Game Theory-based Data Mining Technique for Strategy Making of a Soccer Simulation Coach Agent

Amin Milani Fard[1], Vahid Salmani[1], Mahmoud Naghibzadeh[1],
Sedigheh Khajouie Nejad[2], Hamed Ahmadi[3]

[1] Computer Engineering Dept., Ferdowsi University of Mashhad, Iran
[2] Computer Engineering Dept., Islamic Azad University of Mashhad, Iran
[3] Computer System Software Dept., National Aerospace University of Kharkiv, Ukraine

milanifard@ieee.org, salmani@ieee.org, naghib@ferdowsi.um.ac.ir,
khajouie_nejad@mshdiau.ac.ir, ha_ah65@stu-mail.um.ac.ir

Abstract: Soccer simulation is an effort to motivate researchers to perform artificial and robotic intelligence investigations in a multi-agent system framework. In this paper, we propose a game theoric-based data mining approach to help the coach agent select the best strategy for each soccer player agent in order to gain the most probable payoffs.These payoffs are calculated both static and dynamic i.e. are taken from experience results that are stored in a knowledge-base or is learned knowledge during the game. In this work we have confined ourselves to a model in which opponent strategy remains static. We take advantage of a learning algorithm with a polynomial time complexity in the number of states of the opponent strategy modeled by deterministic finite automata.

1 Introduction

Robotic soccer is a particularly good domain for studying multi-agent systems and has been gaining popularity in recent years with international competitions like RoboCup [Ki97]. Robotic soccer can be used as a standard testbed to evaluate different multi-agent system techniques in a well-defined manner. Noda's *Soccer Server* [No96], pictured in Fig. 1 captures enough real-world complexities to be an indispensable tool for years in RoboCup competitions. Fig. 2 shows the format of the communication message between the server and a specific client. Since the client's vision is limited to 45^o on either side, not all objects are visible at each sensory step. For example, at the beginning of the trace in Fig. 2 the client sees two teammates (Nexus) and one opponent player (Aria), however, after dashing once, it is no longer able to see the opponent. In this work we perform our research approach in the mentioned platform of soccer simulation.

The structure of the paper is as follows. Section 2 describes an overview of some previous researches and also data mining approaches for strategy making. An investigation on game theory approach is declared in section 3, section 4 is dedicated to skills hierarchy and strategy layer, learning in multi-agent systems, and finally section 5 presents our method.

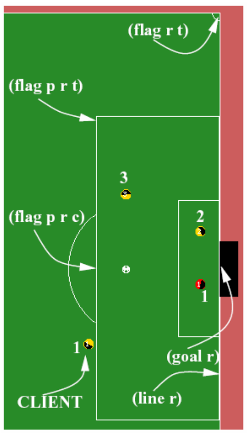

Fig. 1. A partial view of simulated environment

```
... (see 124 ((goal r) 20.1 34) ((flag r t)
47.5 -4) ((flag p r t) 30.3 -24) ((flag p r c)
10.1 -20)((ball) 11 0) ((player Nexus 2) 21 19)
((player Nexus 3) 21 -11) ((player Aria 1) 17
35) ((line r) 40 -26))**-> (dash 80) (see 129
((goal r) 16 43) ((flag r t) 42 -6) ((flag p r
t) 25 -30) ((flag p r c) 5 -40) ((ball) 6
1)((player Nexus 2) 16.3 24) ((player Nexus 3)
15.3 -17) ((line r) 32.8 -27))**-> (turn 1)**->
(dash 60) (see 134 ((flag r t) 40 -9) ((flag p
r t) 23.3 -35) ((ball) 3.7 2) ((player Nexus 2)
14.4 24)((player Nexus 3) 13.3 -22) ((line r)
28.2 -30)) **-> (turn 2)**-> (dash 30) (hear
138 18 shoot the ball)(see 139 ((flag r t) 38.1
-11) ((flag p r t) 22 -39) ((ball) 1.9 0)
((player Nexus 2) 12.8 27)((player Nexus 3)
11.6 -27) ((line r) 25.5 -31)) **-> (say
shooting now) **-> (kick 53 51) (hear 141 self
shooting now) ...
```

Fig. 2. A trace of the simulator's input and output. The player moves to the ball and then shoots it towards the goal. Commands from the player are indicated with "**-> " preceding them.

