International Journal of Intelligence Science, **2013, 3, 69-76**
http://dx.doi.org/10.4236/ijis.2013.32008 Published Online April 2013 (http://www.scirp.org/journal/ijis)

Automatic Negotiation with Profiles and Clustering of Agents

Serban Radu, Eugenia Kalisz, Adina Magda Florea
Department of Computer Science, University Politehnica of Bucharest, Bucharest, Romania
Email: adina.florea@cs.pub.ro

Received January 26, 2013; revised February 28, 2013; accepted March 26, 2013

ABSTRACT

This paper presents a model of automatic negotiation agents in an open environment. Agents are motivated by the gain they may obtain while fulfilling their goals, but their behaviour can change during negotiation according to previous interactions with other agents in the system. Changing behaviour may refer to either the use of different negotiation strategies or to concessions made for other agents, with which they have successfully negotiated in the past. To this aim, an agent develops a set of partners' profiles during negotiation: the preference profile, the cooperation profile, and the group negotiation profile. The first two profiles characterize individuals, while in a group negotiation profile, several agent profiles are clustered according to commonly discovered features. Different approaches to the development of these profiles are presented.

Keywords: Agents; Automated Negotiation; Strategy; Negotiation Profiles; Classification; Q-Learning

1. Introduction

Negotiation between agents appears in different area of research, like electronic commerce, distributed resource allocation or virtual enterprises. In open systems, agents are acting in an environment in which other agents may enter or leave, some of them known before, some others encountered for the first time. In this context, the design of intelligent agents with a complete pre-defined negotiating behaviour represents a challenge for the designer, especially when the agents are conceived to be general purpose and not limited to a specified domain. To overcome existing difficulties, creating automatic negotiating agents is still a fertile are of research, despite the important amount of work in the domain.

An intelligent agent should be capable to forecast the result of a negotiation and the best potential partner to choose when negotiating a specific item. Machine learning techniques can be used to improve both the selection of the negotiation partner and the agent's strategy during negotiation.

Some of our previous work on negotiation was based on reinforcement learning to learn negotiation strategies, according to the agent previous interactions with its partners [1,2]. In this paper we present a different approach to design automatic negotiating agents in which the agents develop different profiles based on interactions with other agents. The profiles try to capture several aspects of these interactions and are used in different ways to tailor agents' behaviour. The work reported in this article extends the results presented in [3] and describes in detail how different profiles can be built, including the selection of group profile and the clustering of negotiation instances, based on machine learning algorithms.

During negotiation, the agents develop three types of negotiation profiles and may use either uninformed or informed negotiation strategies. The negotiation strategies are designed using rules, endowed with preference coefficients, which can be dynamically changed, with respect to the negotiation results.

The negotiation profiles the agent uses are: the preference profile, which implements the agent negotiation strategy, the cooperation profile, which improves the agent interaction with the partners, and the group-of-partners' negotiation profile, which clusters the profiles of several agents.

The proposed model of negotiation is tested in the framework of a multi-agent system, situated in an open environment, in which agents negotiate the tasks associated with building a house and acquiring the necessary materials.

The paper is organized as follows: Section 2 presents

the agents system, our proposed framework for the negotiation system in an open environment. Section 3 describes the agents' negotiation profiles and how each profile influences the behaviour of the agents. Section 4 presents how agents are classified, according to the negotiation behaviour, while Section 5 develops the algorithm for clustering negotiation states. Section 6 presents related work, while Section 7 discusses conclusions and future work.

2. The Agents System

The automated negotiation framework we propose consists of a multi-agent system, acting in an open environment. The agents can enter or leave the system at any time.

