International Journal of Innovative
Computing, Information and Control
Volume **9**, Number **3**, March **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 1171–1188

ERMS: AN EVOLUTIONARY REORGANIZING MULTIAGENT SYSTEM

Barna Laszlo Iantovics[1] and Constantin-Bala Zamfirescu[2]

[1]Informatics and Mathematics Department
Petru Maior University
No. 1, Iuliu Maniu St., Tg. Mures 540088, Romania
ibarna@upm.ro

[2]Lucian Blaga University
No. 10, Victoriei St., Sibiu 550024, Romania
zbc@acm.org

Received December 2011; revised April 2012

Abstract. *Development of computational systems that intelligently can solve problems represents an important research direction. Many results described in the literature prove, that the intelligence of a computational system can offer advantages in the problems solving versus a system that does not have intelligence. The adaptation is considered to be an important property of many intelligent systems. Sometimes the adaptation is realized by learning. In this paper, we propose a novel adaptive multiagent system called ERMS (Extended Centralized Multiagent System with Cooperative Evolutionary Reorganization Capacity), which uses an evolutionary learning technique in order to improve the efficiency of the undertaken problems-solving. ERMS represents an extension of a previously developed multiagent system called CCER (Centralized Multiagent System with Cooperative Evolutionary Reorganization Capacity). The adaptivity of the ERMS multiagent system, consists in the capacity to reorganize its structure based on the information available about the received problems for solving. The obtained results prove that a multiagent system successfully can use evolutionary algorithms to discover emergent patterns of reorganization for the efficient solving of the undertaken problems. In case of complex systems composed from a large number of interacting components, such emergent behavior of the systems, if have as results improvements (i.e., autonomy, efficiency and flexibility) in the problems solving it could be associated with intelligence.*

Keywords: Multiagent system, Evolutionary algorithm, Genetic algorithm, Natural computation, Adaptation, Evolutionary learning, Cooperative problem solving, Intelligent agent, Emergence, Complex system, Evolutionary system

1. **Introduction.** "Intelligent" systems (usually agent-based) are used in many domains of sciences. The development of the next generation intelligent systems (more intelligent than the actually developed) is an important research direction. Highly intelligent systems will be inherently complex having many interacting components (sometimes hybrid components) that require a very long time for their development. In the case of many very complex systems all the necessary data, information and knowledge for their development are obviously unavailable. Another aspect consists in the fact that it is necessary to elaborate more consecutive versions. Even if an intelligent system has some autonomy in increasing its intelligence sometimes by adaptation, there exists a point when the system is unable to make other improvement by itself and it is necessary a human intervention for the increasing of the system's intelligence.

Intelligent agent-based systems represent one of the most important approaches used for autonomous, efficient and flexible solving of difficult problems (tasks) and/or large

numbers of simple problems in many domains [7, 20, 44]. The main motivation consists in the properties of the agents that differentiate them from other computable systems. Agent-based systems can be endowed with capacities that allow to intelligently process computational hard (difficult/complex) problems [12, 19, 25]. There are many applications of the intelligent agents in many domains, including health care [18, 26, 32, 33, 34], which extend traditional developments. Recent implementations include applications, like *telehealth* [43], *analysis of spread simulation of infectious disease* [46], *web-enabled healthcare computing* [8], *patients monitoring* [17], *patients management* [30, 31], *medical diagnosis* [44] and *ubiquitous healthcare* [29].

In this paper, an adaptive multiagent system called *ERMS* (*Extended Centralized Multiagent System with Cooperative Evolutionary Reorganization Capacity*) is proposed. *ERMS* represents an extension, in order to operate in case of larger numbers of agents, of the *CCER* multiagent system (*Centralized Multiagent System with Cooperative Evolutionary Reorganization Capacity*) developed during our previous researches [23]. The results obtained during the development of *CCER* and *ERMS* multiagent systems prove that evolutionary learning algorithms can be successfully used by multiagent systems to establish how to adapt a more efficient problem-solving strategy when the emergent patterns of reorganization of the systems' structure can be discovered.

The upcoming part of the paper is organized as follows. Section 2 analyzes some aspects related with the intelligent adaptive systems. In Section 3 the novel adaptive multiagent system called *ERMS* is described – the evolutionary learning algorithm used by the *ERMS* system in order to learn different reorganizations is presented. At the end of Section 3 the correctness and efficiency in operation of the *ERMS* system is analyzed. In Section 4, *ERMS* System Intelligence is considered. In Section 5, the conclusions of the research are presented.

2. **Intelligent Adaptive Systems.** Many difficult problems solving require a specific sort of computational intelligence (capacities to intelligently handle the difficulties of the problems-solving) [12, 45]. In general a problem is considered difficult (computational hard) based on considerations such as the solving requires a large amount of resources (some times distributed resources); the solving requires a large quantity of problem solving knowledge (some times distributed knowledge); the solving requires a large variety of problem solving knowledge; the problem description contains different types of uncertainties (missing or erroneous data), etc. Intelligent agents must precisely and flexibly handle the solving of difficult problems or solving large amounts of difficult or simple problems. Cooperative agents can form multiagent systems, in that they can collaborate during the problems solving in order to make easier their solving and to improve the accuracy of the obtained problems solutions [12, 19, 20, 45, 47].

Many times it is difficult or even impossible to endow an agent with the necessary knowledge at the moment of its creation. Motivations may consist in considerations, like: some information initially is not known and/or some information are changing in time. These reasons motivate the necessity of endowment of the agents with autonomous learning capability [12, 19]. The purpose of learning consists in improvements in the efficiency and accuracy of the problems-solving.

In literature there are many adaptive multiagent systems [3, 4, 9, 10, 11, 15, 16, 23, 27, 28, 38, 39]. The adaptation many times is realized by leaning. One type of adaptation of a multiagent system consists in the reorganization of the problem-solving resources, in order to solve more efficiently the undertaken problems [10, 11, 23]. In a multiagent system the agents members of the system are called *problem-solving resources* (the agents solve the undertaken problems by the system) [10, 11]. An agent uses different resources (i.e.,

processing power, memory) during its life cycle for the problems solving. The significance of adaptability of the multiagent systems described in the papers [10, 11] is to allocate the resources that are necessary for the problems-solving. The adaptation in a multiagent system is realized at the system's level.

The papers [3, 15, 28] present some adaptive multiagent systems that efficiently use the available resources in the problems-solving process. In order to satisfy either the surplus or lack of resources, the presented multiagent systems self-organize depending on the necessities. The agents are endowed with capacity of *decomposition* and *composition*. The decomposition represents the capacity of an agent to create an identical copy with itself. The copy of an agent can solve the same set of problems like the initial agent (has the same problem-solving specializations). By composition, two agents are combined to become one in order to free computational resources. An agent obtained as a result of the composition of more agents has all the specializations of the agents used in the composition.

TRACE [10, 11] is an adaptive multiagent system, formed from coalitions of agents. The adaptability of the *TRACE* system consists in the capacity of each coalition to buy or sell resources (agents). Within the frame of the *TRACE* multiagent system, the resources are distributed in coalitions. The distribution is realized, depending on the necessities or on the surplus of resources within each of the coalitions. A coalition desires to buy resources, if it does not detain all the necessary resources for solving of problems. A coalition can sell resources if it does not need all the resources.

3. ERMS Multiagent System Description.

3.1. Previous researches.

CCER is an adaptive coalition-based multiagent system developed during our previous researches [23]. The adaptability of the system consists in the capacity to reorganize its structure depending on the received problems for solving. For the establishment of the reorganization an evolutionary learning algorithm is used. For the problems allocation for solving, a novel allocation protocol is used that represents an adaptation of a centralized problem allocation protocol as described in [12, 45].

The main advantage of the *CCER* multiagent system consists in the capacity to adapt its structure in order to solve more efficiently the undertaken problems. Simulation results show the increased efficiency of the *CCER* multiagent system when larger numbers of problems are transmitted for solving [23]. *CCER* system can reorganize its structure at the beginning of a problems solving cycle if some specific information, called *problems pattern* about the problems transmitted for solving exist. A problems solving cycle begins at the undertaking of a set of problems, and is finished when all the problems are solved. The establishment of reorganization requires a polynomial complexity search in a rule base. Thus, in the verification of the preconditions of some rules from the rule base, the identified rule is fired, neglecting the rest of the rules.