2 Related Works

Data mining (DM), also known as Knowledge-Discovery in Databases (KDD), is the process of automatically searching large volumes of data for patterns such as association rules. DM has been defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" [Fr92] and "the science of extracting useful information from large data sets or databases" [Ha01]. A data mining process in the field of RoboCup soccer simulation involves gathering useful information out of the game data and acquires useful knowledge about the game situation known as game pattern or strategy.

In competitive domains such as soccer games, any knowledge about the opponent may help players design a strategy in order to win the game. This idea proposes an approach for strategy making based on the observation of their input-output behaviors using a classification task [Al00]. An extension of this approach to the RoboCup was presented in [Le02]. The behavior of a player in the robosoccer can be understood in terms of its inputs (sensors readings) and outputs (actions).

Tactics, formations, positioning, and player types, are common concepts in soccer. CMUnited team brought the concepts of formation and positioning to RoboSoccer [St99, St98] and used dynamic switching of formations depending on the result and time of the game. FC Portugal [Lu00] extended this concept and introduced tactics, situations and player types. FC Portugal team strategy definition is based on a set of player types (that define player strategy, ball possession and ball recovery behaviors) and a set of tactics that include several formations (4-3-3, 4-4-2, Open 4-3-3, 3-4-4, 5-3-2, etc.).

Situation based strategic positioning mechanism [Lu00] is used for strategic situations. For active situations, the agent position on the pitch is defined by specific ball possession, ball recovery or stopped game decision mechanisms. The agent analyses which is the tactic and formation in use and its positioning (and corresponding player type) and calculates its base strategic position in the field. This position is then adjusted with regard to the ball position and velocity, situation (attack, defense, scoring opportunity, etc.) and player type strategic information. Player strategic characteristics include admissible regions in the field, ball attraction, specific positional characteristics for some regions in the field, alignment in the offside line, tendency to stay behind the ball, and attraction by specific points in the field.

Dynamic positioning and role exchange is based on previous work from Peter Stone et al. [St99, St98] that suggested the use of flexible agent roles with protocols for switching among them. In FCPortugal team, players may exchange not only their positioning but also their player types in the current formation. Positioning exchanges are performed only if the utility of that exchange is positive for the team. Utilities are calculated using the distances from the player's positions to their strategic positions. Positioning importance depends on the formation and ball position.

3 Learning in multi-agent systems

Multi-agent systems learning techniques mainly involve machine learning algorithms, game theory, utility theory, and complex systems. [Mi97]. That is, we assume the existence of a large, sometimes infinite, set of examples *E*. Each example $e \in E$ is a pair $e = \{a, b\}$ where $a \in A$ represents the input the agent receives and $b \in B$ is the output the agent should produce when receiving this input. The agent must find a function f which maps $A \rightarrow B$ for as many examples of A as possible. The set E is usually first divided into a training set which is used for training the agent, and a testing set which is used for testing the performance of the agent. In a multi-agent scenario the agent is no longer learning to extrapolate from the examples it has seen of fixed set E, instead its changing target concept makes a moving target function problem [Vi98].

Game theory provides us with the mathematical tools to understand the possible strategies that utility-maximizing agents might use when making a choice. It is mostly concerned with modeling the decision process of rational humans, a fact that should be kept in mind as we consider its applicability to multi-agent systems. The simplest type of game considered in game theory is the *single-shot simultaneous-move* game. In this game all agents must take one action. All actions are effectively simultaneous. Each agent receives a utility that is a function of the combined set of actions. A single-shot game is a good model for the types of situations often faced by agents in a multi-agent system where the encounters mostly require coordination [Vi03].

In the one-shot simultaneous-move game we say that each agent *i* chooses a strategy $s_i \in S_i$, where Si is the set of all strategies for agent *i*. These strategies represent the actions the agent can take. When we say that *i* chooses strategy s_i we mean that it chooses to take action s_i. The set of all strategies chosen by all the agents is the strategy profile for that game and it is denoted by $s \in S \equiv \times_{i=1}^{I} S_i$. Once all the agents make their choices and form the strategy profile s then each agent *i* receives a utility which is given by the function $u_i(s)$. Notice that a player's utility depends on the choices made by all the agents. A Game [Ne85] is a 3-tuple $G = \{N, \alpha, \Pi\}$ Where:

- N is the number of players,
- $\alpha = \{\alpha^i\}_{i=1\ldots N}$; $\alpha^i = \{\alpha^i_1, \ldots \alpha^i_{Mi}\}$ is the set of actions available to player *i*, and
- Π: $\times_i\ \alpha^i \rightarrow R^N$ is the Payoff function, i.e. Π assigns each player a real number payoff for any combination of all players actions.