The agents are mainly designed according to the BDI (Belief-Desire-Intention) model, and have a set of goals, selected from the set of desires. In order to achieve their goals, they develop plans, as a sequence of actions to be performed, or they provide services or ask for services from the agents, such as buying or selling objects. Some actions can not be executed by the agent itself, and they also become, together with the services, objects to be negotiated by the agents. To unify these two cases, the agents are using negotiation objects, which can wrap up one or several negotiation attributes. For example, an agent A may ask an agent B to perform a service for him, for instance to paint its house, and specify for the negotiation object the attributes over which he is willing to negotiate (price, deadline, quality etc.). Alternately, the agent A may offer to agent B competitive service for package delivery, specifying a particular set of attributes to this service.

The agents negotiate using a heuristic negotiation model, based on the Contract Net Protocol, in which the agents compute their gain with respect to their private value for the object negotiated. Details about the negotiation protocol will be given in Section 4.2.

The agents' behaviour is conducted by the gain they may obtain after a negotiation, but also, in some situations, by the necessity to cooperate with other agents, in order to fulfill their goals.

Each agent has an associated set of rules, divided in two: *behaviour rules*, which implement the way the agent fulfills the goals assigned to it and *negotiation rules*, which describe the negotiation strategy. In **Figure 1** we present the structure of a BDI negotiating agent. Agents based on models different from BDI may enter the system, provided that they use the same negotiation protocol. During negotiation and interaction with other agent, a BDI agent develops a set of negotiation profiles, as described in the next section.

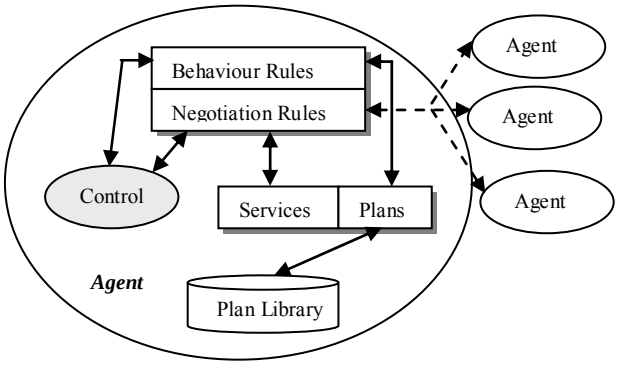

Figure 1. Agent structure.

3. Agents Negotiation Profiles

Each agent has a set of negotiation profiles:

- the *partner cooperation profile*, which tries to elect the results of the agent interactions with the other agents in the system;
- the *group negotiation profile*, which refers to a common profile of a group of negotiation partners;
- the *preference profile*, describing the agent negotiation strategy.

The partner cooperation profile is characteristic to each agent and contains the agent name and a set of attributes, upon which the agent can change its preferences. The cooperation profile is updated during the negotiation process, at the end of each negotiation.

A cooperation profile is formed of the following attributes, as presented in **Table 1**:

- the *name* of the partner agent;
- *how many times* the agent negotiated with its partner;
- the *number of successful negotiations*;
- the *total gain* obtained, based on the difference between the outcome of a negotiation and the private value of the agent for the negotiation object;
- the *gain ratio*, that is how much is the gain obtained while negotiating with the partners, as a percentage of the agent total gain;
- the *number of negotiation rounds*, during the last negotiation;
- the agents' beliefs about the partner abilities and/or credentials, represented in the field interesting degree. We call this attribute the *interesting degree* of the partner, and we quantify it as: *very interesting*, *interesting*, *moderately interesting*, and *not interesting* (denoted from 4 to 1). For example, if the partner has an ability to perform a task, which is lacking to the agent, then the partner is interesting or very interesting to the agent. Moreover, if the negotiation is successfully concluded in a small number of steps, and the gain is positive, then the partner is very interesting;
- the *classification* of the partner agent, which represents the current agent belief about the cooperation

Table 1. Partner cooperation profile of an agent.

