffusion scheduling algorithm are introduced: one respecting only requirements for the CPU and memory, and a second also utilizing the requirement for inter-agent communication. Two pairs of experiments with irregular parallel computations are presented with comparative results showing the efficiency in respect to inter-agent communication in both local and wide area networks.

Keywords: mobile agents, migration, computational tasks, communication cost, local scheduling, diffusion scheduling.

1 Introduction

High Performance Computing in distributed systems remains a hot topic in computational theory as well as practice. Almost every huge computational project requires a dedicated strategy that allows the internal needs of the problems being solved to be dealt with. In order to support the different needs of such projects, miscellaneous approaches and software environments were developed ([1], [2], [4], [8], [11]). Our approach utilizes a multi-agent systems paradigm. We have modeled, designed and built the Octopus platform and Smart Solid agent. The Octopus platform supports multi-agent applications in a varying and heterogeneous network environment, whilst the Smart Solid agent is a base of multi-agent application agents supporting computational tasks. Among others, the Smart Solid agent provides the following important features:
– dynamic creation and allocation of the proper number of computational units (agents),
– making autonomous scheduling decisions locally by agents,
– automatically scaling up to a varying set of computers and background load.
A new advantage to the Smart Solid agent scheduling strategy is the utilization of information about the remaining communication between computing agents. We expect a better allocation of agents causing a decrease of inter-agent communication time, which should be significant in wide computer networks.

2 Brief overview of the architecture

Octopus platform
The Octopus platform, from an multi-agent applications point of view, is a set of virtual computational nodes (VCN) with logical connection topology. Such topology defines the immediate neighborhood of each platform VCN. The Octopus platform also provides a description of the relative performance of all VCNs, information about connection speed between VCNs through logical paths, and the current load of each VCN.

Smart Solid agent
The Smart Solid agent is composed as a pair $A_i = (T_i, S_i)$ where:

T_i is the computational task, which includes the computational code and all data required for computation,

S_i is an agent shell responsible for the agent's specific logic.

The index i stands for an unambiguous agent identifier.

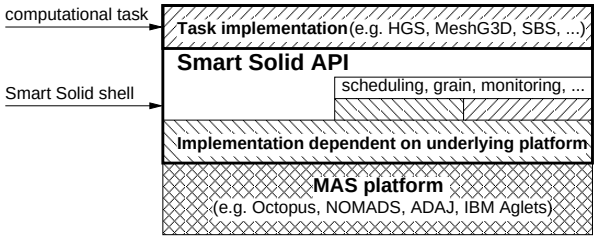

Figure 1. Smart Solid agent architecture.

Each task T_i has to denominate the current requirement for computational power (E_i, M_i) where:

E_i is the task remaining time measured in units common for all application tasks,

M_i is the RAM requirement in bytes.

Task T_i can also provide information about its future communication with other tasks described by the set of pairs $C_i = \{(T_\zeta, data_\zeta)\}$ where T_ζ is another application task, and $data_\zeta$ is the amount of data, expressed in bytes, which will be exchanged between tasks T_i and T_ζ.
In order to perform agent partitioning and migration, the task must allow for the pausing and continuation of it's computation. The task can be also partitioned into two subtasks, but the task partitioning possibility strongly depends on the computational problem to be solved.

Smart Solid API (white rectangles in Figure 1) is a set of interfaces and routines that can be referenced by a computational task in order to access the supporting platform and Smart Solid services. Current implementation of the Smart Solid API allows to run a multi-agent application under the control of the Octopus platform, but the chosen architecture enables the utilization of other platforms without changes to the computational task's implementation (see [3], [5], [6], [7]). The Smart Solid shell, by it's implementation, also realizes all agent specific logic such as: searching for resources required by contained task, or balancing the load of computational nodes located in the immediate neighborhood of the VCN on which the agent is allocated. These functions are realized by diffusion scheduling using migration and partitioning of application agents.

Multi-agent application
The multi-agent computational application has to be designed to start from one task T_1 executed on any platform VCN. Every agent, especially the starting one containing task T_1, may partition itself and produce two new agents containing subsets of the initial computed problem. An application evolves over time because of agents individually trying to accomplish their goals.