Two player games involve only two players, *i* and *j*. They are often represented using a game matrix such as the one shown in Fig. 4. In that matrix we see that if agent 1 (the one who chooses from the rows) chooses action A and agent 2 chooses action B then agent 1 will receive a utility of 3 while agent 2 receives a utility of 4.

	A	B
A	1,2	3,4
B	3,2	2,1

Fig. 4. A sample two player game matrix

It is possible that a player will choose randomly between its action choices, using different prior probabilities for each choice. These types of strategies are called mixed strategies and they are a probability distribution over an agent's actions. We say that a mixed strategy for agent *i* is $\sigma_i \in \Sigma_i \equiv P(S_i)$ where $P(S_i)$ is the set of all probability distributions over the set of pure strategies S_i [Vi03]. The *Nash equilibrium* in an n-player game is a set of strategies, $\Sigma = \{\sigma^1, \ldots, \sigma^n\}$; such that, given that for all I player *i* plays σ^i, no player *j* can get a higher payoff by playing a strategy other then σ^j. [Mo96] It has been shown that every game has at least one Nash equilibrium, as long as mixed strategies are allowed. If the system is in equilibrium then no agent will be tempted to take a different action.

In a 2 player game, consider player *A* chooses a strategy and plays by it. Player *B* tries to learn *A*'s strategy and design his strategy as a best response to it. We assume *A* restricts itself to strategies realizable by *Deterministic Finite State Automata* (DFA). This is due to DFS strategies have been accepted widely as a model of bounded rationality [Ru85, Ne85], and also learning the structure of an automaton has been shown to be a very hard problem [Ke94].

The number of states in the automata is considered as a measure of their complexity. A series of "folk theorems" have shown that if the players are restricted to automata of size sub-exponential in the game length (i.e. the number of rounds or in our paper number of cycles in a simulated soccer game) then cooperative behavior can be achieved at equilibrium. Consider two players, A, and B, playing this game. Each player's strategy, σ^i, $i \in \{A, B\}$, is a sequence of actions taken by player *i*. Strategy *i* can be represented by a DFA where *i*'s actions are given in every state of the automaton and the transitions are determined by the actions taken by *i*'s opponent. For example if the automaton in Fig. 5 represents *A*'s strategy in the famous *Prisoner's Dilemma* (PD) then, both players will stay in the initial state if both perform *cooperate*. *A* will move to the other state if he performs *cooperate* and B performs *defect* [Mo96]. This shows that in the repeated PD game, if all players are rational, the only equilibrium is mutual defection. However Neyman [Ne85] and Rubinstein [Ru85] have shown that even if only one player is restricted to an Automaton with a limited number of states, any payoff pair in the Individually Rational Region (Fig. 6) can be accomplished as an equilibrium payoff.

	D	C
D	1,1	0,4
C	4,0	3,3

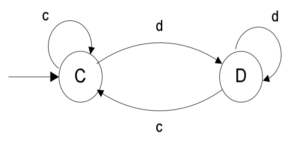

Fig. 5. A's strategy – example

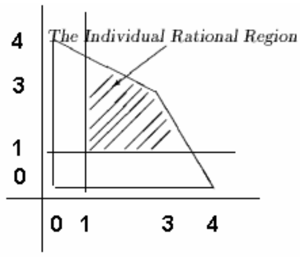

Fig. 6. Payoffs of the PD game

We denote the automaton that represent *i*'s strategy ($i \in \{A, B\}$), and have *n* states by A^n_i. When playing against an automaton, the game history is eventually cyclic. If player A is an automaton, and B is indeed trying to maximize his payoff, it is enough for him to consider only simple cycle in A i.e. cycle in which every state is passed only once. Thus, when considering the possible payoffs included by an automaton, it is sufficient to examine its simple cycles [Mo96].

Four types of states in an automaton for the PD game were defined. We will group the states of the automaton into chunks of connected (in the automaton graph) states of the same type. Let N_A be the number of states in *A* automaton, and NC_A be the number of chunks of equi-type states. Let the complexity relation between two automata denotes by $A <_c A'$ if $N_A < N_{A'}$, or $N_A = N_{A'}$ & $NC_A < NC_{A'}$. The class of simple automata is defined as: $C_{simp} \equiv A : \exists <\alpha,\beta>$ *s.t.* *A* supports $<\alpha,\beta>$ and $\forall A'$: *A'* supports $<\alpha,\beta> \Rightarrow A <_c A'$.