Agent Name	Bob	Alice
No. of Negotiations	7	5
Of Which Successful	4	3
Total Gain	3	2
Gain Ratio	80	60
Negotiation Rounds	6	4
Interesting Degree	3	2
Partner Classification	very cooperative	cooperative

potential of the partner. The partners are classified in six cooperation classes: *highly cooperative*, *very cooperative*, *cooperative*, *slightly cooperative*, *non-cooperative* and *unknown*, the last one being the default value in case there is no specific information regarding the agent.

The first seven attributes described above are updated by the agent after each negotiation with a specific agent. The last attribute will be filled in by a more elaborate process, to be described in Section 4.1.

The group-of-partners' negotiation profile is created by grouping into classes the partner agents, with which the agent negotiated in the system. For each of the six values of the cooperation classes, the group negotiation profile contains the list of all agents that belong to a certain class. The agents, for which no class was found out yet, belong to the unknown class. The preference profile will be detailed in Section 4.2.

4. Agents Classification and Strategies

4.1. Agent Classification

The characterization of the cooperation potential of a partner agent is done by classifying the partners into cooperation classes, which classify the cooperation ability of the partner into six classes. The classification is done using the C4.5 learning algorithm [4], in which a decision tree is a classifier for the cooperation degree, expressed as a recursive partition of the instance space. In the decision tree, the attributes are represented by the first seven fields of the partner cooperation profile and the class by the partner classification field.

Decision trees are capable of handling datasets that may have errors. Also, decision trees are capable of handling datasets that may have missing values, like the gain value in our case.

A tree is either a leaf node labeled with a cooperation class, or a structure containing a test for an attribute, like the number of successful negotiations, linked to two or more nodes (or sub trees). So, to classify some instance for the cooperation potential, first we get its attribute-vector, and apply this vector to the tree. The tests are performed with these attributes, like the number of successful negotiations, the gain, the gain percent, reaching one or other leaf, to complete the classification process.

Through a top-down decision tree and a heuristic selection criterion, the process chooses the best test to split the data, creating a branch.

In order to build the classification tree, we need a set of training instances, with associated attributes, and a corresponding class. There are two possibilities to obtain the set of training instances. The first possibility is to let the system run and collect information, based on agent interaction. We can build the tree after a number n of negotiations, and then rebuild the tree at successive times, $n + 1$, $n + 2$, increasing thus the accuracy of the classification. The other possibility is to let the designer of the system to create a set of training instances, and run the C4.5 algorithm on this set. An initial classification tree will be thus obtained, which may be later on rebuild and refined, after the actual negotiation will take place.

The C4.5 algorithm should work with inadequate attributes and should obtain correct results. Also, it should decide if testing some supplementary attributes will increase or not the predictive accuracy of the decision tree. Considering an attribute A with random values, choosing this attribute will give a high informational gain. The gain should be greater than a given threshold, in order to eliminate the non relevant attributes.

There are cases when the classification can't be done. This happens when a leaf is obtained, in which not all the objects belong to the same cooperation class. In this case, the notion of membership to a cooperation class with a certain probability is used or the leaf is labeled with the cooperation class with the largest number of instances. If the classification ends in a leaf with an equal number of instances from two cooperation classes, the decision about the correct classification in a cooperation class is done randomly.

When there are missing attribute values in the training set, for instance the interesting degree of a partner, it is assigned the value of that attribute, which appear most often. A different approach for missing attributes is to assign values distributed over the values of the attribute A proportionally to the relative frequency of these values in the set of objects.

Figure 2 presents a part of the decision tree used for classifying the instances.

While interacting with the partners, the agents are classified in the right class. At the end of negotiation, an agent can change its cooperation class. There is a tradeoff between how often the agent cooperation class is updated,

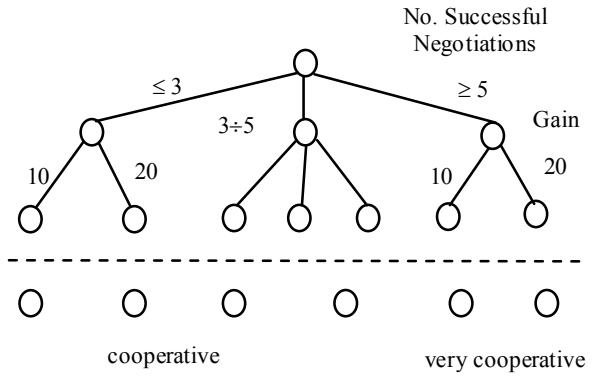

Figure 2. Classification using the partner cooperation profile.

which may be time consuming, and the accuracy of the classification.