3.2. ERMS multiagent system architecture.

Each agent member of an intelligent cooperative multiagent system must have a *role*. A role defines the manner in which the agents who undertake that role contribute to the problems solving in the multiagent system [12, 19]. An agent who undertakes a role must have a set of specializations and necessary resources which allows to the agent to fulfill its role. A *multiagent system architecture* specifies different generic information [12, 45] (i.e., existent roles, relations between the roles, organization of the agents, specific cooperative problems solving methods used in the frame of the system, etc.) about the multiagent systems endowed with that architecture. The information that must describe a multiagent system architecture depends on conditions, like: complexity of the system, number of member agents, the

problems that have to be solved, the cooperative problem solving methods that can be used by the multiagent system, etc.

In the following, we propose a novel multiagent system architecture called *ERMS* (*Extended Centralized Multiagent System with Cooperative Evolutionary Reorganization Capacity*). We call *ERMS* a multiagent system with the *ERMS* architecture. The proposed architecture defines: a partially centralized multiagent system organization; a specific adaptation based on the reorganization of the system; a specific evolutionary learning algorithm; a specific cooperative problems-solving and the roles that can be undertaken by the agents. We call a problems pattern the description, values of different parameters related to a set of problems transmitted for solving, which may consists in information, like: what type of problems are transmitted for solving, the number of problems transmitted for solving.

In an *ERMS* multiagent system there are defined the following roles (see Figure 1): by centralized problem allocation for solving denoted *supervisor*; by problem-solving denoted *contractor*; by cooperative problem-solving (problem solving and local problem allocation for solving) denoted *manager*. An agent who undertakes the *contractor* role during the fulfilling of the role will solve problems. An agent who undertakes the *manager* role during the fulfilling of the role will solve problems and allocate problems for solving to some agents members of the system. An agent who undertakes the *supervisor* role during its life cycle will allocate problems for solving to other agents. In the multiagent system there is only one agent with supervisor role. The agent initially established with the supervisor role is not changing its role during the multiagent system's life cycle.

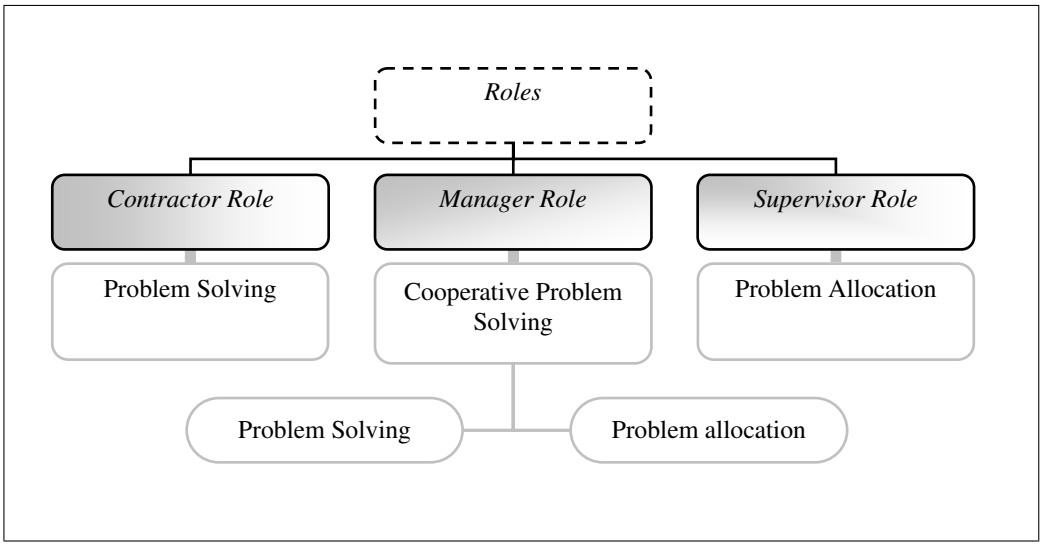


FIGURE 1. The roles in the ERMS multiagent system

An *ERMS* multiagent system at a problems-solving cycle is composed from a set Ms (1) of agents, each of them having as unique identifier a natural number that is not changed during its life cycle. Each agent, except the supervisor, is associated one of the following abbreviations: "Ct" for *contractor agents* (for example, the notation Ct_i has as meaning the agent with the identifier i has contractor role), "Mg" for *manager agents* (for example, the notation Mg_j has as meaning the agent with the identifier j has manager role) and "Fr" for *free agents* (for example, the notation Fr_k has as meaning the agent with the identifier k has contractor role and it operates as free agent) that illustrate the specific of the agents contribution to the problems-solving. The agents, except the supervisor, can change their role during the system's operation. For example, an agent at

a problems-solving cycle may operate as contractor and at another problem solving cycle as manager.

$$Ms = Co \cup \{Su\} \cup Fr. \tag{1}$$

Fr denotes a set of agents called free agents. A free agent Fr_h ($Fr_h \in Fr$, $ID(Fr_h) = h$) has contractor role, $Role(Fr_h) = contractor$. A free agent does not belong to any coalition. Su is the agent with the supervisor role, $Role(Su) = supervisor$. Co represents the set of coalitions of agents.

A coalition of agents is composed from one or more agents with *manager* role and usually more agents with *contractor* role. In a coalition of agents each agent with contractor role is subordinated to a single agent with manager role. Figure 2 presents a coalition of agents denoted Co_q. Mg_b ($Mg_b \in Co_q$, $role(Mg_b) = manager$, $ID(Mg_b) = b$) and Mg_c ($Mg_c \in Co_q$, $role(Mg_c) = manager$, $ID(Mg_c) = c$) represent the managers of the coalition. $Ct_1, Ct_2, \ldots, Ct_x$ ($\{Ct_1, Ct_2, \ldots, Ct_x\} \subset Co_q$) represent the contractors subordinated to Mg_b. $Ct_a, Ct_b, \ldots, Ct_m$ ($\{Ct_a, Ct_b, \ldots, Ct_m\} \subset Co_q$) represent the contractors subordinated to Mg_c. $P_f, \ldots, P_v$ represent the problems that must be solved by Mg_b and the subordinated agents. $P_g, \ldots, P_z$ represent the problems that must be solved by Mg_c and its subordinated agents. The dashed arrows between the agents used in Figure 2, illustrate the communication and cooperation links between the agents. t_a indicates a link by cooperation type (problem allocation for solving) between the agents.

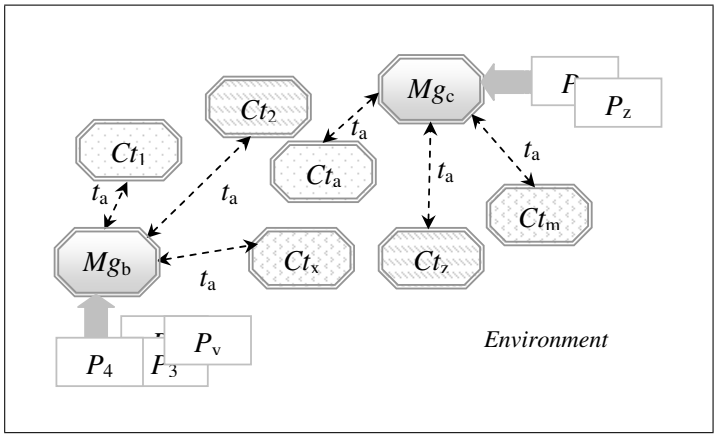


FIGURE 2. A coalition Co_q of agents in the ERMS system

Figure 3 presents an *ERMS* multiagent system at a problems solving cycle, composed from the disjointed coalitions $Co = \{Co_1, Co_2, \ldots, Co_n\}$ ($\forall i \neq k, Co_i \in Co, Co_k \in Co, Co_i \cap Co_k = \varnothing$), the supervisor agent Su and the free agents $Fr = \{Fr_1, Fr_2, \ldots, Fr_m\}$. $P = \{P_1, P_2, \ldots, P_y\}$ represents the problems transmitted for solving to Su. The dashed arrows used in Figure 3 illustrate the cooperation links in the multiagent system. t_b indicates a link by cooperation type between the supervisor agent and a free agent. t_c indicates a links by cooperation type between the supervisor agent and a coalition of agents.