Mor et al., in [Mo96] proved that C_{simp} class is learnable by a polynomial time algorithm of O(n) to construct the appropriate automaton against learned strategy. Suppose player *A* designs an automaton that is "tuned" towards a certain payoff vector, and player *B* tries to learn that automaton and play accordingly. It is reasonable that *A* will choose an automaton that gives *B* a payoff of 1 (or $1+\varepsilon$) to maximize his own payoff. However, we might want to allow more complex situations, emerging from various possible beliefs of the players. Consider, for instance a setting in which *B* can opt out of the game, and be matched with a different partner. If both players believe *B* can receive an expected payoff of θ if he opts out, then *A* will construct his automaton such that award *B* at least θ in equilibrium. Let us assume that *A* restricts himself to strategies that grant *B* a payoff of at least β at equilibrium. Still, among all these strategies, *A* will choose that which maximizes his own payoff. Consider again the prisoner's dilemma. Given that *A* maximizes his payoff for a certain minimal payoff he attributes to *B*, the only possible payoffs to be received by both players can be represented in the upper and rightmost boundaries.

An example of a polynomial learning algorithm for PD problem was explained in [Mo96]. Assume player *B* knows that *A*'s automaton is simple, i.e., *A*'s automaton is in *Csimp*, the learning algorithm for *B* is shown in Fig. 8. Notice that *B* does not know how many states there are in *A*'s automaton (n). In the algorithm *Learn0-3(n)*, *B* could have played C all the time in order to play according to *A*'s automaton (a chunk of states 0 connected to a chunk of states 4). But, if *B* would have played so, *A* could have taken advantage of that and play D forever. Hence a step where *B* will play D to prevent *A* from abusing him was added in *B*'s learning algorithm. *B* will discover the size of *n* in polynomial time since he will know it after $\log_2$ number of steps [Ne85]. K_p denotes the number of states in A^n_i in which player *A* gets payoff *p*.

4 Skills hierarchy and strategy layer

Soccer agents' skills includes turning towards a point, kicking the ball to a desired position, dribbling, intercepting the ball, marking opponents, etc. These skills can be divided into different layers which together form a hierarchy of skills. Fig. 8 shows this hierarchy which consists of three layers which skills of each layer use skills from the layer below to generate the desired behavior. The bottom layer contains low-level player skills which can be directly specified in terms of basic action commands known to the soccer server. At this abstraction level the skills correspond to simple actions such as turning towards a point. The middle layer contains intermediate skills which are based on low-level skills. The skills in this layer do not deal with the exact format of server messages anymore but can be specified in terms of the skills from the layer below. Finally, the skills at the highest level are based on intermediate skills. The way in which a skill is executed depends on the arguments which are supplied to it.

```
Learn(n):
        Play C
        If payoff = 0 then Learn0-3(n)
        else Learn3-4(n)

Learn0-3(n):
        For i=1 to $\infty$ {
                Play C for $2^i$ times
                Play D for $4^i$ times
        }

Learn3-4(n):
        $K_3$=0
        Repeat {
                Play C for $K_3$ times
                Play D
                Play D
                If payoff = 1 then $K_3 \leftarrow K_3 + 1$
                else braek
        }
        $K_S = 0$
        While (payoff = 4) {
                Play D
                $K_0 \leftarrow K_0 + 1$
        }
        Repeat {
                Play C for $K_3$ times
                Play D for $K_0$ times
        }
```

Fig. 7. The PD Learning algorithm

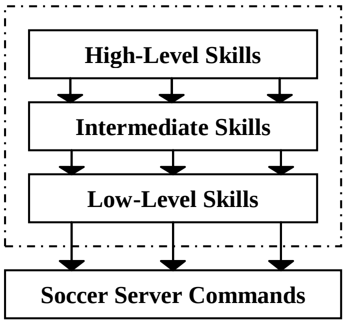

Fig. 8: The skills hierarchy consisting of three layers.

Which skill is to be selected in a certain situation depends on the team strategy. It is important to realize that the execution of each skill eventually leads to a basic soccer server action command. In this model a player that starts with the ball dribbles until it sees an opponent at a predetermined distance. However, a more flexible and powerful approach would include allowing the dribbling player to learn at the time of dribbling, passing, and shooting.