4.2. Agent Strategies

The negotiation strategy of an agent is implemented in the form of conditional rules, with an associated preference coefficient (*PC*). If in a certain situation, several rules are applicable, then the negotiation strategy is responsible for selecting the rule to be applied from the conflict set. The preference coefficients indicate the designer priority among the rules, and are used to select the best rule to be applied. The preference coefficients may be static, that means defined together with the rules, and they will not be modified during the negotiation process. On the other hand, this preference coefficient may be dynamic and modified during negotiation.

When the preference coefficients are dynamically changed, according to the negotiation output, the rules' priorities are modified. The preference coefficients are included in the preference profile of the agent. In order to highlight the use of the preference profile and coefficients, in achieving the agent strategy, we need to express the negotiation rules and also the communication protocol.

In what follows we are going to give some examples of negotiation rules, which are written in a Prolog-like language and which are using communication primitives from the multi-agent system protocol.

The agents communicate using the following Contract Net Protocol (slightly modified):

a) ***cfp* (*A, X, NO, P*)** is the communication primitive, which represents a call for proposals from agent *A* to all the acquaintances *X*, regarding a negotiation object *NO*, with an associated price *P*;

b) ***Propose* (*X, A, NO, P, Step*)** is the communication primitive, which represents the response of agent *X* to the *cfp*, with the negotiation object *NO*, price *P* and negotiation step *Step*;

c) ***Accept* (*X, NO*)** indicates the acceptance of a proposal issued by *X*, for the *NO*;

d) ***Reject* (*X, NO*)** indicates the rejection of a proposal issued by *X*, for the *NO*;

e) ***Counterpropose* (*A, X, NO*$_1$*, P*$_1$*, Step*)** defines a communication primitive, which represents the counterproposal of agent *A* to the proposal of agent *X*, with the negotiation object NO_1, price P_1 and negotiation step *Step*.

In a Prolog-like language we define a set of predicates:

- *Propose* (*X*, *A*, *NO*, *P*, *Step*) defines a predicate, which is true when agent *A* receives the response (*b*) of agent *X* to the *cfp*, with the negotiation object *NO*, price *P* and negotiation step *Step*;
- *Accept* (*X*, *NO*) defines a predicate which, when true, triggers an acceptance message (*c*) of a proposal issued by *X*, for the *NO*;
- *Reject*(*X*, *NO*) defines a predicate which, when true, triggers a rejection message of a proposal issued by *X*, for the *NO*;
- *Counterpropose* (*A*, *X*, NO_1, P_1, *Step*) defines a predicate which, when true, represents the counterproposal (*e*) of agent *A* to the proposal of agent *X*, with the negotiation object NO_1, price P_1 and negotiation step *Step*;
- *tp* (*Ag_Name*, *Atr_Name*, *Value*) is a predicate which selects from the cooperation profile, for a given agent name (*Ag_Name*), the value (*Value*) of the attribute (*Atr_Name*) in the associated field.

