STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVIII**, Number 3, 2013

AN AGENT BASED APPROACH FOR PARALLEL CONSTRAINT VERIFICATION

RADU D. GĂCEANU[(1)], HORIA F. POP[*,(1)], AND SERGIU A. SOTOC[(1)]

Abstract. The medical procedure result in case of cardiac arrest is highly dependant on several factors including the quality of chest compressions, early defibrillation and advanced life support. Timing and precise procedure compliance are critical in such situations. This paper presents an agent-based approach for detecting erroneous procedure follow-ups. The Erlang programming language is chosen for implementation because of its reputation in outstanding concurrency support thus being a perfect environment for agent-based solutions. We provide experiments on a synthetic dataset as well as on a dataset provided by the Romanian Resuscitation Council.

1. Introduction

In case of cardiac arrest, the result of the medical personnel performance is greatly influenced by the time and accuracy of applying predefined procedures. Even though investments in the infrastructure are continuously being done, the survival rate in case of cardiac arrest remains low [6].

In order to improve this situation the Romanian Resuscitation Council (CNRR) [1] promotes high quality education in resuscitation (basic and advanced life support) acting in Romania at national level since 1998.

Our aim is to help the Romanian Resuscitation Council to detect situations where the standard procedures have been infringed so that they can direct their efforts in such weaker areas. We propose an agent-based approach for detecting breaches in the standard protocols. Since the search space may be large (nation-wide registries), providing scalable solutions is desired. The

Received by the editors: April 15, 2013.

2010 *Mathematics Subject Classification.* 68T42.

1998 *CR Categories and Descriptors.* I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Multiagent systems*.

Key words and phrases. Intelligent Agents, Parallel Constraint Verification, Erlang.

This paper has been presented at the International Conference KEPT2013: Knowledge Engineering Principles and Techniques, organized by Babeş-Bolyai University, Cluj-Napoca, July 5-7 2013.

* Corresponding author.

Erlang programming language advocates the benefits of concurrent programming so it seems a good choice for our agent-based approach.

Approaches that are similar to ours are presented in [7, 8, 16]. Authors from [16] present a review of distributed constraint satisfaction problems and some extensions to the considered models. They propose the use of multi-agent systems in distributed constraint satisfaction problems. In their view an agent is responsible for instantiating several variables and in order to reach a solution agents have to negotiate over the enforced rules. Authors from [7] employ a two dimensional grid for the agents environment in order to solve CSP problems. A solution to the problem is given by the positions of the agents on the grid provided that no constraint is unsatisfied. In [8] propose a methodology called constrained partition and coordinated reaction for distributed constraint satisfaction is presented. The introduced approach partitions the set of constraints and associates an agent to each subset of constraints. The process starts from an initial instantiation of variables and, according to the authors, the solution emerges through incremental local revisions of the variable instantiations. The methodology was applied on the job shop scheduling problem and experiments show better results compared to other methods.

Our approach considers an agent for each variable and instead of finding an assignment for all variables such that all constraints hold (like in a CSP), our aim is to find all variable assignments for which some constraint does not hold. The proposed problem has practical reasons and is clearly motivated in Section 2.

The paper is organized as follows: Section 2 states the motivation of our approach, Section 3 presents the theoretical background. The proposed approach is presented in Section 4 and is followed by Section 5 where we show our experimental evaluation. In Section 6 we present a comparison with related work. Finally, conclusions and ideas for future work are presented in Section 7.

2. Motivation

In cardiac arrest cases, the outcome of the intervention on the patient is greatly influenced by the time and precise compliance to well defined procedures. Technological growth has facilitated the decrease of rescue teams to patient arrival times. Nevertheless survival rates remain unsatisfactory [6]. The focus is now on improving the performance of rescue team procedures as this is thought to be the main cause of low survival rates [9].

In this regard, the Romanian Resuscitation Council (CNRR) [1] aims to promote high quality education in resuscitation (basic and advanced life support). Since 2004, CNRR is the unique partner of the European Resuscitation

Council in Romania. Tasks performed by CNRR since 1998 until today include: promoting the quality of education in resuscitation; translation of the resuscitation guidelines in 2000, 2005 and 2010; formations in resuscitation and trauma at national level; implication in the REMSSy program, a project for the development of the Romanian Emergency Medical Services; editing of text books 2006, and releasing the printed version of the Romanian resuscitation guide lines 2010; active involvement in increasing the quality of the medical care in fields as resuscitation and post-resuscitation care; management of a nation wide network of instructors for advanced and basic life support; implication into the European Registry of Cardiac Arrest (EuReCa) since 2010; main promoter of the Romanian Registry of Cardiac Arrest; Organization of international conferences in the field of resuscitation 2005 and trauma in 2007.

