Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial
ISSN: 1137-3601
revista@aepia.org
Asociación Española para la Inteligencia Artificial
España

Cetnarowicz, Krzysztof; Cieçiwa, Renata; Rojek, Gabriel
Behavior Evaluation with Earlier Results Collection in Multi-Agent System
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 9, núm. 28, invierno, 2005, pp. 85-92
Asociación Española para la Inteligencia Artificial
Valencia, España

Behavior Evaluation with Earlier Results Collection in Multi–Agent System

Krzysztof Cetnarowicz*, Renata Cięciwa, Gabriel Rojek*****

*Institute of Computer Science
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
cetnar@agh.edu.pl
**Department of Computer Networks
Nowy Sącz School of Business — National-Louis University
ul. Zielona 27, 33-300 Nowy Sącz, Poland
rcieciwa@wsb-nlu.edu.pl
***Department of Computer Science in Industry
AGH University of Science and Technology
Al. Mickiewicza 30, 30-059 Kraków, Poland
rojek@agh.edu.pl

Abstract

This article concerns a security system which enables the detection of undesirable agents, intruders in a multi–agent system. The intruders are identified on the basis of their behavior which is determined by actions they undertook. The process of behavior evaluation is distributed — each agent makes autonomous behavior evaluation of other agents. In order to distinguish if an agent is good or bad (an intruder), results of behavior evaluations of this agent have to be collected and processed. The problem of collecting and processing of the results of distributed behavior evaluation is the main topic of this article. The conception of storing agents' behavior evaluation results obtained during earlier periods of time is presented. The decision about an agent is undertaken on the basis of the analysis of the gathered results of his behavior evaluation.

Keywords: multi–agent system, security, behavior evaluation, ethically-social security mechanisms, intruders detection.

1. Introduction

Nowadays the number and frequency of destructive attacks is increasing. Thus it is crucial to identify intruders properly and undertake correct reaction to the detected attack. In critical situations the reaction time should be very short in order to stop the undesirable activity in secured system. Security system has to detect and react to new kind of dangers that have never been encountered before.

The aim of our work is to obtain a computer security system which should identify intruders that are unknown at the moment of system creation. Some analogies to the mechanisms existing in the human society which provide a person surrounded by other possible dishonest people with security were observed. The main security assumptions related to the ethically-social mechanisms could be stated as follows:

- distributed evaluation of entities (agents, processes in the system) the way that people in the society evaluate other people,
- evaluation on the base of behavior (actions which are undertaken) instead of evaluation on the basis of resource structure as it is done by e.g. computer antivirus software.

Security systems which mention these paradigms were presented in our earlier work presented in e.g. [4, 5, 2]. There are two main problems connected with the creation of security system that fulfill the stated assumptions:

- creation of security functions, mechanisms that are built-in all agents and that realize evaluation on the basis of their behavior,
- collection and processing of results of distributed behavior evaluations that are made autonomously by an agent existing in the system.

The first stated problem was solved with the use of immunological mechanisms and some additional mechanisms to store information of all actions undertaking in a system. The second stated problem is the main issue of this article however in the past some simple solutions were presented.

2. Agent's Algorithm of Behavior Evaluation

All agents in the system has been equipped with some additional goals, tasks and mechanisms in order to ensure security of the entire multi-agent system. These mechanisms has been named *division profile*. The name *division profile* is inspired by M–agent architecture which could be used to describe an agent (M–agent architecture was introduced among others in [1, 3]).

A more detailed description of the division profile appears in [4, 2, 5]. This article contains only some information that is crucial to the modified approach to behavior evaluation process with the elimination table.

Actions undertaken by agents can be perceived as objects, which create a sequence registered by all agents in the environment. On the basis of analysis of these sequences an agent can evaluate the others and determine *a good* or *a bad* acting agents (called also *intruders*).

The division profile of an agent has three stages of functioning:

1. creation of the collection of sequences of actions,
2. generation of the detector set on the basis of *good* (*self*) sequences of actions,
3. behavior evaluation.