Two examples of strategy rules involving an automated negotiation process about selling a house are presented below.

r_1: *propose*(*Alice*, *Bob*, *House*, 50000, *S*),
tp(*Alice*, *No_Successful_Negotiations*, v_1),
tp(*Alice*, *No_Negotiations_with_Partner*, v_2),
$v_1 < v_2 - 5$,
tp(*Alice*, *Interesting_Degree_of_Partner*, v_3),
$v_3 > 4$,
tp(*Alice*, *Gain_Percent*, v_4),
$v_4 > 30$
$\rightarrow$
accept(*Alice*, *House*) PC_1
r_2: *propose*(*Alice*, *Bob*, *House*, 60000, *S*),
tp(*Alice*, *No_Successful_Negotiations*, v_1),
$v_1 > 7$,
tp(*Alice*, *Gain*, v_2),
$v_2 > 500$,
tp(*Alice*, *Interesting_Degree_of_Partner*, 3),
$\rightarrow$
accept(*Alice*, *House*) PC_2

The rules may be simplified if we consider the classification in the partner cooperation profiles. For example, we can write the following rule:

r_3: *propose*(*Mary*, *Bob*, *Car*, 55000, *S*),
tp(*Mary*, *Classification*, *very cooperative*),

→
accept(*Mary*, *House*) PC_3

It is up to the designer to decide if the negotiation rules are written at a low level of granularity, as r_1 and r_2, or at a higher level, as r_3. At an even higher degree of granularity, instead of writing negotiation rules for a partner profile, we can write negotiation rules for group negotiation profiles. Moreover, these group negotiation profiles can be used in other ways to tailor the agent behaviour. The next section will describe how we can obtain the group negotiation profiles.

5. Computing the Preference Coefficients

The preference profile implements the negotiation strategy using rules, with the associated preference coefficient.

There are two types of rules used in the strategy. The first category of rules is used when the negotiation begins and is called uninformed strategy rules. The second category of strategy rules is used when there is enough information about the partner and is called informed strategy rules. In the beginning of negotiation, uninformed strategy rules can be applied. According to the success or failure of the negotiation, the preference coefficients are changed accordingly.

In order to update the preference coefficients, we have established a formula, which makes the connection between the preference coefficients and how much the agent gains using the rules associated to these coefficients. We suppose that all the rules applied during negotiation have an equal contribution to gain or to loss. For a certain agent acting in different negotiations, if there are n possible rules to be applied, a priority between the rules will be established, according to the preference coefficients. Suppose that N_R represents how many times the rule R was applied in a negotiation and M is the number of negotiation steps. Denote by α the ratio between N_R and M.

$$\alpha = N_R / M \tag{1}$$

The preference coefficient is updated according to the situation, if the negotiation ends with a deal or not. If a deal is reached at the end of the negotiation, then the preference coefficient is updated by the Formula 2:

$$PC_{\text{new}} = \begin{cases} (1+\alpha) * PC_{\text{old}} & PC_{\text{new}} \in [0,1] \\ (1+\alpha) * PC_{\text{old}} \div 2 & PC_{\text{new}} > 1 \end{cases} \tag{2}$$

If a negotiation is ending without a deal, then the preference coefficient is updated by the Formula 3:

$$PC_{\text{new}} = (1-\alpha) * PC_{\text{old}} \tag{3}$$

Some predefined values are put in the beginning for the preference coefficients. There is also a mechanism used to adjust the coefficients and to realize a combination between the initial preferences of the user for the rules and the change in time of the coefficients, according to the result of the negotiation.

A second approach to update the preference coef- ficients that we have considered is to use a reinforcement learning algorithm. In reinforcement learning, agents revise their strategies based on observed failure or success. In Q-learning [5], a reward function provides feedback on actions taken in order to estimate a ranking of state-action pairs.

To apply the Q-learning algorithm in our situation, we consider that each preference coefficient is indexed on a state and an action, for taking into account the preference coefficients. The actions from the Q-learning algorithm are represented by the rules applied by an agent when negotiating. The states from the Q-learning algorithm represent now the internal states of the agents. For each tuple (s, a), where s is the internal state of the agent and a represents the rule applied during negotiation, our preference coefficients are updated using a formula, in a similar manner to the Q-learning algorithm:

$$\begin{aligned} &PC(s,a) \leftarrow PC(s,a) \\ &+\alpha\left[r(s)+\gamma \max_{a'} PC(s',a') - PC(s,a)\right] \end{aligned} \tag{4}$$

where $\max_{a'} PC(s',a')$ is the expected preference coefficient of the next internal state of the agent s', when applying the rule a'. α is the learning rate representing the impact of the update value and $r(s)$ is the immediate reward for the internal state of the agent s. The immediate reward in our case is the gain obtained during negotiation. The factor γ specifies how much the values of the preference coefficients are discounted at each stage.

