A Negotiation Learning Model for Open Multi-Agent Environments

Adina Magda Florea, Eugenia Kalisz

Department of Computer Science
University "Politehnica" of Bucharest
Spl.Independentei 313, Bucharest, Romania
E-mail: adina@cs.pub.ro , ekalisz@cs.pub.ro

Abstract.
The paper presents a model of heuristic negotiation between self-interested agents which allows the use of arguments, negotiation over multiple issues of the negotiation object, single and multi-party negotiation, and learning of the agent's negotiation primitives. The model uses negotiation objects and negotiation frames to separate the object of negotiation from the negotiation process. In order to negotiate strategically, the agents use a reinforcement learning algorithm applied on a specific state space representation of the negotiation process.

Keywords: multi-agent systems, negotiation, negotiation object, learning

1 Introduction

Negotiation is essential in settings where autonomous agents have conflicting interests and a desire to cooperate. Automated negotiation among intelligent agents has thus become increasingly important in applications that require computer supported decision making, like e-commerce, distributed resource allocation, or virtual enterprises. The environments of such applications are inherently open as they are populated with self-interested agents designed and/or owned by different people and there is no complete information about the preferences or decision-making processes of the participating agents. In order to be really autonomous and achieve performance when conducting a negotiation, an agent should be able to anticipate both the outcome of the negotiation and the best potential partner with which to start a negotiation. Machine learning approaches such as reinforcement learning can contribute to adapt the agent's strategy during negotiation and trading, achieve better outcomes and increased payoffs.

In this paper we propose a negotiation framework that includes negotiation objects comprising several aspects of the negotiated item, and different sets of negotiation primitives for cognitive agents and, in particular, BDI agents. In the context of an open environment, a mechanism to learn how to negotiate is needed but learning should take place without prior knowledge of the environment and the agents in it. To this end, we propose a reinforcement learning approach that may permit the negotiator to learn which negotiation primitive to use in a certain state of the negotiation based on a Q-learning rule.

The paper is structured as follows. Section 2 presents the negotiation model of self-interested agents in an open environment, Section 3 presents the negotiation primitives and protocol, Section 4 describes the negotiation learning model and associated representation of the negotiation states, Section 5 deals with related work, while Section 6 is devoted to conclusions and further work.

2 Negotiation Model

The negotiation model we propose comprises a set of self-interested cognitive agents that are able to reason on the different issues regarding the object to be negotiated. A BDI (Belief-Desire-Intention) model of agents [1] is further required in order to support the extended set of primitives defined in Section 3. In a BDI model the agents are endowed with beliefs about the environment and the other agents in the environment, intentions to execute actions structured into plans, and desires, which represent the outcomes the agents want to achieve. A consistent sub-set of desires form the agent goals towards which plans are to be developed.

The agent environment is open, agents being able to enter and leave the environment during their life time. A facilitator is supposed to be present and be aware/informed of agents' identities and abilities. No further details about the facilitator-agents interactions are given but some schemes of these interactions can be found in [2].

The different aspects to be dealt with in a negotiation are grouped into a negotiation object. A negotiation object (NO) is the range of issues over which agreements must be reached, as defined in [3]. The object of negotiation may be: an article that the agent A wants to buy from B; an action that the negotiator agent A asks agent B to perform for it; a service that agent A asks to B, e.g., design a swimming pool; an offer of a service agent A is willing to perform for B provided B agrees to the conditions of A, e.g., a communication company offering to a potential customer a competitive long distance calls service.

A negotiation object has a number of attributes, such as price, deadline or timing, quality, penalties, etc., each attribute having a name, a value, a type, and a flag indicating if the attribute may be modified or not. During negotiation, the values of some attributes may be modified or some extra attributes may be added to the negotiation object, e.g., a number of free minutes for long distance calls in the communication service offered by a company. A negotiation object has thus a number of attributes that can be negotiated, and some others that can not be modified during negotiation. Such a structure for a negotiation object allows capturing a wide range of situations, to properly specify modifications during negotiation, and to estimate the utility of a modified NO during the negotiation. It is supposed that the agents involved in a negotiation have access to a common understanding of the semantics of a NO.