Having introduced adversarial behaviors, some additional issues must be considered. First, if the adversaries are permitted to continually adjust to each other, they may evolve increasingly complex behaviors with. This potential stumbling block in *competitive co-evolution* has been investigated by several researches with genetic algorithms [Gr96, Ha96, Ro95]. Second, since a robotic soccer team is supposed to play against many different opponents, often for only a single match, it must be able to adapt quickly to opponent behaviors.

5 The proposed method

Nexus soccer simulation team of Ferdowsi University of Mashhad [Sal05] is based on a layered architecture. This kind of architecture allows each layer to get information from the lower layer, and provides higher level services for the upper layer. The Strategy layer shown in Fig. 9, is a high level layer that uses World Model information and basic skills to play a soccer game. This layer is actually designed for the coach agent; however, other players can also use this layer to cooperate with other players.

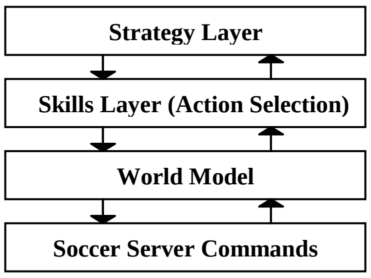

Fig. 9. Team architecture

Coach agent in soccer simulation environment is a privileged client used to provide assistance to the players [Ch02]. There are two kinds of coaches, the *online coach* and the *trainer*. The trainer can exercise more control over the game and may be used only in the development stage, whereas the online coach connects to server during the game and provides additional advice and information to the players. The coach agent can control the play-mode, broadcast audio messages containing information, and getting noise-free information about the movable objects. The online coach is thus a good tool for opponent modeling, game analysis, and giving strategic tips to its team mates.

In our proposed model, the coach agent constructs a knowledge-base of the game in the main memory containing 11 game matrixes for each 11 soccer player agents and assumes opponent's strategy realizable by a DFA. The number of states in that DFA is a complexity measure. The coach would then apply the polynomial time learning algorithm of O(n) in which n is the number of states of the opponent automaton for all 11 game matrixes with respect to the payoffs assigned by the game knowledge-base as shown in Fig. 10. A team strategy is mostly made using a knowledge-base or a set of <*state*, *action*> pairs. Using a special formation is another way in which each player has some predefined duties. These predefined duties are divided into static and dynamic.

	Intercept	Outplay	Pass	Shoot	Dribble	...
Intercept	1,1	0,3	0,4	0,4	0,3	...
Outplay	3,0	0,2	3,0	3,0	2,1	...
Pass	4,0	0,3	0,1	0,2	0,2	
Shoot	4,0	0,3	0,1	0,1	0,2	
Dribble	3,0	1,2	0,2	0,2	0,1	
...	...	...	...	...	...	...

Fig. 10. A sample agent game matrix

The static duties are knowledge-base of the strategy, with set of rules which are generally based on the real soccer rules and ideas and also the environment of the server. On the other hand the dynamic duties are where the strategy decisions are made with respect to the dynamic parameters such as game status, opponent's behavior and some statistics of the game.

A team's success is directly influenced by each agent’s actions. To determine team’s efficiency, average results within 10 matches of three teams were set up accordingly. Nexus2005 [Sa06] was a fuzzy improvement of Nexus2003 [Sa05] action section mechanism and Nexus2006 take advantage of a probabilistic action evaluation method.

Table. 1. Results of competition within 10 matches

Games	Average within 10 games
StrategicNexus vs. Nexus2006	1.7 - 1.6
StrategicNexus vs. Nexus2005	1.9 - 1.6
StrategicNexus vs. Nexus2003	2.6 – 1.4

6 Conclusion and future work

In this paper an effective learning mechanism was introduced in order to be used on the strategy making method of the coach agent in a polynomial time. The coach agent creates a set of 11 game matrixes for each 11 soccer player agents and then applies the learning mechanism algorithm with respect to the payoffs. These payoffs are calculated both static and dynamic i.e. are taken from an experience in the knowledge-base or an acquired knowledge during the game. In this work we confined ourselves to a model in which opponent strategy remains static. A more general model in which players have to learn non-fixed strategies by mutual learning will be considered in our future work.

References

[Al00] Ricardo Aler, Daniel Borrajo, In´es Galv´an, and Agapito Ledezma. Learning models of other agents. In Proceedings of the Agents-00/ECML-00 Workshop on Learning Agents,, pages 1–5, Barcelona, Spain, June 2000.