The internal state of the agent is characterized by several parameters. These parameters include: the partner agent, the negotiation object, the utility of the current offer, the number of negotiation parameters, the number of negotiation rounds with the partner agent, how much the agent gained in a previous negotiation.

To adequately update the preference coefficients using the Q-learning algorithm, the agent must encounter repeatedly the same pair (s, a), therefore the negotiation must be performed several times with the same agent and for the same object.

We have to consider that the matrix $PC(s, a)$ is huge, because it is possible to have many internal states of the agent, for each possible combination of the parameters for its internal states. For instance, if there are ten negotiation rules possible to be applied in a certain

state, then the matrix will have ten columns and the lines represent different combinations of state parameters. One line can represents the following internal state: the name of the partner agent, the negotiation round and the gain obtained in a previous negotiation. It is clear that there are too many internal states, for each possible combination of state parameters and the number of lines of the matrix increase exponentially, as the number of state parameters grows. That's why the learning algorithm used for updating the preference coefficients will converge in a very long time.

An idea to reduce the matrix dimensions of $PC(s, a)$ and to improve the convergence time of the learning algorithm is to group the internal states of the agent, according to the k-means algorithm [6]. Specifically, clusters of internal states are created and the number of lines of the matrix will significantly decrease, because each line will represent a cluster of states. In this way, the learning algorithm will converge in a reasonable time and the values of the preference coefficients are updated at the end of the learning algorithm application. An example of the clustering of internal states of an agent, according to the parameters of its internal state, is presented in **Figure 3**.

Each state representing the output of a negotiation is a point in a multidimensional space. The algorithm classifies the data set through a certain number of k clusters. The idea is to randomly define k centroids, one for each cluster. It is better to place the centroids as much as possible far away from each other. A better way to initialize the centroids is to use the k-means++ algorithm [7], in which the first centroid is randomly choosed from the initial data set. Then, each centroid is choosed from the remained objects, with a probability:

$$\frac{D(x_i)^2}{\sum_{x \in X} D(x)^2} \tag{5}$$

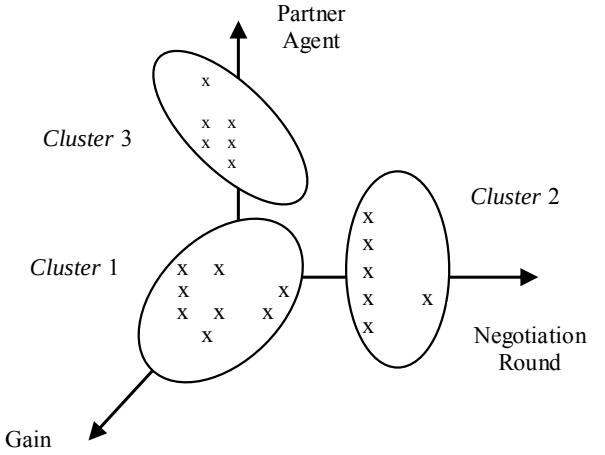

Figure 3. Clustering the states of the agent.

for each object $x_i \in X$, where $D(x)$ is the smallest distance between the point x and a centroid already chosen.

The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early clustering is done. At this point, it is necessary to compute again k new centroids as centers of the clusters resulting from the previous step. After these k new centroids are computed, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop, it is noticed that the k centroids change their location step by step, until no more changes are done. In other words, centroids do not move any more.