An agent Ct_k with contractor role ($role(Ct_k) = contractor$, $ID(Ct_k) = k$) is endowed with a specialization set $Spec(Ct_k) = \{S_1, S_2, \ldots, S_r\}$, which allows the solving of problems from a set $Cl = \{Cl_1, Cl_2, \ldots, Cl_r\}$ of classes of problems; where S_i represents the specialization necessary for solving of the problems from the class Cl_i of problems ($\forall i = \overline{1, r}, S_i \rightarrow Cl_i$).

An agent Mg_k with manager role ($role(Mg_k) = manager$, $ID(Mg_k) = k$) is endowed with a specialization set $Spec(Mg_k) = \{S_1, S_2, \ldots, S_r\} \cup \{A_1, A_2, \ldots, A_e\}$. $S_1, S_2, \ldots, S_r$

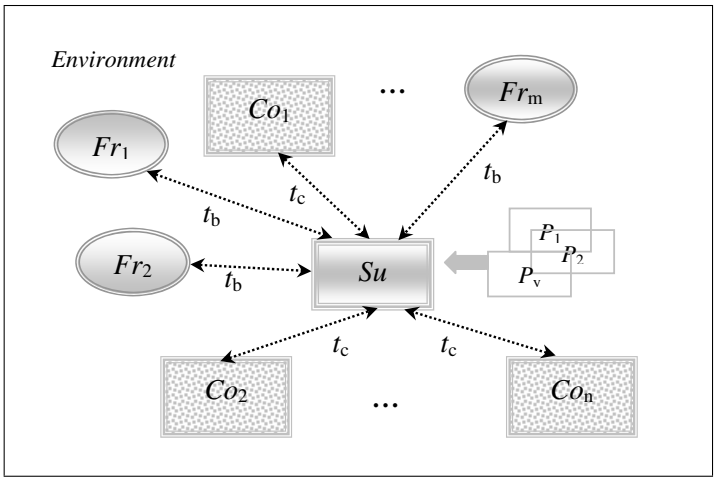


FIGURE 3. ERMS multiagent system at a problems solving cycle

allows the solving of the problems from a set $Cl = \{Cl_1, Cl_2, \ldots, Cl_r\}$ of classes of problems; where S_i represents the specialization necessary for solving of the problems from the class Cl_i of problems. If Mg_k is manager in the coalition Co_y, then $A_1, A_2, \ldots, A_e$ represent knowledge detained by Mg_k about the subordinated contractor agents from Co_y.

As examples of information detained by a manager agent Mg_m from a coalition Co_u ($Mg_m \in Co_u; role(Mg_m) = manager; ID(Mg_m) = m$), about the subordinated contractor agents, we mention: the number of contractor agents from Co_u; the specializations of each contractor agent; the information detained about the problems that are undertaken for solving by each contractor agent.

Su detains information about the coalitions of agents and the free agents. As examples of information detained by Su about a free agent Fr_p ($role(Fr_p) = contractor$, $ID(Fr_p) = p$), we mention: Fr_p specializations and capacity; the problems that are currently solved by Fr_p. As examples of information detained by Su about a coalition Co_r ($Co_r \in Co$), we mention: the number of member agents of Co_r; the capacity of the contractor agents members of Co_r; the problems that are currently solved by Co_r. The problems are transmitted for solving to Co_r by Su. Each manager from Co_r transmits the obtained problems solutions to Su. However, Su knows when in the Co_r coalition is finished an undertaken problem-solving.

3.3. **The ERMS system operation.** The *ERMS* multiagent system represents an extension of the *CCER* multiagent system [23]. One of the improvements consists in the use of coalitions with more manager agents, each of them with a set of subordinated contractor agents. This improvement was realized in order to eliminate the excessive centralized architecture of the *CCER* multiagent system. The centralization in a large-scale multiagent system may be a bottleneck in the operation of the system. Another adaptation consists in the use of genetic problem solving specializations (problem solving methods based on genetic algorithms).

Figure 4 illustrates a single problem-solving cycle in the *ERMS* multiagent system. A P_h problem-solving begins at its undertaking and is finished when its solution So_h is obtained. Fr_v ($Fr_v \in Fr$, $role(Fr_v) = contractor$, $ID(Fr_v) = v$) represents a free agent. Co_a ($Co_a \in Co$) represents a coalition of agents. Mg_y ($Mg_y \in Co_a$, $role(Mg_y) = manager$, $ID(Mg_y) = y$) a manager in Co_a. Ct_s ($Ct_s \in Co$, $role(Ct_s) = contractor$, $ID(Ct_s) = s$) is a contractor agent in Co_a subordinated to Mg_y.

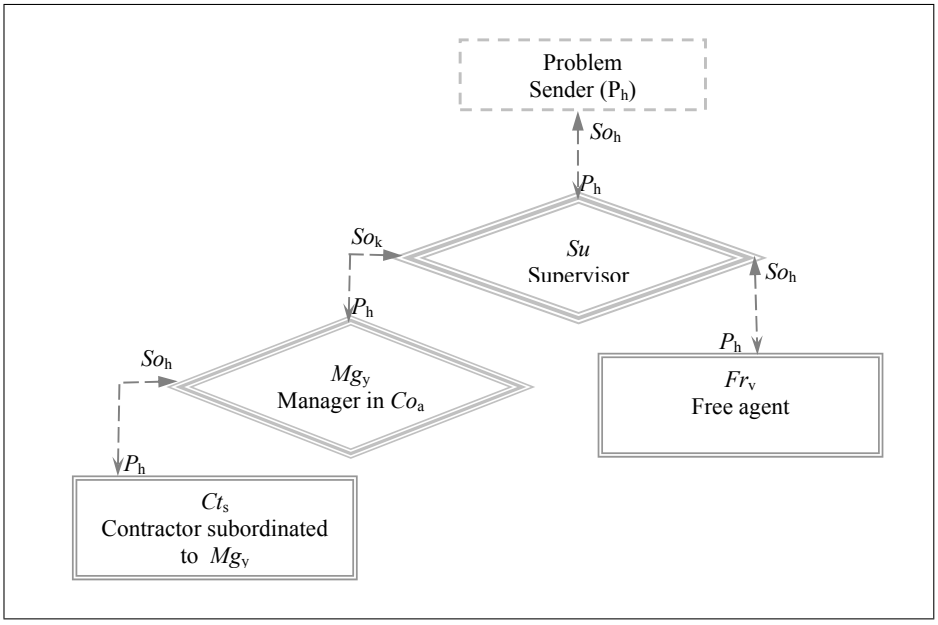


FIGURE 4. A problem's solving cycle in the ERMS multiagent system

In the *ERMS* multiagent system, each coalition and free agent can solve problems transmitted for solving by the supervisor agent. In a coalition, each transmitted problem is received by a manager from the coalition. In the case of a received problem, the manager agent will solve the problem or will transmit it for solving to a subordinated contractor agent. The manager agents from the coalitions are responsible for the problems solutions undertaking from the subordinated contractor agents and their transmission to the supervisor agent.

A problem denoted P_h transmitted for solving has the description $Desc_h$ (2).

$$Desc_h : \langle sender_h, f_h, type_h, priority_h, S_q \rangle. \tag{2}$$

In (2) there are used the following notations: $sender_h$ specifies the sender of P_h; $type_h$ represents the type (minimization or maximization) of P_h; f_h represents the objective function that must be minimized or maximized; $priority_h$ represents the priority of the problem; S_q represents the specification of the specialization to be used for the problem-solving. It is always solved a maximization problem, a received minimization problem is transformed into a maximization one.

To a specialization S_q is associated a set of parameters called pr_q (3) having an associated identifier in_q.

$$(pr_q, in_q) : \langle se_q, oi_q, om_q, cd_q, ln_q, nc_q, n_q, pc_q, pm_q \rangle. \tag{3}$$

se_q specifies the used selection operator (for example, is used the Monte Carlo linear selection method, se_q=*"Monte Carlo"*). cd_q describes the encoding of the genes from the chromosomes (for example, the genes are binary encoded, $cd_q \in \{0, 1\}$). ln_q represents the length of the chromosomes (for example, $ln_q = 20$ denotes chromosomes composed from 20 genes). Each chromosome obtained during a problem-solving process has the same length. nc_q represents the number of chromosomes from a generation (for example, $nc_q = 50$). Each generation has the same number of chromosomes. n_q represents the number of the created generations of chromosomes during a problem-solving. oi_q specifies the used crossover operator (with a single crossover point, for example). om_q specifies the used mutation operator. pc_q represents the probability of crossover. pm_q specifies the probability of mutation.