A negotiation frame (NF) specifies the framework for negotiating a particular negotiation object. A negotiation frame contains the name of the frame, the set of negotiation primitives allowed in that framework, the negotiation protocol to be followed, a place holder for the negotiation object and a placeholder for the agent(s) with which it will negotiate - either single party or multi-party negotiation. When wishing to start a negotiation, an agent retrieves a negotiation frame that is most appropriate for negotiating the object, adds the NO to this frame, and adds the list of acquaintances or just one agent to that frame. An agent may have a library of negotiation frames or it may query the facilitator for new negotiation frames. In order to use or to understand a NF, an agent needs to understand the negotiation primitives of the frame, which function as the ontology of the negotiation, and be able to follow the protocol specified in that NF. The negotiation frame models the negotiation process by separating the semantics of the negotiation protocol from the semantics of the negotiation object.

3 Negotiation Primitives

The negotiation primitives we propose in our model may be split in a basic negotiation set and an extended one. The *basic negotiation primitives* comprises a set of primitives that are, in a form or another, quite frequent in heuristic negotiation (according to the classification of negotiation techniques in [3]). These primitives are:

- *Propose NO* - request of a negotiation object
- *Accept NO* - accept the request for the NO
- *Reject NO* - reject the request for the NO
- *ModifReq NO NO'* - modify the request by modifying some values of attributes and/or adding attributes to the NO to obtain NO'.

For example, the negotiator A issues a request for an item, an action, or service to be performed, or service to be offered, the request being directed, let's say, to agent B. The agent B may accept the request, may reject it, and may modify the request by changing the value of an attribute of the NO or by adding a new attribute. Negotiation may continue by performing several consecutive steps in which one or the other agent modifies the NO, a successful contract has been concluded or the negotiation failed.

In case of BDI agents, we extend the negotiation set by a set of negotiation primitives that represent *arguments* that the negotiator, and in some cases the party agent may use during negotiation. Each argument type defines preconditions for its usage. If the preconditions are met, then the agent may use the argument. Among the possible argument types mentioned in the literature [3, 4], we have selected for the *extended negotiation set* the following arguments and associated negotiation primitives:

- *Appeal to past promise* - the negotiator A reminds agent B of a past promise regarding the NO, i.e., agent B has promised to the agent A to perform or offer NO in a previous negotiation. *Preconditions:* A must check if a promise of NO (future reward) was received in the past in a successfully concluded negotiation. *Negotiation primitive: Remember NO*;
- *Promise of a future reward* - the negotiator A promises to do a NO for the other agent A at a future time. *Preconditions:* A must find one desire of agent B for a future time interval, if possible a desire which can be satisfied through an action (service) that A can perform while B can not. *Negotiation primitive: Promise NO*;

- *Appeal to self interest* - the agent A believes that concluding the contract for NO is in the best interest of B and tries to persuade B of this fact. *Preconditions:* A must find (or infer) one of B desires which is satisfied if B has NO (e.g., A believes that customers will want convenient communication services) or, alternatively, A must find another negotiation object NO' that is previously offered on the market (e.g. another communication service) and it believes NO is better than NO'. *Negotiation primitive: CompareD NO Desire* or *CompareO NO NO'*;
- *Threat* - the negotiator makes the threat of refusing doing/offering something to B (e.g. A threatens B that it will interrupt electricity delivery if B does not pay the bill) or threatens that it will do something to contradict B's desires. *Preconditions:* A must find one of B's desires directly fulfilled by a NO that A can offer or A must find an action that is contradictory to what it believes is one of B's desires. *Negotiation primitive: TreatForbid NO* or *ThreatDo NO.*