Detecting failures in the standard procedures is highly important in order to improve rescue team performance and possibly save lives. We propose an agent-based approach for detecting breaches in the standard protocols. Ideally, nation-wide registries should be available for analysis so any solution in this direction should be scalable.

Our programming environment is the Erlang programming language. We have chosen Erlang because it is promoted as a language with built-in support for concurrency and distributed programming. Moreover, using a declarative language (like Erlang) seems to be a more natural choice for implementing agents. For example it is much easier to implement a rule-based system in a declarative language than in an imperative one (a rule base could represent the intelligence part of an agent).

Rule based systems are becoming key components in nowadays fields like Semantic Web [10]. In order to make better use of artificial intelligent research experience and the practical application of Semantic Web technologies, the World Wide Web Consortium (W3C) has adopted the Web ontology language (OWL) as a language for processing Web information [4]. Authors from [5] note that rules represent because the standard ontology language OWL provides only basic forms of reasoning. However, ensuring the correctness of a rule-based approach may not always be a trivial task since the actual system behaviour and the resources required to realize such rules may be difficult to predict. These issues are even more disturbing in distributed rule-based systems where several communicating rule-based programs exchange information via messages. Also, a communicated fact may be added asynchronously to the state of the rule based system at run-time, potentially triggering a new strand of computation which executes in parallel with current processing [10]. Since agents in a multi-agent system can operate asynchronously and in parallel their use in a distributed rule-based environment may result in increased overall speed. A multi-agent based approach in general is more desirable compared

to a centralised approach because it makes the system more robust, reliable, scalable, flexible, cost effective, easier to develop and reuse [14].

3. Theoretical background

An agent is an entity that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors [13]. An agent that always tries to optimize an appropriate performance measure is called a rational agent. Such a definition of a rational agent is fairly general and can include human agents (having eyes as sensors, hands as actuators), robotic agents (having cameras as sensors, wheels as actuators), or software agents (having a graphical user interface as sensor and as actuator). Software agents are generally thought to act independently of the user intervention. According to [13, 12] agents exhibit the following characteristics:

- autonomy
- reactivity
- pro-activity
- sociability
- intelligence
- mobility
- self-organization.

The most attractive property is self-organization, that is, the ability to improve its behaviour without external influence or guidance.

An agent always has an associated environment, being able to act autonomously in order to accomplish its objectives. It only plays a role and operates in the environment through its sensors and effectors.

An abstract view of an agent is best expressed by Figure 1. However, such a view of an agent is fairly general and hence this abstract architecture should be refined. As described in [15] the agent's decision function may be separated into perception and action subsystems. In this sense, the agent may be seen as a pair $Ag =< see, action >$, where the function *see* maps environment states to percepts and the function *action* maps sequences of percepts to actions:

$$see : E \to Per \tag{1}$$

$$action : Per^* \to Ac \tag{2}$$

An intelligent agent has all the abilities of a software agent plus the skill of communicating with other agents, being reactive at the changes in the environment, having at least a partial representation of it. An intelligent agent is driven by completing its goals, it has its own resources, is able to completely understand the environment and is also able to learn.

Usually agents coexist and interact forming Multi-agent Systems (MAS).

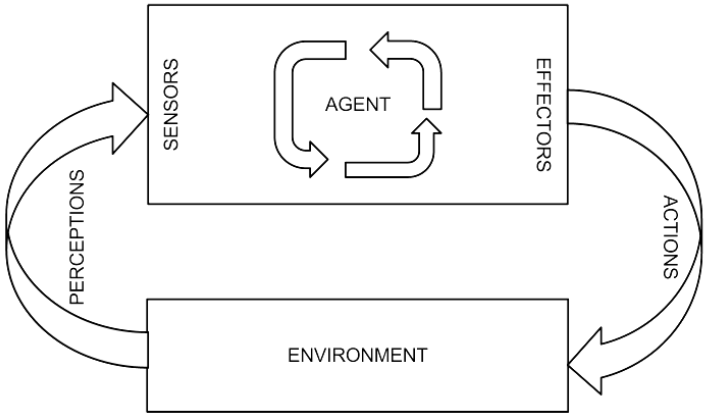

FIGURE 1. An agent perceiving its environment. The agent takes sensory input from the environment, and outputs actions that modify the environment.