In (4) is presented a genetic problem solving specialization denoted S_q.

$$(S_q, id_q) : \langle in_q, out_q, type_q, method_q \rangle. \tag{4}$$

In (4) there are used the following notations: S_q specifies how the specialization is called (to each specialization is associated a name); id_q denotes S_q identifier; in_q denotes the identifier associated to the input parameters for S_q; out_q denotes the output parameter for the S_q specialization (the problem solution obtained after its solving); $type_q$ represents the type minimization or maximization; $method_q$ contains the description of the problem-solving method. A specialization is defined by a problem-solving method, the problem type and the input parameter values with that are initialized the parameters of the genetic algorithm.

An agent with contractor role receives problems specified in the form (2). The agent based on the problem description establishes the necessary problem-solving specialization by the form (4). The result obtained after the running of the genetic method described in the specialization represents the problem solution. The solving of a problem using a genetic algorithm does not supposes the interpretation of a syntactically described code. However, it does not decreased the problem-solving time. The manager agents can solve problems like the contractor agents. A manager agent must have specializations, which allows problems allocation for solving to contractor agents. A manager agent must distribute a problem to a contractor agent if it does not have the necessary capacity to solve the problem.

A problem-solving specialization does not represent a code difficult to use in the problems-solving. The form (4) can be adapted for problems solving using different problem solving methods based on evolutionary algorithms (evolutionary programming, genetic programming, evolutionary strategies, etc.).

3.4. **The adaptivity of the ERMS multiagent system.** In the following, we consider an *ERMS* multiagent system, denoted *ASE*. The problems transmitted for solving at a problems-solving cycle may match a *problems pattern*. A problems pattern describes different information related with the transmitted problems for solving. Formula (5) presents the general form of a problems pattern denoted Pat_i specified by the specializations denoted $S_1, S_2, \ldots, S_n$ used in the problems-solving; numbers of problems denoted $nr_1, nr_2, \ldots, nr_n$ by each specialization and the associated priorities denoted $priority_1, priority_2, \ldots, priority_n$.

$$Pat_i : \langle S_1, nr_1, priority_1; S_2, nr_2, priority_2; \ldots; S_n, nr_n, priority_n \rangle \tag{5}$$

ASE system reorganization is described by a set Rl (6) of rules detained by Su in a rule base.

$$Rl = \{Rl_1, Rl_2, \ldots, Rl_k\}. \tag{6}$$

An Rl_i ($Rl_i \in Rl$) rule has the form (7).

$$(Rl_i, Id_i) : \langle Pat_i \rangle \rightarrow \langle Inst_i \rangle. \tag{7}$$

Id_i represents the Rl_i rule identifier (each rule has as identifier a unique natural number). Pat_i represents a problems pattern. $Inst_i$ defines an instance of the *ERMS* multiagent system architecture, distribution of the agents in coalitions and allocation of roles to the agents.

ASE system can reorganize autonomously its structure at the beginning of each problems-solving cycle, if the problems pattern is known at the beginning of the problems-solving cycle, and Su has a rule in its rule base whose precondition matches the problems pattern. A reorganization determination implies a search by Su in its set $Rl = \{Rl_1, Rl_2, \ldots, Rl_k\}$ (6) of rules. The rule whose precondition matches the known problems

pattern is selected. The postcondition of the selected rule defines the new instantiation of the *ASE* system. Su, after establish the multiagent system instance, announces each agent to which coalition must migrate and what role must undertake or must operate as a free agent. Each agent will migrate autonomously into the coalition which must belong or will operate as a free agent. The determination of a new instance of the *ASE* system has a polynomial complexity.

Fundamental aspects of the *evolutionary algorithms* has been analyzed by many authors [1, 5, 13, 20]. *Evolutionary learning techniques* that are based on methods of evolutionary computation [23, 24], represent a subclass of learning techniques [12, 19]. In the following, an evolutionary learning algorithm called *Evolutionary Reorganization Learning*, which allows the construction of a rule by the form (7), denoted Rl_i is described. Each rule detained by Su will be created using the learning algorithm. *ASE* system has *no+1* member agents (a supervisor agent and *no* agents that can operate in the frame of the coalitions undertake *contractor* or *manager* roles or can operate as free agents with contractor role). Each agent, except the supervisor agents is attached a unique identifier (in the system does not exists two agents with the same identifier).

Algorithm – Evolutionary Reorganization Learning

{In: Pat_i – a problems pattern}

{Out: Rl_i – the constructed rule}

Step 1. The creation of the initial generation of chromosomes.

$t = 1$.

@Initialize the chromosome population $P(t)$.

Step 2. Search for the best-fitted instantiation of the ASE architecture.

While $(t \leq gen)$ *do*

@Create a copy Cr_y of the best-fitted chromosome from $P(t)$.

@Select chromosomes from $P(t)$ using a selection method. Choose chromosomes from $P(t)$ to enter in the mating pool *mp*. Let P_1 be the selected chromosomes.

@Using the *rc* crossover operator applied with the probability *pr* recombine the chromosomes in *mp* forming the population P_2.

@Mutate the chromosomes in P_2 using the operator *mc*, with the probability *pm*.

@Replace in P_2 the worst-fitted chromosome with Cr_y. Let P_2 be the obtained population of chromosomes.

$t = t + 1$.

$P(t) = P_2$ (the new generation of chromosomes is constructed).

EndWhile.

Step 4. Construction of the rule that describes the ASE system's organization.

@Select the best-fitted chromosome Cr_y from $P(t)$.

@Generate a unique rule identifier denoted Id_i.

@Based on the selected chromosome Cr_y construct the Rl_i rule.

$(Rl_i, Id_i) : \langle Pat_i \rangle \rightarrow \langle Inst_i \rangle$.

EndEvolutionaryReorganizationLearning.

gen, n, pm, pr, no are parameters of the algorithm. gen represents the number of generations of chromosomes constructed during a learning process. Let us denote with $P(t)$ the t'th generation of chromosomes. In $P(1)$ each chromosome is generated by random. Each generation has the same number n of chromosomes. Each chromosome has no genes. A chromosome $Cr_w = [G_1, G_2, \ldots, G_{no}]$ specifies an instance of the ASE architecture.

Except Su, to each agent Ag_j ($ID(Ag_j) = j$) a gene $G_j = (v_j[1], t_j[2], w_j[3])$ corresponds in each chromosome. The first layer parameter value v_j from the gene G_j specifies the affiliation of Ag_j to the coalition with the identifier v_j. Each coalition is identified with a natural number v_k, where $v_k \in [1, no]$. All the agents with the same value v_k associated to their corresponding gene are members of the same coalition, the coalition with the identifier v_k. A gene that has a value different from all other genes' value means that the corresponding agent to the gene is a free agent. The second layer value t_j ($t_j \in T$, where $T = \{'m', 'c'\}$) from the gene G_j specifies the role of the Ag_j agent. If $t_j =' m'$ then $role(Ag_j) = manager$. If $t_j =' c'$ then $role(Ag_j) = contractor$. If $t_j =' c'$ and Ag_j is not a free agent, then w_j value specifies the identifier of the manager agent from the same coalition to who will be subordinated Ag_j. A free agent, denoted Ag_j, has contractor role ($role(Ag_j) = contractor$, $t_j =' c'$) and $w_j = 0$ (does not subordinated to any manager agent). An offspring generation is created using specific selection, crossover and mutation operators.

By mutation, denoted mc, are generated new chromosomes by small variations of the genes' values in the chromosomes. We define mc as an application by the form (8). Cr specifies the chromosome space.

$$mc : Cr \to Cr. \tag{8}$$

The mutation is applied on each layer of each gene $G_j = (v_j[1], t_j[2], w_j[3])$ from each chromosome with the probability pm, $pm = \{pm_v, pm_t\}$, where pm_v, pm_t, are the corresponding probabilities to the mutation of v_j and t_j. pm_t represents the probability to be created a contractor agent that will operate in the frame of a coalition. $1 - pm_t$ represents the probability to be created a manager agent, where $pm_t > 1 - pm_t$. By mutation, the v_j value may increase or decrease, the new value of v_j must be between 1 and no.