3 Diffusion scheduling

The diffusion scheduling idea is the most important advantage of our strategy for supporting large scale computations in multi-agent systems. The presented scheduling algorithm is an analogy to the molecular diffusion phenomenon (see [13]). Therefore, we have introduced the following diffusion parameters:

Binding energy $E_{i,j}$ of the agent A_i allocated on VCN P_j is characterized by the following

conditions:
– $E_{i,j}$ is a descending function of E_i and C_i,
– $E_{i,j}$ is a nonascending function of the load on VCN P_j,

Binding energy gradient is a vector:
$\nabla_{i,j} = ((j,l), E_{i,l} - E_{i,j})$:
$E_{i,l} - E_{i,j} = \max_{\zeta \in Q_j}\{E_{i,\zeta} - E_{i,j}\}$, where Q_j is a set of indices of VCNs from the immediate neighborhood of P_j that are less loaded than P_j,

Local load concentration of P_j:
$L_j = \frac{E^j_{total}}{perf_j(M^j_{total})}$, where
$E^j_{total} = \sum_{i \in \omega_j} E_i$, $M^j_{total} = \sum_{i \in \omega_j} M_i$,
ω_j – is a set of indices of all agents allocated on VCN P_j,
$perf_j$ – relative performance of VCN P_j strongly dependent on available RAM.

Diffusion scheduling algorithm:

if $Q_j = \emptyset$ **then**
  continue T_i
else
  compute $\nabla^t_{i,j}$
  if $E_{i,l} - E_{i,j} > \epsilon$ **then**
    pause T_i;
    migrate along the gradient $\nabla^t_{i,j}$;
    continue T_i
  else
    partition $T_i \rightarrow \{T_{i_1}, T_{i_2}\}$;
    create $\{A_{i_j} = (T_{i_j}, S_{i_j})\}$, $j = 1, 2$;
    disappear;
  end if
end if

The above algorithm is processed by the Smart Solid agent shell just after agent creation, after each migration between VCNs, and also when the underlying platform gives notice of a significant change of local resource utilization. If any of the diffusion algorithm operation can not be performed completely (the algorithm is computed in the two face distributed transaction), then this algorithm is interrupted and the agent continues the contained task computation. Such a condition prevents an agent from going into an unexpected state caused by some denial circumstances, like task inability to perform partitioning.

4 Experiments

Our goal is to show the influence of respecting the inter-agent (inter-task) communication cost in presented diffusion governed scheduling during the processing of irregular parallel computation. We executed four different experiments. First, we compared the same computations run in the same computational environment, but utilizing different binding energy formulas: one respecting inter-agent communication requested by computational tasks, and the second in which communication cost is omitted. Additionally we performed such a pair of experiments in two different network environments: having the same bandwidth of network connections between Octopus platform VCNs, and a second, simulating a wide area network by having lower bandwidth between two similar subnets of the Octopus platform VCNs.
If communicating agents are located on distant VCNs because of a lack of agents' affinity in the scheduling rule, then communication time decreases overall application performance. We are trying to show the effectiveness of the diffusion scheduling rule respecting inter-agent communication in comparison to disorderly diffusion scheduling respecting only task requirements for CPU time and memory.

4.1 Binding energy formulas

As presented in the previous section, we have assumed only two conditions for the binding energy formula. This formula can be formed in accordance with different factors which can be relative to the characteristic of computations being performing.
For the experiments we have defined the following simplified formulas:

Binding energy omitting communication

$$E_{i,j} = (\xi_1 E_i + \frac{\xi_2 E^j_{total}}{perf_j(M^j_{total})})^{-1} \qquad (1)$$

where
ξ_1, ξ_2 – scaling coefficients.

The above formula simply carries the influence of the agent load requirement (E_i) and the VCN utilization (the second element in the sum). Such an

expression of binding to the node is good enough for multi-agent applications in which the weight of each agent is comparable and inter-agent communication is scattered.

Binding energy respecting communication

$$E_{i,j} = (\xi_1 E_i + \frac{\xi_2 E^j_{total}}{perf_j(M^j_{total})} + \xi_3 c_{i,j})^{-1} \quad (2)$$

where
$c_{i,j}$ – entire time needed to perform the requested communication C_i from node P_j,
ξ_1, ξ_2, ξ_3 – scaling coefficients.