The negotiation protocol can be intuitively described by using a tree as shown in Figure 1 (drawn for the basic negotiation set), where A is the negotiator and B is the agent with which A negotiates. Nodes in the tree represent states in which one or the other of the negotiating agents have to issue a negotiation primitive: S^A states correspond to the negotiator's decisions, while S^B states to the agent with which A negotiates. Double circled states are terminal states, when the negotiation ends. Transitions from one state to another are labelled with the possible negotiation primitives. The negotiation tree is built from the point of view of the negotiator, i.e., the S^A states are states in which agent A has the control over which primitive to issue, while from S^B states it is B that will respond, therefore A has no control. The described model is similar to a tree in a game of chance, with S^B nodes corresponding to chance nodes.

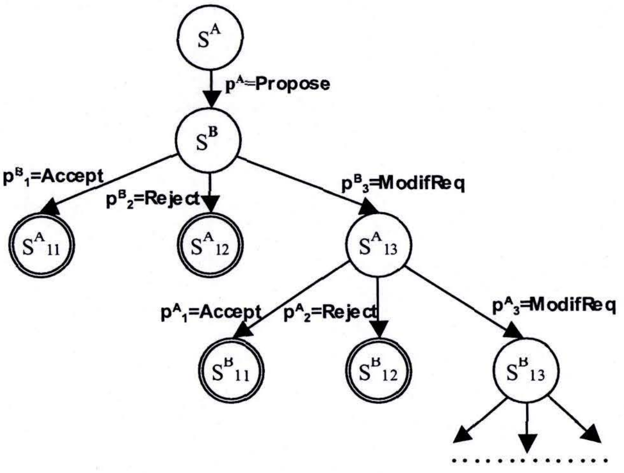

Figure 1: Negotiation tree with alternate S^A and S^B states

An alternate way for the specification of the negotiation protocol is by using Definite Clause Grammar as described in [5] and [6]. This second way of specifying the negotiation protocol is useful for a compact representation of the protocol in the negotiation frame and for a formal executable definition of the negotiation steps possible to be followed by an agent.

4 Negotiation Learning Model

The previous section showed that an agent may have many possibilities/negotiation primitives to conduct a negotiation towards a successful (or unsuccessful) contract. It is the agent decision making model that dictates which is the negotiation strategy to be used [3]. Heuristic strategies are quite powerful in this case but they are domain dependent, rather difficult to figure out, and time consuming. We propose a reinforcement learning approach that may permit the negotiator to learn which negotiation primitive to use in a certain state of the negotiation.

Reinforcement learning is the task faced by an agent that learns adequate behaviour through interactions with a dynamic environment, by reward and punishment, which can be considered the reinforcement signal received by the agent from the environment [7]. In our model, the agent tries to learn an approximation of an optimal policy by using a Q-learning algorithm [8]. A Q-learning algorithm is a model free reinforcement learning in which the agent uses *Q(a,s)* – the value of doing action *a* in state *s*. A policy is a mapping of states to actions that maximizes some long-run measure of reinforcement, in our case the utility of the states in which the agent is faced with a decision regarding the negotiation primitive to be issued. Utilities of states are easily linked to *Q(a,s)* values as $U(s) = \max_a Q(a,s)$. In a Q-learning algorithm we have the following definition of the Q function:

$$Q(s,a) = R(s) + \sum_{s' \in S} T(s,a,s') \max_{a'} Q(s',a') \qquad (1)$$

where S is the state set, A is the set of actions, R is the reward function $R: S \times A \rightarrow \mathrm{R}$, a is the action taken in the current state s, s' is the next state, a' is the action taken in the state s' and $T(s,a,s')$ is state transition function defined over the probability distribution of state transitions, according to a Markov Decision Process (MDP). However, an agent using a Q-learning algorithm must not learn a model of the environment as Q updates can be computed using the Q-learning rule, where α is the learning rate:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(R(s) + \max_{a'} Q(s',a') - Q(s,a)) \qquad (2)$$

The Q-learning rule has an *anticipatory* character, since it uses future states and actions for computing *Q(a,s)*, namely the value of doing action *a* in current state *s*.