If v_j specifies that the agent is a free agent then there is not applied the mutation to the rest of the layers. t_j is set to $'c'$ ($t_j =' c'$ – the agent will operate as having contractor role) and w_j is set to 0 ($w_j = 0$ – the agent is not subordinated to any manager agent).

If v_j does not specify a free agent then there is applied the mutation to the second layer t_j that could change its value, the new value of t_j must be in the set $T = \{'m', 'c'\}$, where $t_j \in T$. If $t_j = 'm'$ then w_j value is set to 0. If $t_j = 'c'$ then is randomly generated one of the identifiers of manager agents from the same coalition and the identifier is set to w_j.

In case of each coalition is verified if all the managers of the coalition have subordinated contractor agents. The role of a manager that does not have subordinated contractor agents is changed into contractor and the agent is subordinated randomly to a manager agent from the same coalition that have at least on other contractor.

The crossover rc is used to create new chromosomes by combining the genetic information of parent chromosomes. We define rc as an application by the form (9).

$$rc : Cr^2 \to Cr^2. \tag{9}$$

rc realizes a (2, 2) transformation, two parents are combined to obtain two offspring. rc is applied with the probability $\{pr; \{pr_v, pr_t, pr_w\}\}$; where pr represents the probability of the chromosomes to be selected for inclusion in the mating pool, pr_v, pr_t, pr_w represent probabilities to crossover the layers 1, 2 and 3 in the chromosomes. During the crossover of two chromosomes there are recombined the values of the genes from the same layer: [1]

for the v_i values (that specify the agents membership to coalitions); [2] for the t_i values (that specify the agents roles) and [3] for the w_i values (if an agent is contractor then w_i specify its manager agent to who the contractor agent is subordinated).

Each chromosome of a population is evaluated using a real-valued fitness function Fit by the form (10), where $\forall Cr_b \in Cr, Fit(Cr_b) \geq 0$, which counts how efficiently the multiagent system with the structure specified by the chromosome Cr_b can solve the undertaken problems at a problems-solving cycle.

$$Fit : Cr \to R^+. \tag{10}$$

A chromosome's fitness is evaluated simulating the problems-solving that matches the problems pattern. The efficiency of the problems-solving at a problems solving cycle, has as meaning the problems solving time (all the undertaken problems at the beginning of the problems-solving cycle are solved). Let Cr_h ($Cr_h \in Cr$) and Cr_k ($Cr_k \in Cr$) two chromosomes. $Fit(Cr_h) > Fit(Cr_k)$ means that Cr_h is best-fitted then Cr_k (it is solved a maximization problem, an initially transmitted minimization problem is transformed to a maximization one). The best-fitted chromosome Cr_y from the least generation $P(gen)$ specifies the postcondition of the constructed Rl_i rule.

For each gene $G_j = (v_j, t_j, w_j)$ from a chromosome Cr_w must be satisfied the following validity restrictions:

A). Let $v_j = nr_j$, then $nr_j \in [1, no]$ and must have the values 1,..., $nr_j - 1$ in the first layer of the Cr_w chromosome;

B). Let $t_j = tr_j$, then $tr_j \in \{'m', 'c'\}$;

C). Let $w_j = wr_j$. If $tr_j =' c'$ and nr_j does not have a unique value (it does not specify a free agent), then wr_j must specify the identifier of a manager agent from the same coalition (coalition with the identifier nr_j) as the agent with the identifier j. If $tr_j =' m'$ then $wr_j = 0$. If $tr_j =' c'$ and nr_j has a unique value (specify a free agent), then $wr_j = 0$.

D). Let $t_j = tr_j$. If $tr_j =' m'$ then it must have at least one subordinated contractor agent.

In the case of each invalid chromosome obtained during a learning process (are not satisfied all the restrictions (A), (B), (C) and (D)) a transformation Trf (11) is applied that corrects the invalid genes values.

$$Trf : Cr \to Cr. \tag{11}$$

The validity of each randomly generated chromosome from the initial generation is verified. In case of an invalid chromosome is applied the transformation Trf (11). During a learning process, the validity of each newly obtained chromosome by applying the mutation mc or crossover rc is verified. A chromosome is considered valid if it specifies a correct multiagent system structure.

A survival mechanism based on the fitness measure Fit is applied to select the chromosomes of the new generation from the offspring and parent generations. In the *Evolutionary Reorganization Learning* algorithm two types of selections are used. The selection for crossover operator is used to decide which members of the recent generation $P(t)$ will be used as parents of the new generation $P(t+1)$. The selection for the replacement operator is used to obtain, which chromosomes from $P(t)$ and their offspring will effectively enter in the new generation $P(t+1)$. The best-fitted chromosome Cr_z from the last generation $P(gen)$ represents the solution (a valid multiagent system structure).

3.5. **Correctness in operation of the ERMS system.** During its operation for the reorganization, the *ERMS* system uses a set of learned rules. All the rules are correct, their postcondition specify correct reorganization of the *ERMS* system. The correctness of the *Evolutionary Reorganization Learning* algorithm can be theoretically demonstrated.

Each chromosome obtained during a learning process, represents a valid instantiation of the *ERMS* multiagent system architecture. Only the mc (8) mutation and rc (9) crossover operators can modify the genes values from the chromosomes. A transformation Trf (11) is applied to each obtained invalid chromosome. However, at the end of each learning process a correct multiagent system instantiation is obtained. In every new generation, the best-fitted chromosome from the previous generation is transferred, which guarantees that the best multiagent system instantiation is not lost during the learning process.

The *ERMS* system can reorganize its coalitions at the beginning of each problems-solving cycle, if there is known at the beginning of the problems-solving cycle the problems pattern and there is a rule whose precondition match that pattern. In case of selection of a rule at the beginning of a problems pattern, the postcondition of that rule specifies the necessary correct reorganization of the system. If it is not matched any rule at the beginning of a problems solving cycle then the system remains with the previous structure.

For the validation of the *ERMS* multiagent system, there have been realized experimental simulations for the learning processes and the testing of the system's operation with some learned rules by sending to the system problems at the beginning of problem-solving cycles with known and unknown pattern. The simulations purposes were to establish the measure in that the distribution of agents in coalitions and allocation of roles to the agents, influences the efficiency of the problems-solving. During the simulations, they used agents endowed between 1 and 16 problem-solving specializations (*a specialization* is a problem-solving method where some parameters could be initialized during a problem-solving). Requirements for an agent to solve a problem is to have the necessary role (role that allows problems-solving), the necessary problem-solving specialization and resources.

We have simulated learning processes were generated at least $gen = 57$ generations of chromosomes. The problems have been solved in problems-solving cycles. At each problems-solving cycle the problems have been transmitted for solving at the beginning of the cycle. In the case of each problems pattern, there have been realized 50 simulations for the construction of the rule based on the problems pattern. The necessity to run multiple times learning processes on the same data was because it has been used an evolutionary learning technique (as in any heuristic search in the problem space running the same evolutionary algorithm on the same data with the same parameters many converge to different solutions). The most appropriate parameters values used during the simulations were: mutation probability $pm = \{pm_v, pm_t\} = \{0.0093, 0.8\}$; crossover probability $\{pr; \{pr_v, pr_t, pr_w\}\} = \{0.09; \{0.2, 0.3, 0.33\}\}$. There were made experiments for the following conditions: numbers of problems transmitted for solving (denoted $prno$): 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140 and 150; numbers of agents (denoted no): 10, 20, 25, 30, 35 and 40; chromosomes numbers (denoted n) in the generations (during a learning process each generation having the same number of chromosomes): 20, 25, 30 and 35.

Figures 5-8 present the changing of the average problem-solving time (expressed in ms milliseconds), from generation to generation (each generation where composed from 30 chromosomes), for the first 29 generations of chromosomes using 25 agents, and 57 generations of chromosomes using 20 agents for the construction of 3 rules.