The second formula also includes the third element ($c_{i,j}$) which can weaken the binding of the agent A_i to the VCN P_j when the cost of communication between the agent A_i and other application agents is large accordingly. On the other hand, it is intended to increase the affinity between communicating application agents and to enhance the allocation of such agents on the same or neighboring VCNs. At the same time this element should push aside the agents which are not tied by the communication with the agent A_i.

4.2 Sample problem description

The sample problem under consideration is stochastic in nature and an irregular parallel computation built on the base of the HGS – genetic global optimization algorithm ([10], [12]). Modification to the multi-agent implementation of the HGS algorithm rests on the addition of intensive message exchange between subagents and their parents in division hierarchy – during computation every child agent exchanges many messages with it's parent. Such intensive communication became a part of the whole computational process.
Because of the stochastic nature of the chosen computational problem, we had to undertake some preliminaries to make a comparison of computational results reasonable. Before the exact test run of all pairs of experiments, we ran the computations in such a way we could store all randomly generated values. Then we reused these stored values for the execution of the described experiments and forced them to perform their genetic computations with the same characteristics.

4.3 Hardware environment

We have utilized a set of 61 computers (see Figure 2) running under the control of the Linux operating system. On each computer one VCN was run. These computers have the following specifications:
– **VCN A** :
double Pentium III 1GHz, 1GB of RAM
– **VCN B0** – **VCN B14** :
Celeron 1.7GHz, 512MB of RAM
– **VCN C0** – **VCN C14** :
Pentium III 1GHz, 512MB of RAM
– **VCN D0** – **VCN D9** :
Celeron 1.7GHz, 512MB of RAM
– **VCN E0** – **VCN E9** :
Pentium 4 1.5GHz, 512MB of RAM
– **VCN F0** – **VCN F9** :
Celeron 600MHz, 512MB of RAM

4.4 Experiment with homogeneous platform connections

The purpose of the experiment is to compare some behaviors of multi-agent applications working in the local area network with utilization of different binding energy formulas by the diffusion scheduling algorithm.

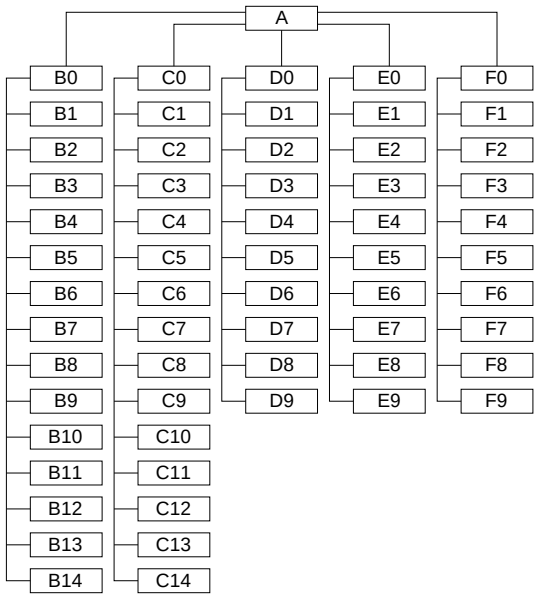

Figure 2. Topology of utilized VCNs.

Figure 2 presents the logical topology of the utilized network of computers. All cable connec-

tions in the local area network are very similar, and bandwidth available for message exchange between application agents is not differentiable.

The starting agent of the multi-agent application is delivered to the node VCN A. The first action of this agent is to perform a diffusion scheduling algorithm (see section 3), and after allocation on some VCN, it starts performing the HGS algorithm. During computation, new child agents are created and process the same algorithms as the starting agent, but with the subsets of the parent's data.

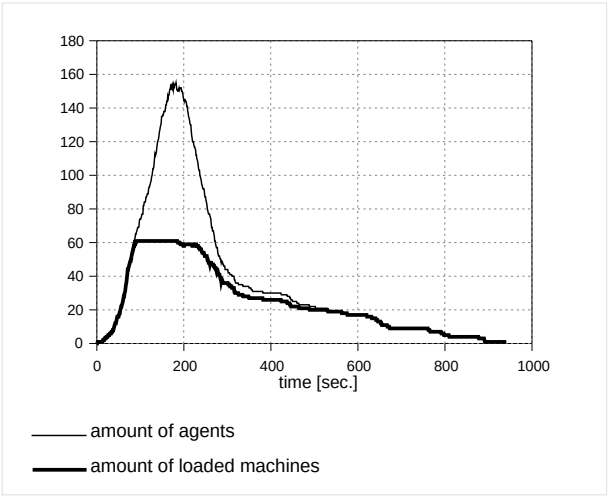

Figure 3. The dynamics of the system utilizing the binding energy formula (1).