Definition 3.1. *In computer science, a multi agent system (MAS) is a system composed of several interacting agents, collectively capable of reaching goals that are difficult to achieve by an individual agent or monolithic system.*

The precise nature of the agents is not clearly established. MAS may also include human agents. Examples of multi agent systems include human organizations and society in general.

Remark 3.1. *In a multi agent system, agents can be software agents, robots and also humans.*

Remark 3.2. *As a consequence to Remark 3.1 it follows that there is no single agent system.*

A multi agent system has the following advantages:

- it is inherently a distributed architecture and thus critical failures and performance bottlenecks are avoided
- allows interconnection of legacy systems
- problems like task allocation, team planning, complex phenomena simulation are naturally modelled in terms of interacting agents
- efficiently operates with information from spatially distributed sources
- suitable in situations where knowledge is distributed temporally and spatially

- enhances system performance at least in the following aspects of computational efficiency, robustness, reliability, and extensibility.

Some of the important things that need to be taken into consideration when dealing with a MAS are: communication between agents, collaboration and coordination [12]. The interactions between agents in a MAS can be either cooperative or selfish [12, 15, 11]. The information exchange is done using Agent Communication Languages like KQML [2] and FIPA ACL [3].

4. Proposed model

We propose an agent-based model for the parallel constraint verification problem.

Definition 4.1. *Given:*

- *A set of variables* $\mathcal{V} = \{V_i | i = \overline{1, |\mathcal{V}|}\}$
- *A set of domains* $\mathcal{D} = \{D_i | i = \overline{1, |\mathcal{D}|}\}$*, where each* D_i *is finite and discrete,* $|\mathcal{V}| = |\mathcal{D}|$ *and* $V_k \in D_k, \forall k = \overline{1, |\mathcal{V}|}$
- *A set of rules* $\mathcal{R} = \{R(P_i) | i = \overline{1, |\mathcal{R}|}\}$*, where each* P_i *represents a subset of variables from* $\mathcal{V}$ *, and each rule* $R(P_i)$ *represents a relation over the given subset of variables* P_i*.*

The constraint verification problem consists in finding all ordered bags

$$S =< v_1, v_2, \ldots, v_{|\mathcal{V}|} > \tag{3}$$

s.t. $\exists S^{'} \subseteq S, \exists i \in \{1, \ldots, |\mathcal{R}|\}$ *where* $R(P_i)$ *is not satisfied.*

We consider the environment E of an agent as a finite set of states:

$$E = \{e_i | i = \overline{1, |\mathcal{V}|}\}, \tag{4}$$

where $e_i \subseteq < v_1, \ldots, v_{|\mathcal{V}|} >$.

The set of all possible actions an agent may choose from is:

$$Ac = \{\alpha(R_i) | i = \overline{1, |Ac|}\}, \tag{5}$$

where $\alpha(R_i)$ denotes the action taken by the agent when rule $R_i \in \mathcal{R}$ is triggered.

Definition 4.2. *A behaviour, b, of an agent in an environment is a sequence of state transitions:*

$$b : e_0 \xrightarrow{\alpha(R_0)} e_1 \xrightarrow{\alpha(R_1)} \ldots \xrightarrow{\alpha(R_{u-1})} e_u \tag{6}$$

Definition 4.3. *An agent in the environment is:*

$$Ag : \mathcal{B}^E \rightarrow Ac \tag{7}$$

where $\mathcal{B}^E$ is the subset of all possible behaviours b that end with an environment state.

A sketch of our approach to the constraint verification process is given in Algorithms 4.1 and 4.2. The main algorithm spawns a new agent for each considered record of data. By a record we mean an ordered set of values for all variables from $\mathcal{V}$. In its behaviour, each agent is checking the rules consistency against the provided data record.