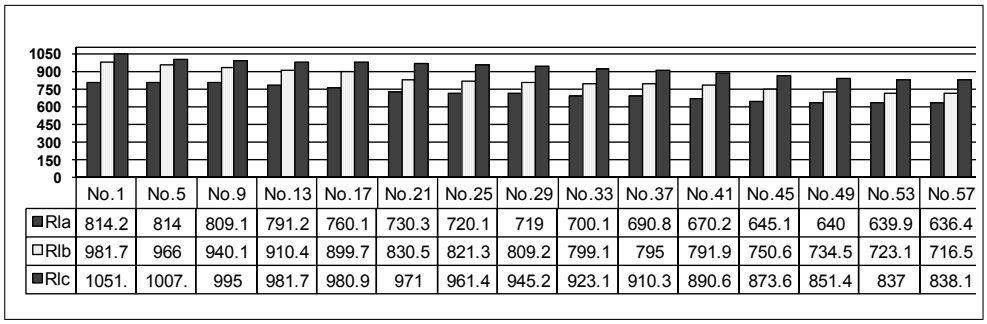
Figure 5 presents the time decrease using $no = 20$ agents, for the construction of 3 rules, based on 3 problems patterns, each of them composed from $prno = 70$ problems. The using of the rules has an improvement (as time decrease) of $\sim 28\%$ for Rl_a, $\sim 37\%$ for Rl_b, and $\sim 25\%$ for Rl_c. By using $no = 20$ agents that solve $prno = 70$ problems arranged in patterns the average improvement obtained during the simulations was approximatively by 29%.

Figure 6 presents the time decrease using $no = 20$ agents, for the construction of 3 rules, based on 3 problems patterns, each of them composed from $prno = 140$ problems. The using of the rules has an improvement of $\sim 20\%$ for Rl_d, $\sim 23\%$ for Rl_e, and $\sim 20\%$ for Rl_f. By using $no = 20$ agents that solve $prno = 140$ problems arranged in patterns the average improvement obtained during the simulations was approximatively by 22%.

Figure 7 presents the time decrease using $no = 25$ agents, for the construction of 3 rules, based on 3 problems patterns, each of them composed from $prno = 70$ problems. The using of the rules has an improvement of $\sim 30\%$ for Rl_g, $\sim 40\%$ for Rl_h, $\sim 20\%$ for Rl_i. By using $no = 25$ agents that solve $prno = 70$ problems arranged in patterns the average improvement obtained during the simulations was approximatively by 31%.

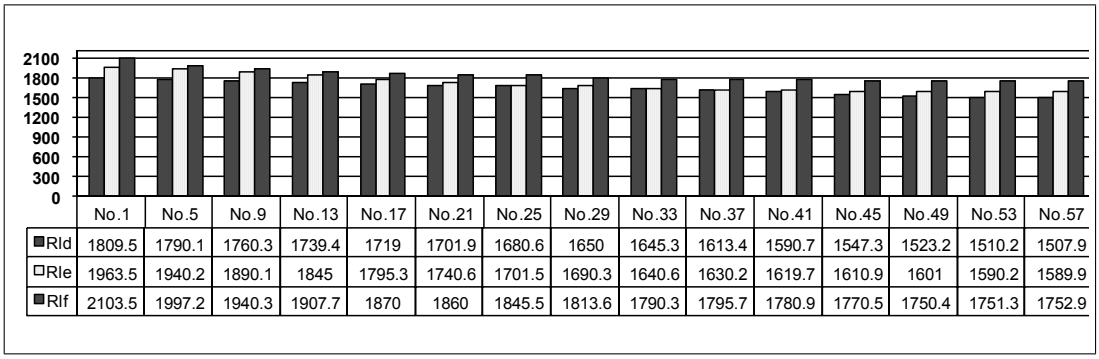
Figure 8 presents the time decrease using $no = 25$ agents, for the construction of 3 rules, based on 3 problems patterns, each of them composed from $prno = 140$ problems. The using of rules has an improvement of $\sim 21\%$ for Rl_j, $\sim 33\%$ for Rl_k, $\sim 19\%$ for Rl_l. By using $no = 25$ agents that solve $prno = 140$ problems arranged in patterns the average improvement obtained during the simulations was approximatively by 24%.

During a problems-solving at a problems solving cycle there are always some costs associated with the computation time. There is a cost for checking if the problems sent for solving respects a known pattern (there is a rule in the rule base that has as precondition the problems pattern which have been learned during a learning process). If the problems do not respect any pattern then there is a cost without any improvement. If at the beginning of a problems-solving cycle is verified a rule, then there is a cost for the checking of the reorganization.



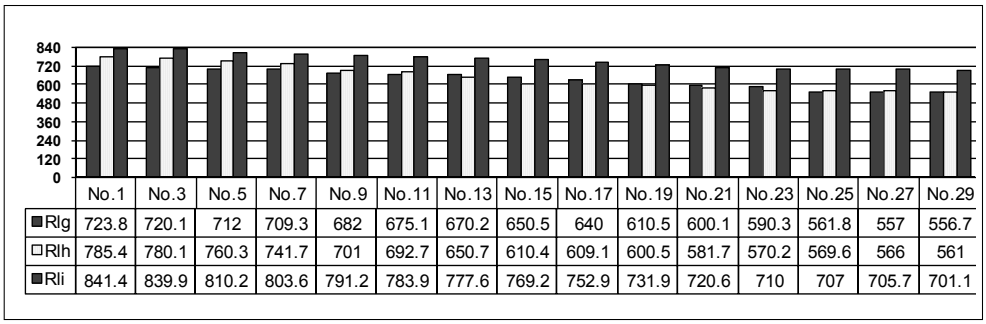
	No.1	No.5	No.9	No.13	No.17	No.21	No.25	No.29	No.33	No.37	No.41	No.45	No.49	No.53	No.57
Rla	814.2	814	809.1	791.2	760.1	730.3	720.1	719	700.1	690.8	670.2	645.1	640	639.9	636.4
Rlb	981.7	966	940.1	910.4	899.7	830.5	821.3	809.2	799.1	795	791.9	750.6	734.5	723.1	716.5
Rlc	1051.	1007.	995	981.7	980.9	971	961.4	945.2	923.1	910.3	890.6	873.6	851.4	837	838.1

FIGURE 5. Construction of the rules Rl_a, Rl_b, Rl_c, generations 1-57, $no = 20$, $prno = 70$, $n = 30$



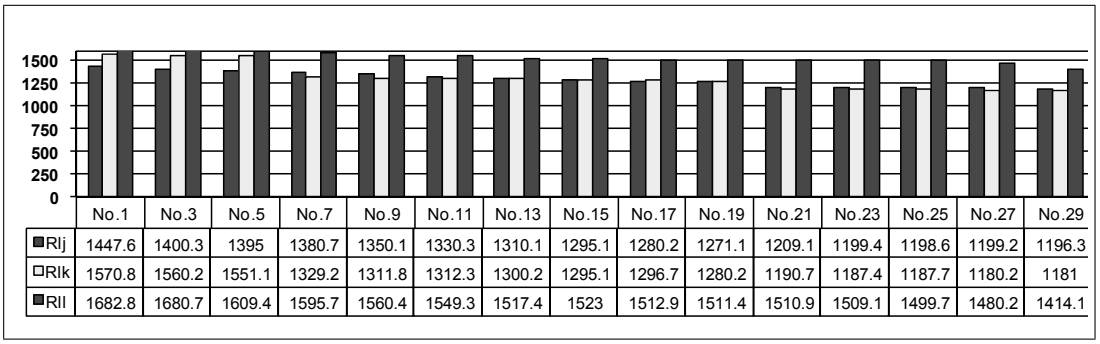
	No.1	No.5	No.9	No.13	No.17	No.21	No.25	No.29	No.33	No.37	No.41	No.45	No.49	No.53	No.57
Rld	1809.5	1790.1	1760.3	1739.4	1719	1701.9	1680.6	1650	1645.3	1613.4	1590.7	1547.3	1523.2	1510.2	1507.9
Rle	1963.5	1940.2	1890.1	1845	1795.3	1740.6	1701.5	1690.3	1640.6	1630.2	1619.7	1610.9	1601	1590.2	1589.9
Rlf	2103.5	1997.2	1940.3	1907.7	1870	1860	1845.5	1813.6	1790.3	1795.7	1780.9	1770.5	1750.4	1751.3	1752.9

FIGURE 6. Construction of the rules Rl_d, Rl_e, Rl_f, generations 1-57, $no = 20$, $prno = 140$, $n = 30$