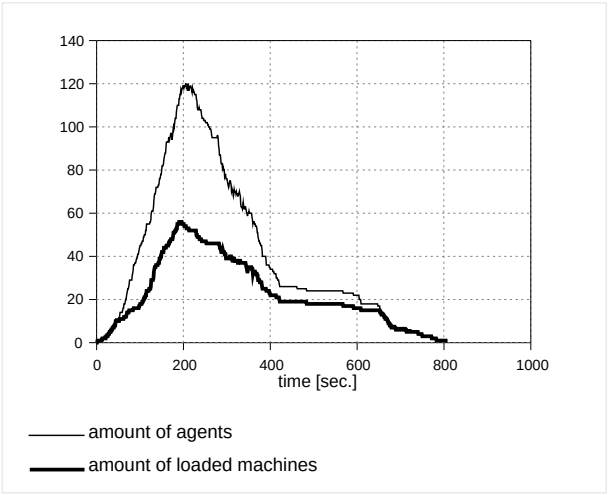

Figure 4. The dynamics of the system utilizing the binding energy formula (2).

Binding energy formula	Agents amount	Total times [sec.]		Parallel computation time [sec.]
		Migration	Communication	
(1)	203	126	29587,2	941
(2)	203	195	26131,2	808

Table 1. The execution times of HGS computations with diffusive scheduling.

Figures 3 and 4 show the dynamics of the computing system for two executions of the same experiment but with different binding energy formulas (see section 4.1). The thin line depicts the number of computing agents, whilst the thick line depicts the number of utilized VCNs.
As we can see, at the beginning of the process the multi-agent HGS application creates a large amount of agents, which are looking for required resources like CPU power and RAM. In the second example the requirement for good connection conditions is also fulfilled allowing agents to communicate efficiently with their parents. In the first example VCNs are occupied by new agents almost immediately after creation (two lines in Figure 3 begin going up almost in the same speed), but in the second example agents need some time to migrate to the VCN from which they could communicate with better efficiency – usually they migrate one after another through similar paths. This slows down the migration of other agents, which we can see in Figure 4 as a decrease in the growth of the thick line representing the VCNs' utilization.

Even though the binding energy respecting the agents' communication requirements slow down the resources allocation, it allows the agents to be better scheduled and then allows them to perform their computation with communication faster than agents from the first example, which had been allocated earlier but finished their computation later.

4.5 Experiment in wide area network

Even more important than respecting the agents' communication requirements in the local area network is to not omit the communication factor in the wide area network.
Our connection topology for the second experiment remains the same as the first experiment except for two connections which now have three

times lower throughput – this has been done to simulate distant connections. Figures 5 and 6 depict these slower connections with shaded lines.

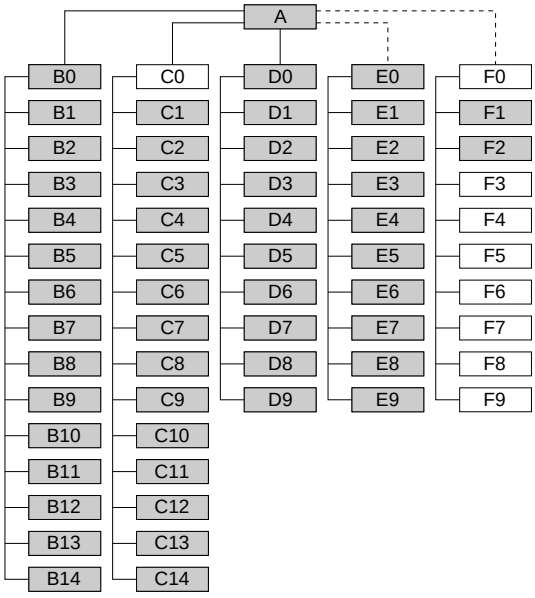

Figure 5. Topology of the Octopus platform simulating distant subnetworks. Marked VCNs were utilized during the experiment with binding energy (1).