2.1. Creation of the Collection of Sequences of Actions

Each agent in the environment creates the sequence of his own (*good*) actions. He also registers sequences of actions of the other agents existed in the environment.

Example 1:
The system consists of three agents: *Ag1, Ag2, Ag3*. An agent can register only the last 7 actions undertaken by each agent in the environment ($h = 7$). If A and B indicate the possible actions of an agent, the observed sequences of actions could be stated as follows:
Ag1: ABBABAA,
Ag2: BAAABBB,
Ag3: AAABAAA

2.2. Generation of the Detector Set

The algorithm of detectors generation refers to the negative selection - the method of T–lymphocytes generation (presented in [6, 7, 8, 10]). An agent generate the preliminary set of detectors with the length equals to l (the sequences from this set represents every possible actions in every possible order). Detectors reacting with any sequence from own collection are rejected from the preliminary set. Sequences from this set which will pass a negative selection create the final set of detectors.

Example 2:
For *l=3* and two types of actions the preliminary set of detectors, which is generated by

each agent in the environment, is equal to $R_0 = \{AAA, AAB, ABA, ABB, BAA, BAB, BBA, BBB\}$ An agent compares the detectors from this set with the collection of his own actions and rejects the reacting ones. The agents: *Ag1, Ag2, Ag3*, mentioned in the example 1, create the final set of detectors as follows:
Ag1: $\{AAA, AAB, BBB\}$
Ag2: $\{ABA, BAB, BBA\}$
Ag3: $\{ABB, BAB, BBA, BBB\}$

2.3. Behavior Evaluation

An agent a, which division profile is at his behavior evaluation stage, can assign the coefficient m_a^k to an agent k, where $1 \leq k \leq j$ (j is the number of neighboring agents which are visible for agent a).

The coefficient m_a^k is a number of counted matches between:

- detectors of an agent a which evaluates behavior,
- a sequence of actions undertaken by agent number k.

Marking the length of a detector as l and the length of the sequence of actions as h, the coefficient m_a^k is a number from a range $\langle 0, h-l+1 \rangle$. The maximum of counted matches is equal to $h-l+1$, because every fragment of the sequence of actions, which has a length equal to the length of a detector, can match only one detector.

Example 3:
Let us assume that the agent *Ag3* has such a sequence of actions AAABAAA and the agents *Ag1, Ag2* have the final sets of detectors as it was mentioned in the example 2. The agent *Ag1* can notice 3 matches between his detectors and the sequence of actions of the agent *Ag3* (the detector AAA - 2 matches, the detector AAB - 1 match), thus the coefficient $m_{Ag1}^{Ag3} = 3$. The agent *Ag2* can find only one match between his detectors and the sequence of actions of the agent *Ag3* - the coefficient $m^{Ag3} = 1$

3. Interaction between an Agent and the Environment

In order to distinguish if an agent is good or bad this agent have to be evaluated by all agents in the environment and the results of this evaluation have to be collected and processed. Mechanisms of collecting and processing of the results are built-in into environment of multi–agent systems, so the environment have to interact with all agents. The solution of the problem collection and processing of the results are presented below from an agent's and the environment's point of view.

3.1. Behavior of an Agent

An agent a in case of receiving a request of evaluation of an agent number k sends back only the coefficient o_a^k in the range $0 \leq o_a^k \leq 1$. The coefficient o_a^k is given by function:

$$o_a^k = \left(\frac{m_a^k}{h-l+1} \right)^4 \qquad (1)$$

where $h-l+1$ is the maximum of counted matches of an agent a. The oefficient o_a^k is sended back to the environment.

The power function of behavior evaluation increases a weight of high coefficient m_a^k. As a result, an agent with high number of counted matches obtains coefficient o_a^k much higher than an agent with low number of counted matches. The exponent of power function has been set empirically (the discussion of the use of power function is presented in [9]).