	No.1	No.3	No.5	No.7	No.9	No.11	No.13	No.15	No.17	No.19	No.21	No.23	No.25	No.27	No.29
Rlg	723.8	720.1	712	709.3	682	675.1	670.2	650.5	640	610.5	600.1	590.3	561.8	557	556.7
Rlh	785.4	780.1	760.3	741.7	701	692.7	650.7	610.4	609.1	600.5	581.7	570.2	569.6	566	561
Rli	841.4	839.9	810.2	803.6	791.2	783.9	777.6	769.2	752.9	731.9	720.6	710	707	705.7	701.1

FIGURE 7. Construction of the rules Rl_g, Rl_h, Rl_i, generations 1-29, $no = 25$, $prno = 70$, $n = 30$



	No.1	No.3	No.5	No.7	No.9	No.11	No.13	No.15	No.17	No.19	No.21	No.23	No.25	No.27	No.29
Rlj	1447.6	1400.3	1395	1380.7	1350.1	1330.3	1310.1	1295.1	1280.2	1271.1	1209.1	1199.4	1198.6	1199.2	1196.3
Rlk	1570.8	1560.2	1551.1	1329.2	1311.8	1312.3	1300.2	1295.1	1296.7	1280.2	1190.7	1187.4	1187.7	1180.2	1181
Rll	1682.8	1680.7	1609.4	1595.7	1560.4	1549.3	1517.4	1523	1512.9	1511.4	1510.9	1509.1	1499.7	1480.2	1414.1

FIGURE 8. Construction of the rules Rl_j, Rl_k, Rl_l, generations 1-29, $no = 25$, $prno = 140$, $n = 30$

The testing of the system was realized using 25 agents (rules constructed for a system composed from 25 agents) by sending to it 60% known and 40% unknown problems patterns. The results of simulations show an average improvement of about 19.14% decrease in the solving time when the system reorganizes its structure versus the system does not (the average decrease of the solving time was 23.24%, with 4.1% reorganization costs). The optimal number of parameters related to the number of agents was 25. The most appropriate number of chromosomes in the simulation settings was 30. A smaller number of chromosomes have increased the number of necessary generations, increasing the total learning time. A larger number of chromosomes have not increased the convergence time, and after 29 and 57 generations it does not have significant improvement.

4. **Considerations Related with the Intelligence – ERMS System Intelligence.** In numerous studies are given definitions and hypotheses related with different types of *biological intelligence*, like *human intelligence* [14, 36, 37, 40], *animal intelligence* [2] and *plant intelligence* [41, 42]. Based on some recent studies [2, 14, 36, 37, 40, 41, 42], we can conclude that the biological intelligence in general, human intelligence particularly could not be defined unequally because it is not completely understandable. Some of the difficulties in its understanding consist in aspects like the biological complexity and variety. Human intelligence cannot be defined but may be measured from different points of view. The *psychometric* approach is by far the most widely used in practical settings for measurement of the human intelligence [36].

The humans have attained the actual level of intelligence during a very long evolution. As long as the intelligence of the humans and computational systems are completely different, they cannot be directly compared; therefore, the intelligence must be evaluated

based on different considerations. In many researches the biological intelligence represents a source of inspiration for the development of intelligent systems.

We consider the "intelligence" a property (or a set of properties) of a system (usually an agent) that emerges in some improvements in the problems-solving, many times consisting in increased autonomy, precision and flexibility of the problems-solving. Commonly the computational systems intelligence is considered just based on some properties, like capacity to: learn, adapt, evolve etc. Such considerations as they are only specified could not be considered rigorous for a system's intelligence assessment.

An agent must have only the necessary intelligence. Sometimes, unnecessary intelligence, usually in the case of solving very simple problems, may decrease the efficiency of the problems-solving. Usually, an intelligent agent makes some computations during the problems-solving that improves the efficiency and flexibility of difficult problems-solving. But such computations, in case of simple problems, became unnecessary and time consuming. The establishment of the necessary intelligence for an agent is an important aspect that must be analyzed at its development cycle.

For illustrative purposes, we consider as example the assessment of a system's intelligence based on the capacity to learn knowledge that allows new problems-solving. There are different aspects that must be taken into consideration at the evaluation of the intelligence:

- The learning time. The system can learn on-line or off-line;
- The quantity of learned knowledge. The system can learn more or less knowledge;
- The accuracy/quality of learned knowledge. The learned knowledge could be more or less accurate;
- The usefulness of the learned knowledge. The learned knowledge could be more or less useful. It could happen that the system will not use the learned knowledge;
- The consumption of computational resources during the learning. The learning could require more or less computational resources (could be more or less time consuming);
- The consumption of computational resources during the problems-solving. The use of the learned knowledge may require more or less computational resources. We may consider for example an extremely intelligent system that uses numerous resources for solving of a simple problem making useless computations.

Our consideration is that even if it is not possible to give a unified definition for the intelligence of a system in general, for the evaluation of a system's intelligence we consider necessary the following aspects to be established:

1. existence of one or more properties based on which the system could be considered intelligent. The intelligence is manifested by evolution (the system evolve autonomously), for example.
2. elaboration of a metric that allows the measurement of the intelligence (allows a quantitative evaluation of quality). The metric must indicate the existence of the intelligence. Sometimes it is better to indicate a degree of intelligence, like: *no intelligence*, *limited intelligence*, *normal intelligence*, *increased intelligence*, *extreme intelligence*.

A metric (general evaluator) that allows the measurement of the intelligence of a system must take into consideration aspects, related with the:

- specific and type of the system. For example, the system is a: *static software agent*; *mobile software agent*; *mobile robotic agent*; *static robotic agent*.
- specific, number and complexity of the problems that must be solved by the system. Usually a difficult problem solving requires more intelligence. Different types of problems could require different type of intelligence for their solving.

- the autonomy of the system in attaining the intelligence;
- the necessary cost and duration for attaining of intelligence;
- the measurable improvements that emerge based on the detained intelligence;
- measurable time in that the use of intelligence has as result improvements.

5. **Conclusions.** The main purpose of our research was a study of the intelligence that emerges in an adaptive system composed from relatively simple cooperating agents. We have proposed a multiagent system called *ERMS* capable of solving relatively large numbers of problems using genetic algorithms. Many real life problems-solving require the use of genetic algorithms or combinations of them with other methods [1, 5, 6, 13, 20]. Moreover, in the papers [21, 22] is described a novel class of mobile software agents called *ICMAE* (*Intelligent Cooperative Mobile Agents with Evolutionary Problem Solving Specialization*), that uses problem-solving specializations based on evolutionary problem-solving techniques.

The properties of the *ERMS* system that could be associated with the intelligence consist in the adaptability of the system, manifested by its capacity to autonomously reorganize its structure. The system is able to autonomously learn, using an evolutionary learning technique, how to adapt its structure. As a metric for the evaluation of the systems' intelligence we have considered the problems-solving time resulted from the systems' reorganization based on the specific/pattern of the problems sent for solving.

ERMS multiagent system represents an extension of the *CCER* multiagent system [23] developed during our previous researches. The development presented in this paper and the previously developed *CCER* multiagent system proves that computational system, composed from interacting components (agents), that use methods based on evolutionary computation for learning the required adaptability, exhibits at the level of the system an emergent intelligent adaptive behavior.

Usually, the intelligence of a system gave advantages in some situations, but it could have in some conditions disadvantages as well. We have established some general principles that must guide the development of intelligent systems and estimation of their intelligence. Usually a system's intelligence increases its complexity which the system must be able to handle autonomously inside. The complexity of a system must be hidden from the external parties, to the humans and agents that request problems-solving from the system.

Acknowledgments. The research of Barna Laszlo Iantovics was supported by the project "Transnational Network for Integrated Management of Postdoctoral Research in Communicating Sciences. Institutional building (postdoctoral school) and fellowships program (CommScie)" – POSDRU/89/1.5/S/63663, financed under the Sectoral Operational Programme Human Resources Development 2007-2013.

The research of Constantin-Bala Zamfirescu was supported by the research project between Romania and Slovakia, entitled *Hybrid Medical Complex Systems – ComplexMediSys*, 2011-2012. The partner institutions involved in the project are Petru Maior University, Tg. Mures, Romania and the Institute of Informatics of the Slovak Academy of Sciences, Bratislava, Slovakia.

REFERENCES

[1] T. Back, D. B. Fogel and Z. Michalevitz, *Handbook of Evolutionary Computation*, Oxford University Press, Oxford, 1997.

[2] S. Coren, *The Intelligence of Dogs*, Bantam Books, 1995.