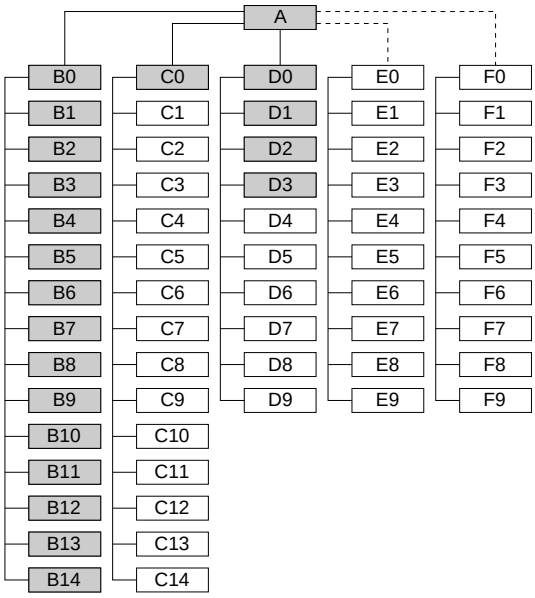

Figure 6. Topology of the Octopus platform simulating distant subnetworks. Marked VCNs were utilized during the experiment with binding energy (2).

For this experiment we have chosen a smaller problem (only 52 agents) to solve using the HGS multi-agent algorithm. Such a small computational application in a wide area network can lose a lot of its performance when only a few intensively communicating agents go to distant subnets.

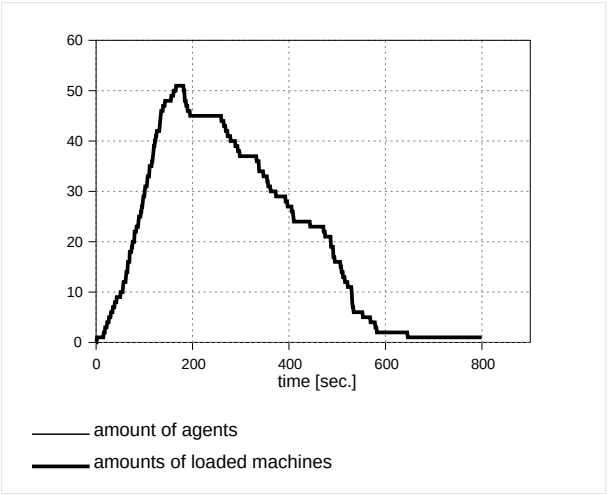

Figure 7. The dynamics of the system utilizing the binding energy formula (1).

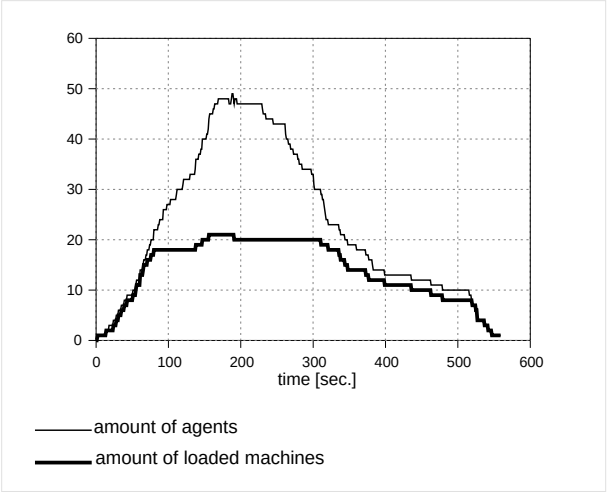

Figure 8. The dynamics of the system utilizing the binding energy formula (2).

Figures 7 and 8 show the dynamics of the computing system for two executions of the same experiment but with different binding energy formulas (see section 4.1).

In this experiment the immediate resource's utilization is more visible (the communication is omitted in the binding energy calculation) – the thick line covers the thin line in Figure 7. This is because of the small number of migrating and communicating agents in comparison to the number of available VCNs – agents can quickly, without conflicts on the network, find suitable VCNs. We can also observe that the second binding energy formula forced all agents to not utilize every available VCNs, and herded them onto VCNs having better communication conditions (see Figure 6).