Example 4:
Let us assume that agents *Ag1*, *Ag2* received requests of evaluation of the agent *Ag3*. Agents *Ag1* and *Ag2* calculate their coefficients (according to equation 1 and behavior analisis in example 3):
$o_{Ag1}^{Ag3} = (3/5)^4 = 0.1296$
$o_{Ag2}^{Ag3} = (1/5)^4 = 0.0016$
Agents *Ag1*, *Ag2* sends calculated coefficients back to environment.

3.2. Behavior of the Environment

Each action undertaken by an agent may cause the change of results of behavior evaluations that are done by other agents in the system. This approach lets us formulate *the algorithm of evaluation management* as follows:

> If an agent k undertakes an action, a request of evaluation the agent k is sent to all agents (except the agent k) by the environment.

After sending the request of evaluation of an agent number k the environment uses *the algorithm of evaluation's collecting and processing*, which consists of following actions:

1. Agents send back coefficients as it is described in Sect. 3.1.
2. The gained coefficients are summed and then this sum is divided by $j-1$ (j is the number of agents):

$$o_*^k = \frac{o_1^k + ... + o_{k-1}^k + o_{k+1}^k + ... + o_j^k}{j-1} \quad (2)$$

3. If the coefficient o_*^k is greater than $\frac{1}{2}$ agent k is eliminated.

Example 5:
The environment got the coeffiecients:
$o_{Ag1}^{Ag3} = (3/5)^4 = 0.1296$
$o_{Ag2}^{Ag3} = (1/5)^4 = 0.0016$
The coefficient o_*^{Ag3} is counted as follows:
$o_*^{Ag3} = (0.1296 + 0.0016)/2 = 0.0656$
The coefficient o_*^{Ag3} is smaller than $\frac{1}{2}$, so the agent *Ag3* is not eliminated, however his behavior is in small degree different from behavior of agents *Ag1* and *Ag2*.

4. Example of Behavior Evaluation Experiment

In this experiment a multi–agent system with asynchronously acting agents was implemented. In the simulated environment there are two types of resources: resources of type A and resources of type B. This situation reflects these operations in computer system which should be executed in couples e.g. opening / closing a file. Resources are used by agents, but refilling all resources is only possible when each type of resources reaches the established low level. The simulated system has three types of agents:

- *type g=0* – agents which take one unit of randomly selected (A–50 %, B–50 %) resource in every full life cycle;
- *type g=1* – agents which take one unit of randomly selected (A–75 %, B–25 %) resource in every full life cycle; type g=1 agents can be treated as intruders, because increased probability of undertaking only actions of one type can cause blocking the system (what is presented in [4, 2]);
- *type g=2* – agents which take one unit of A resource in every full life cycle; type g=2 agents are also called intruders.

Actions of agents of type g=1 are similar to actions of agents of type g=0 but they are also undesirable in the secured system.

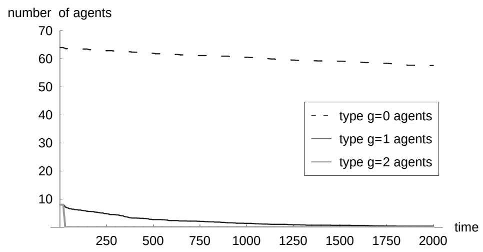

Fig. 1. Number of agents in separate time periods

The case in which initially there are 64 agents of type g=0, 8 agents of type g=1 and 8 agents of type g=2 is presented below. All agents in the system are equipped with the division profile mechanisms with parameters $h = 18$ and $l = 5$. The simulations are run to 2000 constant time periods Δt and 20 simulations were performed. The diagram in Fig.1 shows the average number of agents in separate time periods.

During the first 18 time periods Δt all agents were acting synchronously. In 18th time period all agents have generated their detectors and achieved the third stage of their division profiles — behavior evaluation. From 19th time period agents were acting asynchronously (an agent could be activated in one time period Δt, but had to be activated at least once during ten time periods Δt) and using their detectors to evaluate agents which undertook an actions according to algorithms presented in Sect. 3.

As we can see on the diagram in Fig. 1 agents of type g=2 were being deleted successively from 19 constant time period Δt to 28 constant time period Δt. When the system has achieved behavior evaluation stage all bad agents were identified properly and eliminated when they tried to undertake actions.