Using the k-means algorithm, we have succeeded to group the internal states of an agent into clusters. If the preference coefficients should be updated and improved, the same negotiation must be performed in the same conditions several times. The clusters decrease significantly the number of negotiations performed. Therefore, in order to update the coefficients, a smaller number of negotiations are performed. The time in which the preference coefficients are updated is reduced and the negotiation time is decreased. The coefficients are adjusted in a shorter time, when using clusters, than in the case of individual negotiations.

6. Related Work

The research presented in this article is based on our previous work for developing a multi-agent system framework with agents having adaptive negotiation behaviour, as described in [3]. There, it was put forward a model of self-interested agents acting in an open environment, which capture the most relevant elements of agents' behaviour related to negotiation with other agents. In that approach, agents had simple negotiation profiles, that were used to guide their behaviour. In the present approach, the negotiation profiles are extended, we report new ways for building and updating these profiles and we propose two methods for computing the preference coefficients of the negotiation rules.

Current literature related to negotiation is rich in models of agent negotiation with changing strategies.

In our previous work [1], we have reported on a model of heuristic negotiation between self-interested agents, which allow negotiation over multiple issues of the negotiation object. These comprises different types of negotiation primitives, including argument based ones, and a set of rules to conduct negotiation. In order to negotiate strategically and to adapt negotiation to different partners, the agents use rewards associated to negotiation objects. Also, the notion of regret is used to compare the achieved out-

comes with the best possible results that could have been obtained both in a particular negotiation and in selecting the partner agent. Although used with good results, these previous models did not take into account the specificity of the other agent.

In [8], the authors apply the Q-learning algorithm to analyze and learn customer behaviours and then recommend appropriate products. As compared to our approach, the user profile is not used for negotiation, but to personalise the information to the user interests. The authors use weighting features to recommend products to the user. We use weights to represent the preference coefficients.

In [9], the authors propose a software framework for negotiation, in which the negotiation mechanism is represented by a set of rules, as in our case. The rules are organized in taxonomy, and can be used in conjunction with a simple interaction protocol. Although the rules allow flexible definition of several negotiation strategies, there are no negotiation profiles and the possibility to modify the negotiation, according to these profiles, as in our case.

An implementation of automated negotiation in an e-commerce modeling multi-agent system is described in [10]. A specific set of rules is used for enforcing negotiation mechanisms. An experiment involving multiple English auctions performed in parallel is discussed.

A system for automated agent negotiation, based on a formal and executable approach to capture the behaviour of parties involved in a negotiation is shown in [11]. The negotiation strategies are expressed in a declarative rules language, defeasible logic, and are applied using the implemented system Dr-Device.

In [12], the authors present a system, called GENIUS (General Environment for Negotiation with Intelligent multi-purpose Usage Simulation), that supports the design of different strategies for agent negotiation, and the evaluation of these strategies in a simulated environment. The system allows the negotiation between automated agents, but also between agents and humans. The designer of a strategy can select from a repository a negotiation domain and a preference profile for the agent. As compared to this system, in our approach, there are several negotiation profiles, which are evolved during interactions, and the negotiation domain is specified by the agent rules.

7. Conclusions and Future Work

We have developed an automated negotiation environment, which combines the agents' beliefs about the other agents in the system with the possibility to represent and modify the negotiation strategy. The negotiation strategy is represented in the form of rules with their attached preference coefficients. The preference coefficients are dynamically changed, either using a predefined formula or using a reinforcement learning algorithm.

The system works in open environments, in which there is no previous knowledge about the other agents. Therefore, an agent tries to learn, little by little, during interaction, features characterizing the behaviour of other agents, in a set of partners' profiles. The behaviour of the agent takes into account these profiles and, moreover, the agents can modify their preferences over negotiation results, when they obtain new information.

Two different approaches are proposed for updating the preference coefficients. The paper shows the importance of designing a multi-agent system that can negotiate in a dynamic and efficient manner.