In order to apply the Q-learning algorithm to negotiation, a modification of the negotiation tree is performed, as shown in Figure 2. This time, nodes in the tree represent a state which corresponds to two steps of the negotiation process, one step associated to a primitive issued by agent A and another step associated to the answer of agent B. In this way, an S^A_k and a succeeding S^B_k states are merged into the same state. Transitions are labelled this time with a sequence of two negotiation primitives, one issued by agent A in S^A_k and the second corresponding to the answer of B in S^B_k. Let us make the following notation:

$$a_k = (p_i^A < p_{ij}^B >)_k \tag{3}$$

where p^A_i is one of the negotiation primitives allowed in state S^A_k (where *i* ranges over all such primitives) and $<p_{ij}^B>$ is one of the allowed answers of B in state S^B_k when receiving p^A_i. We call a_k a compound action in the negotiation process.

Using this new representation, we can view the state space as an MDP in which agent A would issue a compound negotiation action a_k in state S^A_k and go nondeterministically into one of the next possible states, S^A_{k+1}, for every action a_k having the same p^A_i. Thus, the non-determinism is generated by the lack of knowledge of the exact answer given by agent B for a certain negotiation primitive p^A_i issued by A.

Our Q-learning rule is now obtained by substituting *s* and *a* in equation (2) with the corresponding states and compound actions defined above, as follows:

$$Q(S_k^A, a_k) \leftarrow Q(S_k^A, a_k) + \alpha(R(S_k^A) + \max_{a_{k+1}} Q(S_{k+1}^A, a_{k+1}) - Q(S_k^A, a_k)) \tag{4}$$

When negotiating with one agent, the updates relationships (4) are not applied for *k*=0, the first level in the tree, as this level always corresponds to a proposal. When a multi-party negotiation is considered, the level *k*=0 is also considered in Q-learning, as the agent has to learn to which agent it would best do a certain proposal.

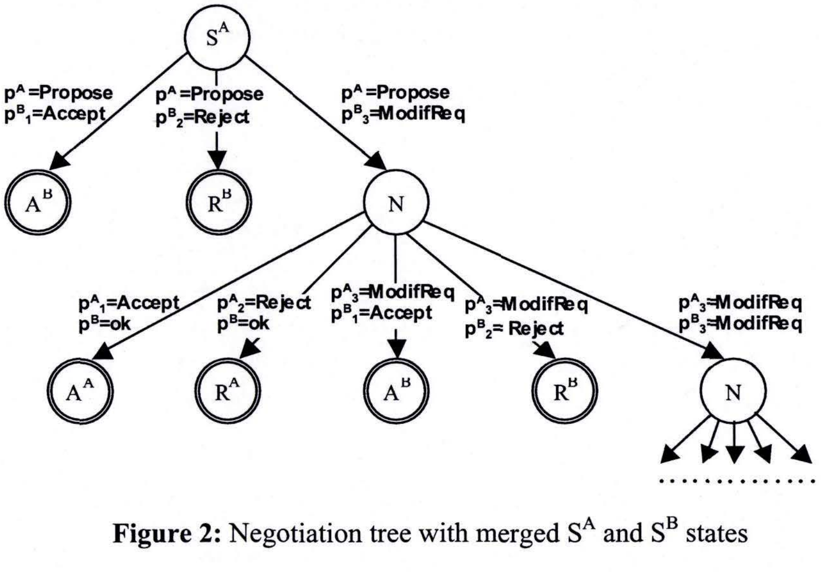
S^A
p^A =Propose p^B_1=Accept
p^A =Propose p^B_2=Reject
p^A =Propose p^B_3=ModifReq
A^B
R^B
N
p^A_1=Accept p^B=ok
p^A_2=Reject p^B=ok
p^A_3=ModifReq p^B_1=Accept
p^A_3=ModifReq p^B_2= Reject
p^A_3=ModifReq p^B_3=ModifReq
A^A
R^A
A^B
R^B
N
....