Algorithm 4.1 Spawn Agents

1: Input: $\mathcal{V}$, $\mathcal{D}$, $\mathcal{R}$
2: **for all** records $L =< v_1, v_2, \ldots, v_{|\mathcal{V}|} >$ **do**
3: $SpawnAgent(AgentBehaviour, L, \mathcal{R})$
4: **end for**

Algorithm 4.2 Agent Behaviour

1: Input: $L =< v_1, v_2, \ldots, v_{|\mathcal{V}|} >, \mathcal{R}$
2: **for all** rules $R \in \mathcal{R}$ **do**
3: $P \leftarrow GetValues(R, L)$
4: $SW \leftarrow CheckConsistency(R, P)$
5: $ProcessResult(P, SW)$
6: **end for**

Algorithm 4.2 describes an agent's behaviour. The agent receives an instance L and the set of rules $\mathcal{R}$. For each rule R from the set of rules the agent checks whether the rule holds given L. $GetValues$ is a convenience function which, given L, returns a list P of variable values occurring in rule R. The $CheckConsistency$ function checks if the rule R holds given P and returns a boolean SW. In the $ProcessResult$ function we only print the values P in a file if SW is $false$, i.e., the rule R did not hold over L.

5. Experiments

In order to evaluate our approach we have performed two experiments. For the first one we have used a dataset provided by the Romanian Resuscitation Council while the second experiment is on a synthetic dataset.

One of the CNRR datasets consists of 698 instances and 49 variables of categorical data recording information such as: date of cardiac arrest, time of collapse, treatment before team arrival, time of CPR (Cardiopulmonary Resuscitation) by rescuer, defibrillation time by rescuer, number of shocks, time of first cardiac rhythm analysis, defibrillator type etc. The data is noisy and

there are a lot of missing values. After a preprocessing phase 62 instances having the following variables were selected for analysis: patient id, time of CA (Cardiac Arrest) establishment, time of CPR, time of first cardiac rhythm analysis, time of defibrillation. Between these variables there is only one constraint: they have to be in increasing order.

In Table 1 we show the problematic instances as resulted from the execution on the CNRR dataset.

Id	CA	CPR	First CRA	Defibrillation	Deceased
73	9:05	9:00	9:00	9:00	6 Feb 2011
19	0:45	0:45	23:00	0:45	None
1225	19:40	19:30	19:27	19:34	5 May 2009
2170	16:25	16:45	16:47	16:46	8 Jul 2011

TABLE 1. CNRR dataset. Column Id represents the patient Id, column CA denotes the time of cardiac arrest, CPR represents the time of cardiopulmonary resuscitation, column $FirstCRA$ represents the time of the first cardiac rhythm analysis, $Defibrillation$ represents the time of defibrillation and column $Deceased$ represents the decease date of the patient.

Instance 73 is not valid because the reported time of cardiac arrest is after the reported time of CPR. In case of instance 19 the time of first cardiac rhythm analysis is far from the other reported times. Instance 1225 is problematic since the times of cardiac arrest, cardiopulmonary resuscitation and first cardiac rhythm analysis are not in chronological order. In instance 2170 reports a defibrillation time before the time of First CRA. We notice that all patients are deceased or the information regarding their survival is missing.

Analysing the considered dataset (after preprocessing and feature selection) we notice that 45.16% of the patients have actually survived. The fact that all the reported problematic instances represent data from patients that have not survived is disturbing. The dataset resulted after preprocessing is indeed small compared to the original and hence a general conclusion regarding a certain correlation cannot be drawn. Nevertheless, the question is if the procedures followed by the rescue team in the case of the reported problems have been accurately followed. Our results show that the procedures may not have been followed in the case of patients from Table 1. Could this be the cause of their death?

Our second experiment was done on a synthetic dataset. We have generated 10000 instances with 10 variables imposing the following rules:

$$R1 : X1 \leq X2 \leq X3 \leq X4, \tag{8}$$

$$R2 : X5 \leq X6, \tag{9}$$

$$R3 : X7 \leq X8, \tag{10}$$

$$R4 : X9 \leq X10. \tag{11}$$

A snapshot of the dataset is shown in Table 2.

Id	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
0	19:18	15:04	8:15	17:09	13:10	5:09	10:42	0:06	19:00	18:35
1	23:06	7:08	3:48	1:57	20:48	22:33	12:42	10:45	6:18	10:08
2	3:47	14:03	12:57	12:36	11:18	8:30	16:49	12:22	22:03	3:25

TABLE 2. Synthetic dataset (snapshot).

Table 3 shows the problematic times (with respect to the considered rules) of the instances considered in Table 2.