At the end of presented simulation the agents of type g=1 were eliminated in 96 %, but the agents of type g=0 were eliminated in 10 % as well. The elimination of good agents has been named *the phenomenon of self–destruction.* This phenomenon could be caused by the random choice of undertaken action. As a result, some sequences of actions of good agents can be similar to actions of bad agents. Thus the algorithms presented in Sect. 3 are not sufficient for the limitation of the phenomenon of self–destruction.

5. Earlier Results Collection

In order to reduce the phenomenon of self–destruction of agents the environment was equipped with the elimination table (presented onto Fig. 2), which allow taking into account the coefficients obtained during earlier live cycles of agents.

In this table j is the number of all agents in the environment and c_v is the length of coefficients' vector. The last c_v returned coefficients o_*^k of an agent k are stored in his own vector. All agents' vectors form the elimination table.

The algorithm of evaluations's collecting and processing (presented in Sect. 3.2) has been changed as follows:

1. Agents send back coefficients as it is described in Sect. 3.1.

2. The gained coefficients are summed and then this sum is divided by $j - 1$ (j is the number of agents):

$$o_*^k = \frac{o_1^k + ... + o_{k-1}^k + o_{k+1}^k + ... + o_j^k}{j-1} \quad (3)$$

3. If the coefficients' vector of an agent k is full the first coefficient in this vector is removed and the other ones are moved left. In the elimination table the coefficient o_*^k is stored at the end of the vector of an agent k.

4. If the gathered coefficients o_*^k meet specific criterion agent k is eliminated (two criterions are presented and tested in next section).

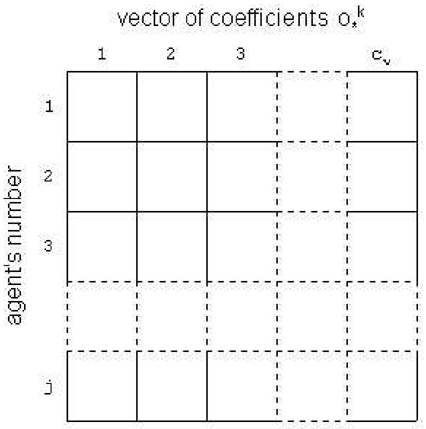

Fig. 2. Elimination table

6. Example of Behavior Evaluation Experiment with Earlier Results Collection

It is crucial to find the proper criterion of agents elimination on the basis of coefficients gathered in the elimination table. In order to select this criterion a multi–agent system with asynchronously acting agents of type g=0, type g=1 and type g=2 was implemented, as it was specified in Sect. 4. All agents in the system are equipped with the same division profile parameters — $h = 18$ and $l = 5$. The multi–agent system analogous to the experiment in Sect. 4 in which initially there were 64 type g=0 agents, 8 agents of type g=1 and 8 agents of type g=2 was researched, but additionally the elimination table with the length c_v was applied. The length of this table was increased up to $c_v = 20$. The simulations were run to 2000 constant time periods Δt. Results presented in the next paragraphs are in all cases the average of 20 runs of simulation.

6.1. The Average Value of Coefficients

First a case was simulated in which an agent k is eliminated if

$$\frac{\sum\limits_{i=1}^{n} o_{*i}^{k}}{n} > 0{,}5 \qquad (4)$$

where n is the number of coefficients gathered in the vector assigned to an agent k ($1 \leq n \leq c_v$).

The diagram in Fig. 3 shows the percent of type g=0 and g=1 agents remained in the system after 2000 constant time periods Δt in the cases of the elimination table with the length $c_v = 1, 2, 3, ..., 20$

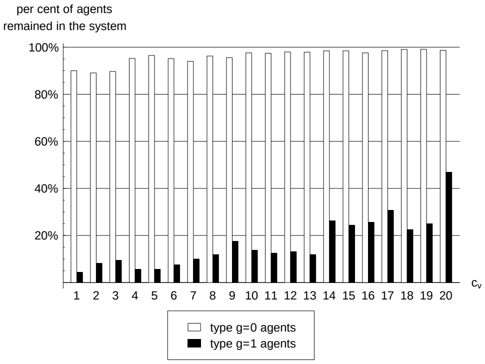

Fig. 3. Per cent of type g=0 and g=1 agents remained in the system

The value $c_v = 1$ means that there is behavior evaluation without the evaluation table, because an agent k is evaluated only on the basis of his last returned coefficient o_*^k.