In our current approach, the negotiation is only single issue, but multi-issue attribute negotiation can also be easily accommodated in our model.

A future direction of research is to try to tailor the partners' profiles according to the different attributes of the negotiation object that were negotiated.

Another future direction of research is to evaluate our approach when there are a great number of agents, and determine how the system scales up.

For the time being, the multi-agent system was tested using a limited number of agents negotiating over different negotiation objects, like wood, bricks and window frames, necessary for building a house. A future development is to test the system with many agents working in a real-world environment.

REFERENCES

[1] A. M. Florea and E. Kalisz, "Adaptive Negotiation Based on Rewards and Regret in a Multi-Agent Environment," *IEEE Computer Society Conference Publishing Service*, 2007, pp. 254-259.

[2] S. Radu and A. M. Florea, "An Adaptive Multi-Agent System for e-Commerce," *Proceedings of Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems*, Fiuggi, 8-14 July 2012, pp. 297-300.

[3] S. Radu, E. Kalisz and A. M. Florea, "Agents Negotiation Profiles for Automatic Transactions in Open Environments," *Proceedings of the* 14*th International Symposium on Symbolic and Numerical Algorithms for Scientific Computing*, Timisoara, 26-29 September 2012, in press.

[4] M. M. Mazid, A. S. Ali and K. S. Ticke, "Improved C4.5 Algorithm for Rule Based Classification," *Proceedings of the* 9*th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, Cambridge, 20-22 February 2010, pp. 296-301.

[5] G. Tesauro and J. Kephart, "Pricing in Agent Economies Using Multi-Agent Q-Learning," *Journal of Autonomous Agents and Multi-Agent Systems*, Vol. 5, No. 3, 2002, pp. 289-304. doi:10.1023/A:1015504423309

[6] J. M. Pena, J. A. Lozano and P. Larranaga, "An Empirical Comparison of Four Initialization Methods for the K-Means Algorithm," *Journal Pattern Recognition Letters*, Vol. 20, No. 10, 1999, pp. 1027-1040. doi:10.1016/S0167-8655(99)00069-0

[7] D. Arthur and S. Vassilvitskii, "K-Means ++: The Advantages of Careful Seeding," *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, 7-9 January 2007, pp. 1027-1035.

[8] A. Srivihok and P. Sukonmanee, "E-Commerce Intelligent Agent: Personalization Travel Support Agent Using Q Learning," *Proceedings of the 7th International Conference on Electronic Commerce*, Xi'an, 15-17 August 2005, pp. 287-292. doi:10.1145/1089551.1089606

[9] C. Bartolini, C. Preist and N. R. Jennings, "A Software Framework for Automated Negotiation," *Proceedings of the 3rd International Workshop on Software Engineering for Large-Scale Multi-Agent System*, Laguna Beach, 22-24 September 2005, pp. 213-235. doi:10.1007/978-3-540-31846-0_13

[10] C. Badica, A. Badita and M. Ganzha, "Implementing Rule-Based Mechanisms for Agent-Based Price Negotiation," *Proceedings of the 2006 ACM Symposium on Applied Computing SAC*, Dijon, 23-27 April 2006, pp. 96-100. doi:10.1145/1141277.1141299

[11] T. Skylogiannis, G. Antoniou, N. Bassiliades, G. Governatori and A. Bikalis, "Dr-Negotiate—A System for Automated Agent Negotiation with Defeasible Logic-Based Strategies," *Journal of Data & Knowledge Engineering*, Vol. 63, No. 2, 2007, pp. 362-380. doi:10.1016/j.datak.2007.03.004

[12] R. Lin, S. Kraus, D. Tykhonov, K. Hindriks and C. M. Jonker, "Supporting the Design of General Automated Negotiators," *Proceedings of the Second International Workshop on Agent-based Complex Automated Negotiations*, Budapest, 10-15 May 2009, pp. 1-20.