Figure 2: Negotiation tree with merged S^A and S^B states

In order to apply effectively the Q-learning algorithm, a clustering of states must be designed. In our proposed model we considered the equivalence of states shown in Figure 3 and we penalise several rounds of negotiation by the associated rewards obtained in the corresponding states; a final state of acceptance, for example, will have a lower utility if obtained after several ModifReq than in the case the agreement would have been reached at once. In Figure 3, *SI* is the state in which the negotiation begins (formerly S^A), *SN* is the generic negotiation state (reached by one or several pairs of ModifReq). The number of ModifReq messages exchanged in the negotiation process is either even or odd, depending on the sender of the concluding message – agent A or agent B.

A negotiation is concluded in two situations:

- agent A has received from B an Accept/Reject message, reaching states SA^B or SR^B;
- agent A sends the concluding message (Accept/Reject) and B acknowledges the reception, by means of an ok message, reaching states SA^A or SR^A.

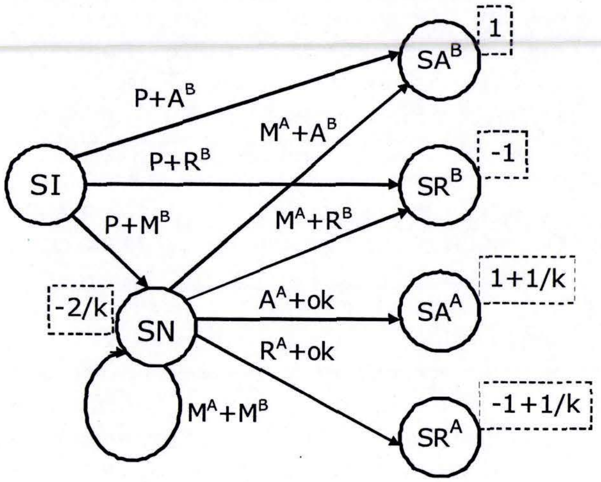

Figure 3: State transitions during the negotiation process.
Primitives: P – Propose, M^x, A^x, R^x – ModifReq/Accept/Reject sent by agent x (A or B), ok – Acknowledge sent by agent B

Each of the two negotiating agents could use a "time-out" policy, suspending the negotiation process after a specific number of iterations. This situation can be considered as a special case of Reject and, consequently, is not treated explicitly. The final negotiation states (SA^B, SA^A – successful, SR^B, SR^A – failed) can be reached either directly from the initial state SI, or after several transitions through the negotiation state SN.

A reward in a final state of acceptance is +1 and in a final state of rejection is -1. Any intermediate state in which a modification of the negotiation object is proposed receives a $1/k$ reward, in which k is assumed to be the maximum number of negotiation rounds. Two successive modifications correspond thus to $-2/k$. The rewards associated to each state are also presented in Figure 3. Table 1 presents the rewards, utilities, and actions to reach a final state from Figure 3.

Table 1: Final state rewards and utilities

Actions to reach a final state	**State**	**State reward**	**State utility**
$\{P, A^B\}$	SA^B	+1	+1
$\{P, R^B\}$	SR^B	-1	-1
$\{P, M^B, A^A, ok\}$	SA^A	+1	$+1 - 1/k$
$\{P, M^B, R^A, ok\}$	SR^A	-1	$-1 - 1/k$
$\{P, M^B, \{M^A, M^B\}^n, A^A, ok\}$	SA^A	+1	$+1 - (2n+1)/k$
$\{P, M^B, \{M^A, M^B\}^n, R^A, ok\}$	SR^A	-1	$-1 - (2n+1)/k$
$\{P, M^B, \{M^A, M^B\}^n, M^A, A^B\}$	SA^B	+1	$+1 - 2(n+1)/k$
$\{P, M^B, \{M^A, M^B\}^n, M^A, R^B\}$	SR^B	-1	$-1 - 2(n+1)/k$

The model in Figure 3 can be extended by considering the extended negotiation set of argumentation primitives as instances of ModifReq, (Appeal to past promise, Promise of a future reward, Appeal to self interest, Threat). In this case, if the negotiation set contains *N* types of messages, then the pairs $\{M^A, M^B\}$ in the negotiation process will have N^2 variants. Although the number of clustered states grows in this case, the Q-learning algorithm can deal effectively with this situation.