The analysis of obtained results indicates that the proposed solution reduces the phenomenon of self–destruction of good agents. For $c_v = 4$ over 95 % of type g=0 agents remained in the system, for $c_v \geq 18$ this percentage exceeds 99 %.

However the length of the elimination table highly influences on number of type g=1 agents remained in the system. About 5 % of intruders remained in the system if $c_v = 4$, when $c_v \geq 14$ this percentage exceeds 20 %. The low coefficients obtained during earlier live cycles of the agents allow them to remain in the system despite deterioration of their actions. The increase of the length of the elimination table reduces also a rate of destruction of bad agents, what can be seen on diagram in Fig. 4.

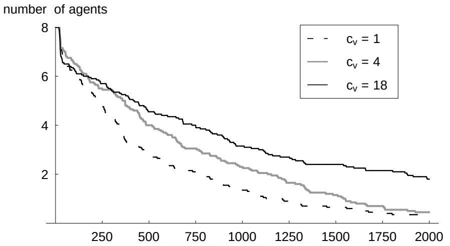

Fig. 4. Number of type g=1 agents in separate time periods for selected values of c_v

6.2. The Weighted-Average Value of Coefficients

In order to increase a weight of last returned coefficient o_*^k we have simulated also a case in which an agent k is eliminated if

$$\frac{\sum\limits_{i=1}^{n} (i \cdot o_{*i}^{k})}{\sum\limits_{i=1}^{n} i} > 0{,}5 \qquad (5)$$

where n is the number of coefficients gathered in the vector assigned to an agent k ($1 \leq n \leq c_v$).

The diagram in Fig. 5 shows the percent of type g=0 and g=1 agents remained in the system after 2000 constant time periods Δt in the cases of elimination table with length $c_v = 1, 2, 3, ..., 20$

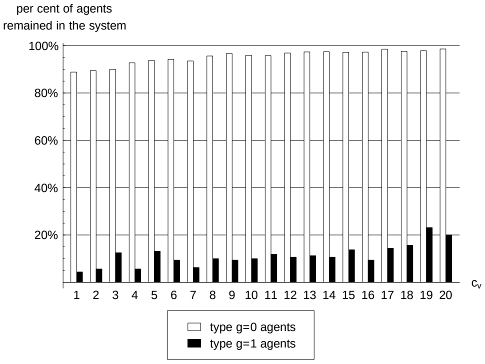

Fig. 5. Per cent of type g=0 and g=1 agents remained in the system

Analyzing the diagram in Fig. 5 for $c_v = 4$ we can notice that over 92 % of type g=0 agents remained in the system, for $c_v \geq 18$ this percentage exceeds only 97 %. The maximal per cent of remained good agents amounted to 98,5 % for $c_v = 20$.

The length of the elimination table influences also on the number of type g=1 agents remained in the system. However the per cent of remained intruders exceeded 20 % only when $c_v > 18$. The differences between rates of the destruction of bad agents for selected values of c_v are visible only during the first 250 periods of time (as it is presented in Fig. 6) when all agents were filling their coefficients' vectors. Afterwards the rate remains on the same level regardless of the increase of the length of the elimination table .

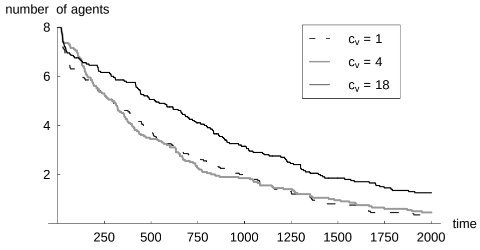

Fig. 6. Number of type g=1 agents in separate time periods for selected values of c_v

7. Conclusion

The main mechanisms of distributed behavior evaluation in multi–agent systems were presented in this article. The results of earlier experiments revealed occurrence of the phenomenon of self–destruction. This observation permits us to formulate a proposition of the elimination table. The elimination table with the parameter c_v means that an agent is attributed to a vector of coefficients with the length c_v. This vector stores last c_v coefficients o^k_* obtained by an agent k during behavior evaluation process.