[3] D. D. Corkill, *A Framework for Organizational Self-Design in Distributed Problem Solving Networks*, University of Massachusetts, 1983.

[4] K. S. Decker, K. Sycara and M. Williamson, Cloning for intelligent adaptive information agents, *Lecture Notes in Computer Science*, vol.1286, pp.63-75, 1997.
[5] D. Dumitrescu, B. Lazzerini, L. Jain and A. Dumitrescu, *Evolutionary Computing*, CRC Press, Boca Raton, 2000.
[6] L. Duta, F. G. Filip, J. M. Henrioud and C. Popescu, Disassembly line scheduling with genetic algorithms, *International Journal of Computers, Communications and Control*, vol.3, no.3, pp.270-280, 2008.
[7] I. Dzitac and B. E. Barbat, Artificial Intelligence + Distributed Systems = Agents, *International Journal of Computers Communications and Control*, vol.4, no.1, pp.17-26, 2009.
[8] L. Eder, *Managing Healthcare Information Systems with Web-Enabled Technologies*, IGI Global, 2000.
[9] S. Fatima, *An Adaptive Organizational Policy for Multi-Agent Systems*, Ph.D. Thesis, University of Hyderabad, 1999.
[10] S. Fatima and G. Uma, An adaptive organizational policy for multi agent systems, *Proc. of the 3rd International Conference on Multi-Agent Systems*, Paris, France, pp.120-127, 1998.
[11] S. Fatima and M. Wooldridge, Adaptive task and resource allocation in multi-agent systems, *Proc. of the 5th International Conference on Autonomous Agents*, Canada, pp.537-544, 2001.
[12] J. Ferber, *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*, Addison Wesley, 1999.
[13] D. B. Fogel, *Evolutionary Computation, Toward a New Philosophy of Machine Intelligence*, IEEE Press, New York, 2000.
[14] L. S. Gottfredson, Foreword to intelligence and social policy, *Intelligence*, vol.24, no.1, pp.1-12, 1997.
[15] F. Guichard and J. Ayel, Logical reorganization of DAI systems, *Lecture Notes in Artificial Intelligence*, vol.890, pp.118-128, 1995.
[16] B. Hayes-Roth, An architecture for adaptive intelligent systems, *Artificial Intelligence*, vol.72, no.1-2, pp.329-365, 1995.
[17] B. Hayes-Roth, M. Hewett, R. Washington, R. Hewett and A. Seiver, Distributing intelligence within an individual, *Technical Report*, Distributed Artificial Intelligence, CA, USA, 1989.
[18] J. Huang, N. R. Jennings and J. Fox, An agent-based approach to health care management, *International Journal of Applied Artificial Intelligence*, vol.9, no.4, pp.401-420, 1995.
[19] B. Iantovics, C. Chira and D. Dumitrescu, *Principles of the Intelligent Agents*, Casa Cartii de Stiinta Press, Cluj-Napoca, 2007.
[20] B. Iantovics, *New Paradigms of the Evolutionary Computation, Association with the Intelligent Agents*, Ph.D. Thesis, Babes-Bolyai University, 2004.
[21] B. Iantovics, *Evolutionary Mobile Agents, International Conference European Integration between Tradition and Modernity*, Petru Maior University Press, 2005.
[22] B. Iantovics, Problem solving using evolutionary mobile agents, *Proc. of the 9th National Conference of the Romanian Mathematical Society*, Lugoj, Timisoara, pp.408-420, 2005.
[23] B. Iantovics, Evolutionary reorganization of the centralized multiagent systems, *Proc. of the 2nd International Conference on Economics, Law and Management*, Tg. Mures, pp.139-153, 2006.
[24] B. Iantovics, *Evolutionary Learning Techniques*, Scientific Bulletin of the Petru Maior University, Tg. Mures, XV-XVI, 2003.
[25] B. Iantovics, Agent-based medical diagnosis systems, *Computing and Informatics*, vol.27, no.4, pp.593-625, 2008.
[26] B. Iantovics, Cooperative medical diagnoses elaboration by physicians and artificial agents, in *Understanding Complex Systems*, 2009.
[27] I. F. Imam, Intelligent adaptive agents, *Technical Report WS-96-04*, 1996.
[28] T. Ishida, L. Gasser and M. Yokoo, Organization self design of production systems, *IEEE Transactions on Knowledge and Data Engineering*, vol.4, no.2, pp.123-134, 1992.
[29] S. Kirn, Ubiquitous healthcare: The OnkoNet mobile agents architecture, *Proc. of the Conference on Objects, Components, Architectures, Services, and Applications for a Networked World , LNCS*, vol.2591, pp.265-277, 2003.
[30] G. Lanzola, S. Falasconi and M. Stefanelli, Cooperative software agents for patient management, *Proc. of the AIME Conference*, pp.173-184, 1995.
[31] G. Lanzola, S. Falasconi and M. Stefanelli, Cooperating agents implementing distributed patient management, *Proc. of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Berlin, Springer-Verlag, vol.1038, pp.218-232, 1996.

[32] M. Maruteri and V. Bacarea, Comparing groups for statistical differences: How to choose the right statistical test, *Biochemia Medica*, vol.20, no.1, pp.15-32, 2010.

[33] M. Maruteri, B. Crainicu and A. Schiopu, RoBioCluster – An open source platform for HPC (High Performance Computing)/Linux Clusters in the biomedical field, *Proc. of the European Federation for Medical Informatics Special Topic Conference & ROMEDINF*, pp.174-177, 2006.

[34] M. Maruteri, B. Crainicu and A. Schiopu, ROSLIMS Live CD – All-in-one cross-platform solution for running biomedical software, *Proc. of the European Federation for Medical Informatics Special Topic Conference & ROMEDINF*, pp.178-181, 2006.

[35] S. Negulescu, C. Zamfirescu and B. Barbat, User-driven heuristics for nondeterministic problems, *Studies in Informatics and Control*, vol.15, no.3, pp.289-296, 2006.

[36] U. Neisser, G. Boodoo, T. J. Bouchard, A. W. Boykin, N. Brody, S. J. Ceci, D. F. Halpern, J. C. Loehlin et al., Intelligence: Knowns and unknowns, *American Psychologist*, vol.51, no.77, 1996.

[37] U. Neisser, G. Boodoo, T. J. Bouchard Jr, A. W. Boykin, N. Brody, S. J. Ceci, D. F. Halpern, J. C. Loehlin, R. Perloff and R. J. Sternberg, Intelligence: Knowns and unknowns, *Annual Progress in Child Psychiatry and Child Development*, 1997.

[38] F. B. Pereira and E. Costa, The influence of learning in the behavior of information retrieval adaptive agents, *Proc. of ACM Symposium on Applied Computing*, New York, pp.452-457, 2000.

[39] F. B. Pereira and E. Costa, How adaptive agents learn to deal with incomplete queries in distributed information environments, *Proc. of the Congress on Evolutionary Computation, Technologies*, vol.1, no.14, pp.1329-1336, 2000.

[40] R. Perloff, R. J. Sternberg and S. Urbina, Intelligence: Knowns and unknowns, *American Psychologist*, vol.51, no.2, pp.77-101, 1996.

[41] A. Trewavas, Green plants as intelligent organisms, *Trends in Plant Science*, vol.10, no.9, pp.413-419, 2005.

[42] A. Trewavas, Mindless mastery, *Nature*, vol.415, no.6874, pp.841, 2002.

[43] A. Ulieru and M. Grabelkovsky, Telehealth approach for glaucoma progression monitoring, *Information Theories and Applications*, vol.10, pp.326-329, 2005.

[44] R. Unland, A holonic multi-agent system for robust, flexible, and reliable medical diagnosis, *Lecture Notes in Artificial Intelligence*, vol.2889, no.2003, pp.1017-1030, 2003.

[45] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press Cambridge, Massachusetts London, England, 2000.

[46] D. Yergens, J. Hiner, J. Denzinger and T. Noseworthy, Multiagent simulation system for rapidly developing infectious disease models in developing countries, *Proc. of the 2nd International Workshop on Multi-Agent Systems for Medicine and Computational Biology*, Hakodate, Japan, pp.104-116, 2006.

[47] C. B. Zamfirescu and F. G. Filip, Swarming models for facilitating collaborative decisions, *International Journal of Computers Communications and Control*, vol.5, no.1, pp.125-137, 2010.