For a cognitive agent, the Q-learning process is the component that determines the values that the agent will use in the decision making process to anticipate the best way of action towards a proposed goal. The Q-learning rule has a *strong* anticipatory character, since it uses future states and actions for computing $Q(a,s)$ – the value of doing action *a* in state *s*.

The learning process could represent a first model generating phase in the agent life. When this phase is completed the agent will use the collected data as anticipations in the negotiation process, choosing accordingly the next primitive to issue. Since the behaviour of the other agents is nondeterministic, the next state, reached after the answer of the negotiation partner agent, is not always the one anticipated by the negotiation model. The second, learning free phase, which uses the negotiation model generated in the learning phase, has a *weak* anticipatory character.

5 Related Work

Machine learning and other heuristics were applied to the negotiation problem [10] using techniques such as evolutionary and co-evolutionary computation models, fuzzy logic, graph-theoretic approaches, or reinforcement learning [11]. In [12] the authors present an evolutionary learning approach for designing adaptive negotiation agents. They use a genetic algorithm for deriving potential negotiation solutions, their adaptive negotiation agents adapt to changing behaviours of their opponents by learning about their preferences through their previous counteroffers. In [13] the author adopts asymmetric multiagent reinforcement learning for solving the dynamic pricing problem, by modelling the dynamic pricing problem as a Markov game. The article employs two learning methods: the value function gradient method and the policy gradient method.

In [14], the authors develop a Q-learning algorithm to determine optimal policies in the framework of two players zero sum Markov games. In [15], an extension of this approach is given to general sum games, where the agents first determine a mixed-strategy Nash equilibrium profile for the game and then use this profile in the Q-learning algorithm to determine an optimal policy. In [16] the negotiation is modelled as a set of two non-stationary Markov Decision Processes and a value iteration algorithm is used to learn an optimal policy to negotiation.

As opposed to the related approaches, our model does not need to build a model of the environment and proposes a representation in which negotiation is seen as a single MDP on merged negotiation states. Moreover, our approach may include different types of negotiation primitives and treat them in a uniform manner and can be extended to multi-party negotiation.

6 Conclusions

We have presented a model of a negotiation process that captures a wide variety of possible negotiation situations and objects and which combines heuristic negotiation with argumentation-based one. We have defined a set of negotiation primitives and a negotiation protocol that comprises several possible design choices which may be selected depending on the particular problem domain. The proposed negotiation primitives are combining facilities of modifying the negotiation object with the possibility to specify different types of arguments which involve these very negotiation objects; it is the first such approach according to our knowledge. We have presented structures for specifying the negotiation object and the negotiation framework which allow separating the negotiated object from the protocol of the negotiation.

The agents may be endowed with different negotiation primitives but they must have a decision making process that will allow then to choose the best one at a given moment. We have model the negotiation process as a Markov Decision Process and we have proposed a Q-learning algorithm which uses rewards of merged states in the negotiation state space to learn how to negotiate. The Q-learning algorithm may also be used by the negotiator to choose among several agents in its list of acquaintances the agent with which to negotiate in case of multi-party negotiation.

Using the Q-learning rule, the agents do not need to model the environment or the other agents with which they negotiate; therefore our proposed approach is suited for open environments and on-line learning. Although the agents using a Q-learning approach do not model the environment, they do contain an anticipatory model of themselves from the point of view of the Q-values they compute during learning. These values allow them to choose the supposed right negotiation primitive at a given instance in the process, according to the negotiation model's prediction of what will happen at a latter instance.