A multi–agent system equipped with the elimination table was simulated and two criterions of agents' elimination has been proposed. Obtained results let us to formulate some directions of use these criterions. In the case in which there is crucial to retain all good agents in the system the average value of coefficients reaches acceptable results. This criterion ensures 5 % reduction of the phenomenon of self–destruction for $c_v = 4$ and for very long tables per cent of good agents remained in the system reaches 99 %. The weighted-average criterion of agents' elimination is useful in the cases in which there is crucial to reduce the number of good agents eliminated from the system without substantial reducing the level of intruders elimination.

Considering the phenomenon of self–destruction and the problem of deleting agents which are intruders, but their behavior is similar to good agents, it could be stated that in general elimination table is desirable in evaluations algorithm. However the obtained results do not differ much from results obtained for algorithms without earlier results collection. The reason for this conclusion is the fact, that agents evaluate behavior on the basis of the last h (h=18 in presented tests) actions, so the information about behavior of an agent is duplicated.

Referencias

[1] Krzysztof Cetnarowicz. M–agent architecture based method of development of multiagent systems. In *Proc. of the 8th Joint EPS-APS International Conference on Physics Computing*, pages 41–46. ACC Cyfronet, Kraków, 1996.

[2] Krzysztof Cetnarowicz, Renata Cięciwa, and Gabriel Rojek. Behavior based detection of unfavorable activities in multi-agent systems. In Gaston Lefranc, editor, *MCPL 2004 IFAC/IEEE/ACCA: conference on Management and Control of Production and Logistics: Santiago de Chile, November 3-5, 2004*, pages 325–330. International Federation of Automatic Control IFAC; IEEE Chilean Chapter on Control, Robotics and Cybernetics, 2004.

[3] Krzysztof Cetnarowicz, Edward Nawarecki, and Malgorzata Żabińska. M–agent architecture and its application to the agent oriented technology. In *Proc. of the DAIMAS'97*. St. Petersburg, 1997.

[4] Krzysztof Cetnarowicz and Gabriel Rojek. Unfavourable behaviour detection with the immunological approach. In Jan Stefan, editor, *ASIS 2003: Advanced Simulation of Systems: proceedings of XXVth internation-*

al autumn colloquium, pages 41–46. MARQ, Ostrava, 2003.

[5] Krzysztof Cetnarowicz and Gabriel Rojek. Behavior based detection of unfavorable resources. In *Lecture Notes in Computer Science, Volume 3038*, pages 607–614. Springer-Verlag, Berlin, 2004.

[6] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, and Rajesh Cherukuri. Self-nonself discrimination in a computer. In Los Alamitos, editor, *Proc. of the 1994 IEEE Symposium on Research in Security and Privacy*, pages 202–212. IEEE Computer Society Press, 1994.

[7] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, and Rajesh Cherukuri. A change–detection algorithm inspired by the immune system. In Los Alamitos, editor, *IEEE Transactions on software Engineering.* IEEE Computer Society Press, 1995.

[8] Steven A. Hofmeyr and Stephanie Forrest. Architecture for an artificial immune system. In *Evolutionary Computation, vol. 7, No. 1*, pages 45–68, 2002.

[9] Gabriel Rojek, Renata Cięciwa, and Krzysztof Cetnarowicz. Algorithm of behavior evaluation in multi–agent system. In *Lecture Notes in Computer Science, Volume 3516*, pages 711 – 718. Springer-Verlag, Berlin, 2005.

[10] Slowmir T. Wierzchoń. *Sztuczne systemy immunologiczne: teoria i zastosowania.* Akademicka Oficyna Wydawnicza Exit, Warszawa, 2001.