Id 0	Id 1	Id 2
19:18 15:04 8:15 17:09	23:06 7:08 3:48 1:57	3:47 14:03 12:57 12:36
13:10 5:09	None	11:18 8:30
10:42 0:06	12:42 10:45	16:49 12:22
19:00 18:35	None	22:03 3:25

TABLE 3. Synthetic dataset (snapshot) — result.

In Figure 2 we show the performance of the execution on the synthetic dataset considering 10 episodes.

6. Comparison to related work

Authors from [7] use a two dimensional grid of agents in order to check that all constraints of a CSP problem are satisfied. As opposed to their approach, our aim is to find all variable assignments for which some constraint does not hold. Thus, even though both approaches deal with software agents and rule-checking, the proposed frameworks are fundamentally different.

In [8], the authors introduce a methodology called constraint partition and coordinated reaction for distributed constraint satisfaction. In their approach the instantiation of a variable depends on a set of agents who can negate or

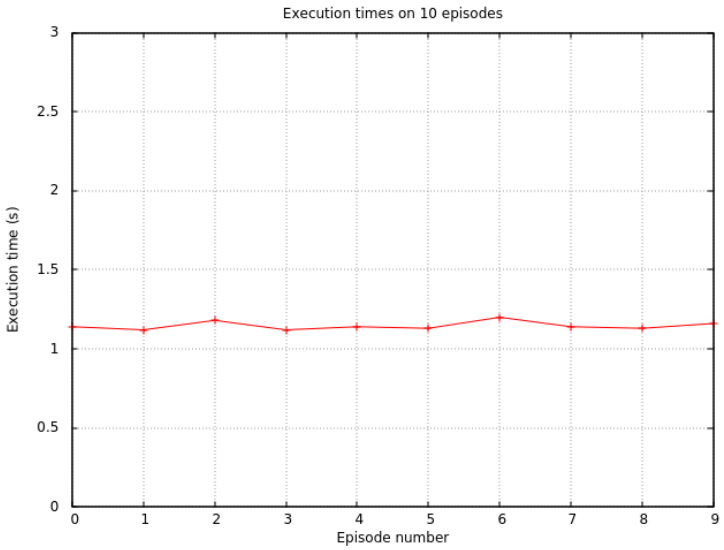

FIGURE 2. Execution performance on 10 episodes.

approve the instantiation of the same variable confirmed by some agent beforehand, and this is what authors from [8] call distributed constraint satisfaction with coordination reaction. A final solution is an instantiation of all variables that all agents agree on, i.e. it does not violate any constraints. The authors from [8] have improved this method by adding the responsibility for enforcing constraints of a particular type to specialist agents. The agents coordinate to iteratively change the instantiation of variables under their jurisdiction according to their specialized perspective. Experiments on a benchmark suite of job shop scheduling problems show promising results. While authors from [8] are focusing on finding variable values that match all rules, our aim is to find all instances that don't match some rule.

Authors from [16] advocate that distributed constraint satisfaction problem solving provide a useful mechanism for formalizing cooperative distributed problems in general. They present the asynchronous backtracking algorithm that allows agents to act asynchronously and concurrently, in contrast to the traditional sequential backtracking techniques employed in constraint satisfaction problems. The authors from [16] use one agent for each variable. Each agent instantiate its variable concurrently and use message passing in order come to an agreement. In contrast to this approach we assign agents to instances, not to variables. As soon as an agent has evaluated an instance, it may resume its execution and check another instance (or it may be replaced

by another newly created agent) and hence the number of agents remains constant with respect to the number of instances. This is not the case in the approach from [16] though, where the number of agents grows linearly with the number of variables; however this is seldom an issue.

7. Conclusions and future work

An agent-based approach to parallel constraint verification is presented in this paper. We use a dataset provided by CNRR and synthetic one.

The CNRR provided dataset contains highly noisy data and hence after preprocessing it we obtain only a small fraction of the original instances. Our agent-based approach is able to detect problematic instances corresponding to patients that have not survived. Actually all the detected instances correspond to patients that are either dead or any information regarding survival is missing.

Our future work will focus on: extending our system with a rule inference engine, fuzzifying the rules, improving the performance of our system.

Acknowledgements

The authors gratefully acknowledge the effort of the CNRR team responsible with collecting and preparing the CNRR data set. We particularily thank Dr. Valentin Georgescu, Dr. Mihaela Godeanu, Dr. Horea Sabău, Dr. Victor Strâmbu, Dr. Oana Tudorache, from the CNRR office in Bucharest.