Binding energy formula	Agents amount	Total times [sec.]		Parallel computation time [sec.]
		Migration	Communication	
(1)	52	53	14102,4	799
(2)	52	195	9590,4	559

Table 2. The execution times of HGS computations with diffusive scheduling.

Table 2 presents execution times for computations performed with and without the utilization of the information about inter-agents' communication. The first of the processed computations – omitting the agents' communication requirements – took over 40% more time than the second computation. This happened because of some set of agents which migrated to distant VCNs **F1**, **F2** and **E0** – **E9**. Whereas, in the second computation, agents migrated only among the local area network and kept away from too costly communication when required resources could be found on local VCNs.

5 Conclusions

The presented first experiments performed with utilization of the new diffusion scheduling rule, respecting the requested communication requirements by computing agents, confirmed the usefulness of the proposed extension. Other valid binding energy formulas might improve the diffusion scheduling efficiency even more – the formula might be complied with dependencies between E_i and VCN utilization. More experiments must be performed to accurately test diffusion scheduling efficiency utilizing different binding energy formulas.

Computing the local scheduling rule that utilizes the communication factor does not decrease overall system performance. We observe that even with more information required by a particular agent and delivered by the Octopus, the application executes faster. This shows the efficiency and flexibility of the solutions based on the Smart Solid agent architecture and supported by the Octopus platform.

The diffusion rule utilizing the communication factor significantly improves diffusion scheduling in networks consisting of many distant subnets.

References

[1] Thain D., Tannenbaum T., Livny M.; Condor and the Grid, in Grid Computing Making the Global Infrastructure a Reality, Berman F., Hey A., Fox G. eds. John Willey&Sons 2003

[2] Luque E., Ripoll A., Cortés A., Margalef T.: A distributed diffusion method for dynamic load balancing on parallel computers. *Proc. of EUROMICRO Workshop on Parallel and Distributed Processing*, San Remo, Italy, January 1995. IEEE CS Press.

[3] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill, and R. Jeffers. Strong mobility and ne-grained resource control in NOMADS. *Lecture Notes in Computer Science*, Vol. 1882, Springer-Verlag 2000, pp. 2-15.

[4] N. Suri, J. M. Bradshaw, M. R. Breedy, P. T. Groth, G. A. Hill, R. Jeffers, and T. S. Mitrovich, An Overview of the NOMADS Mobile Agent System, *Proceedings of the European Conference on Object-Oriented Programming (ECOOP'2000)*, Sophia Antipolis and Cannes, France, 2000.

[5] V. Felea, R. Olejnik, and B. Toursel. ADAJ: A Java Distributed Environment for Easy Programming Design and Efficient Execution, *Schedae Informaticae*, Vol. 13, 2004, pp. 9-36

[6] IBM, http://aglets.sourceforge.net, IBM Aglets.

[7] P. Jurczyk, M. Golenia, M. Malawski, D. Kurzyniec, M. Bubak, and V. S. Sunderam, A system for distributed computing based on H2O and JXTA. In Cracow Grid Workshop 2004, Kraków, Poland, 2004. (accepted).

[8] Globus, Open Grid Services Architecture, http://www.globus.org/ogsa/

[9] Golub G., Ortega J. M.: Scientific Computing. An Introduction with Parallel Computing. Academic Press, 1993.

[10] Schaefer R., Kołodziej J.: Genetic search reinforced by the population hierarchy. in *De Jong K. A., Poli R., Rowe J. E. eds. Foundations of Genetic Algorithms 7* Morgan Kaufman Publisher 2003, pp. 383-399.

[11] Uhruski P., Onderka Z.: The Object Oriented Platform for the Process Migration in the Heterogeneous Networks. *Schedae Informaticae*, Issue 11, Jagiellonian University 2002, pp. 99-114

[12] Momot J., Kosacki K., Grochowski M., Uhruski P., Schaefer R.; Multi-Agent System for Irregular Parallel Genetic Computations. *Lecture Notes in Computer Science*, Vol. 3038, Springer 2004, pp. 623-630.

[13] Grochowski M., Schaefer R., Uhruski P.: Diffusion Based Scheduling in the Agent-Oriented Computing Systems. *Lecture Notes in Computer Science*, Vol. 3019, Springer 2004, pp. 97-104.