Our future works goes towards several directions. First, based on the definition of negotiation object utility presented in [6], we aim to integrate the NO utility in the reward function associated to the negotiation states. In this way, the individual reward will better capture the quality of the modified NO' in a certain moment of the negotiation. Second, we plan to extend the learning mechanism to learn strategies in negotiation, namely to reinforce strategy rules which instruct agents how to negotiate based on different possible arguments. Finally, we plan to obtain extensive experimental results on selected specific problems to evaluate the quality of our learning model.

References

[1] A.S.Rao, M.P.Georgeff, "Modeling Rational Agents Within a BDI-architecture", edited by R.Fikes and E.Sandwall, Morgan Kaufman, Proc. of Knowledge Representation and Reasoning (KR&R-91), 1991, pp. 473-484.

[2] A.Florea, E.Kalisz, "Anticipatory Attributes of Agent Behaviour in MAS", CP627, Computing Anticipatory Systems: CASYS 2001 - Fifth International Conference, edited by D.M.Dubois, American Institute of Physics 0-7354-0081-4/02, p.359-364.

[3] N.R.Jennings, e.a., "Automated negotiation: prospects, methods, and challenges", Int. J. of Group Decision and Negotiation 10 (2), 2001, pp.199-215.

[4] S.Kraus, K.Sycara, A.Evenchik, "Reaching agreements through arumentation: a logical model and implementation", Artificial Intelligence, Elsevier Science, 104, 1998, pp. 1-69.

[5] Y.Labrou, T.Finin, "Semantics and conversations for an agent communication language", In Readings in Agents, M. Huhns and M. Singh, editors, Morgan Kaufmann, San Francisco, 1998, pp.235-242.

[6] A.Florea. "Using Utility Values in Argument-based Negotiation", In Proc. of IC-AI'02, the 2002 International Conference on Artificial Intelligence, Las Vegas, Nevada, USA, 24-27 June 2002, CSREA Press, p.1021-1026.

[7] Kaelbling, L.P., M.L. Littman, and A.W. Moore, "Reinforcement learning: a survey", Journal of Artificial Intelligence Research, 4, 1996, pp. 237-285.

[8] Watkins, C.J. and P. Dayan, "Q-learning", Machine Learning, 8(3), 1992, pp. 279-292.

[9] D.M.Dubois, "Review of Incursive, Hyperincursive and Anticipatory Systems – Foundation of Anticipation in Electromagnetism," in Computing Anticipatory Systems: CASYS'99 – Third International Conference, edited by Daniel M. Dubois, AIP Conference Proceedings 517, American Institute of Physics, Melville, NY, 2000, pp. 3-30.

[10] D.Zeng, K.Sycara, "Benefits of learning in negotiation", in Proc of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI- 97/IAAI-97), AAAI Press, 1997, pp. 36-42.

[11] Y.Shoham, R.Powers, T.Grenager, "Multi-agent reinforcement learning: a critical survey", Technical report, Stanford University, 2003.

[12] R.Y.K.Lau, e.a., "An evolutionary learning approach for adaptive negotiation agents", International Journal of Intelligent Systems, 21(1), 2006, pp.41-72.

[13] V.Könönen, "Dynamic pricing based on asymmetric multiagent reinforcement learning", International Journal of Intelligent Systems, 21(1), 2006, pp.73-98.

[14] M.L.Littman. Markov games as a framework for multi-agent reinforcement learning. Proc. of the 11th Int. Conference on Machine Learning (ML-94), 1994.

[15] J.Hu, M.P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. Proc. of the 15th Int. Conf. on Machine Learning (ML-98), 1998.

[16] V.Narayanan, N.R.Jennings, "An adaptive bilateral negotiation model for e-commerce settings", in 7th Int. IEEE Conf. on E-Commerce Technology, Munich, Germany, 2005, pp. 34-39.