This work was supported by a grant of the Romanian National Authority for Scientific Research, CNDI–UEFISCDI, project number PN-II-PT-PCCA-2011-3.2-0917.

References

[1] Consiliul Naţional Român de Resuscitare. http://www.cnrr.org/.

[2] T. Finin, Y. Labrou, and J. Mayfield. *Kqml as an Agent Communication Language.* Software Agents, B.M. Jeffrey, MIT Press, 1997.

[3] Foundation for Intelligent Physical Agents. http://www.fipa.org/.

[4] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. Owl 2: The next step for owl. *Web Semant.*, 6(4):309–322, November 2008.

[5] Ian Horrocks and Peter F. Patel-Schneider. A proposal for an owl rules language. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 723–731, New York, NY, USA, 2004. ACM.

[6] Ian Jacobs, Vinay Nadkarni, the ILCOR Task Force on Cardiac Arrest, Cardiopulmonary Resuscitation Outcomes, Jan Bahr, Robert A. Berg, John E. Billi, Leo Bossaert, Pascal Cassan, Ashraf Coovadia, Kate D'Este, Judith Finn, Henry Halperin, Anthony Handley, Johan Herlitz, Robert Hickey, Ahamed Idris, Walter Kloeck, Gregory L. Larkin, Mary E. Mancini, Pip Mason, Gregory Mears, Koenraad Monsieurs, William

Montgomery, Peter Morley, Graham Nichol, Jerry Nolan, Kazuo Okada, Jeffrey Perlman, Michael Shuster, Petter A. Steen, Fritz Sterz, James Tibballs, Sergio Timerman, Tanya Truitt, and David Zideman. Cardiac Arrest and Cardiopulmonary Resuscitation Outcome Reports: Update and Simplification of the Utstein Templates for Resuscitation Registries: A Statement for Healthcare Professionals From a Task Force of the International Liaison Committee on Resuscitation (American Heart Association, European Resuscitation Council, Australian Resuscitation Council, New Zealand Resuscitation Council, Heart and Stroke Foundation of Canada, InterAmerican Heart Foundation, Resuscitation Councils of Southern Africa). *Circulation*, 110(21):3385–3397.

[7] Jiming Liu, Han Jing, and Y. Y. Tang. Multi-agent oriented constraint satisfaction. *Artif. Intell.*, 136(1):101–144, February 2002.

[8] Jyishane Liu and Katia P. Sycara. Distributed constraint satisfaction through constraint partition and coordinated reaction. In *Proceedings of the 12th International Workshop on Distributed AI*, 1993.

[9] Mary Ann A. Peberdy, William Kaye, Joseph P. Ornato, Gregory L. Larkin, Vinay Nadkarni, Mary Elizabeth E. Mancini, Robert A. Berg, Graham Nichol, and Tanya Lane-Trultt. Cardiopulmonary resuscitation of adults in the hospital: a report of 14720 cardiac arrests from the National Registry of Cardiopulmonary Resuscitation. *Resuscitation*, 58(3):297–308, September 2003.

[10] Abdur Rakib, RokanUddin Faruqui, and Wendy MacCaull. Verifying resource requirements for ontology-driven rule-based agents. In Thomas Lukasiewicz and Attila Sali, editors, *Foundations of Information and Knowledge Systems*, volume 7153 of *Lecture Notes in Computer Science*, pages 312–331. Springer Berlin Heidelberg, 2012.

[11] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[12] Gabriela Şerban. *Sisteme Mutiagent in inteligenta artificiala distribuita. Arhitecturi si aplicatii*. Ed. Risoprint, Cluj-Napoca, 2006.

[13] Gabriela Şerban and Horia Florin Pop. *Tehnici de Inteligenta Artificiala. Abordari bazate pe Agenti Inteligenti*. Ed. Mediamira, Cluj-Napoca, 2004.

[14] Gerhard Weiss, editor. *Multiagent systems: a modern approach to distributed artificial intelligence*. MIT Press, Cambridge, MA, USA, 1999.

[15] Michael Wooldridge. *Intelligent Agents, An Introduction to Multiagent Systems*. Ed. G. Weiss, 1999.

[16] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, June 2000.

(1) Babeş-Bolyai University, Department of Computer Science, Cluj-Napoca, Romania

E-mail address: rgaceanu@cs.ubbcluj.ro

E-mail address: hfpop@cs.ubbcluj.ro

E-mail address: ssie1324@scs.ubbcluj.ro