[Ch02] M. Chen, E. Foroughi,, Robocup Soccer Server manual 7.07, August, 2002. RoboCup Federation. available: http://sserver.sourceforge.net.

[Fr92] W. Frawley and G. Piatetsky-Shapiro and C. Matheus, Knowledge Discovery in Databases: An Overview. AI Magazine, Fall 1992, pp. 213-228

[Gr96] John Grefenstette and Robert Daley. Methods for competitive and cooperative co-evolution. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 45–50, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS-96-01.

[Ha01] D. Hand, H. Mannila, P. Smyth: Principles of Data Mining. MIT Press, Cambridge, MA, 2001. ISBN 0-262-08290-X

[Ha96] Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In Gerhard Weißand Sandip Sen, editors, *Adaptation and Learning in Multiagent Systems*, pages 113–126. Springer Verlag, Berlin, 1996.

[Ke94] Michael J. Kearns and Umesh V. Vazirani. An Introduction to Computational Learning Theory. MIT press, Cambridge, Massachusetts, 1994.

[Ki97] Hiroaki Kitano, Yasuo Kuniyoshi, Itsuki Noda, Minoru Asada, Hitoshi Matsubara, and Eiichi Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.

[Le02] Agapito Ledezma, Ricardo Aler, Araceli Sanchis, and Daniel Borrajo. Predicting opponent actions in the robosoccer. In Proceedings of the 2002 IEEE International Conference on Systems, Man and Cybernetics, October 2002.

[Lu00] Luís Paulo Reis, Nuno Lau, *FC Portugal Team Description: RoboCup 2000 Simulation League Champion*

[Mi97] Mitchell, T.M.: Machine Learning. McGraw Hill (1997)

[Mo96] Yishay Mor, Claudia V. Goldman, Jeffrey S. Rosenschein, Learn your Opponent's strategy (in Polynomial Time)!, *Adaptation and Learning in Multi-Agent Systems, IJCAI95 Workshop, Montreal, Canada, August 1995, Proceedings. Lecture Notes in Artificial Intelligence Vol. 1042*, G. Weiss and S. Sen (eds.) Springer Verlag, 1996.

[Ne85] A. Neyman. Bounded complexity justifies cooperation in finitely repeated prisoner's dilemma. Economic Letters, pages 227-229, 1985.

[No96] Itsuki Noda and Hitoshi Matsubara. Soccer server and researches on multi-agent systems. In *Proceedings of the IROS-96 Workshop on RoboCup*, November 1996.

[Ro95] Christopher D. Rosin and Richard K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In Stephanie Forrest, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380, San Mateo,CA, July 1995. Morgan Kaufman.

[Ru85] A. Rubinstein. Finite automata play the repeated prisoner's dilemma. ST/ICERD Discussion Paper 85/109, London School of Economics, 1985.

[Sa05] Salmani V., Naghibzadeh M., Seifi F., Taherinia A., “A Two-Phase Mechanism for Agent's Action Selection in Soccer Simulation”, The Second World Enformatika Conference, WEC'05, Istanbul, Turkey, pp. 217-220, February 2005.

[Sa06] Salmani V., Milani Fard A., Naghibzadeh M., "A Fuzzy Two-Phase Decision Making Approach for a Soccer Simulation Agent", The 1st IEEE International Conference on Engineering of Intelligent Systems, April 22-23 2006, Islamabad, Pakistan.

[Sal05] Salmani V., Seifi F., Moienzadeh H., Milani Fard A., Naghibzadeh M., *"Nexus 3D 2005 Team Description"*, RoboCup Soccer simulation team description, International Symposium of RoboCup, 2005, Osaka, Japan

[St98] Peter Stone. *Layered Learning in Multi-Agent Systems*. PhD Thesis, Computer Science Dep., Carnegie Mellon University, December 1998

[St99] Peter Stone and Manuela Veloso. Task Decomposition, Dynamic Role Assignment, and Low-Bandwidth Communication for RealTime Strategic Teamwork. *Artificial Intelligence*, 110 (2), pp. 241-273, June 1999.

[Vi03] Jos´e M. Vidal, Learning in Multiagent Systems: An Introduction from a Game-Theoretic Perspective In Eduardo Alonso, editor, Adaptive Agents: LNAI 2636. Springer Verlag, 2003.

[Vi98] Vidal, J.M., Durfee, E.H.: The moving target function problem in multi-agent learning. In: Proceedings of the Third International Conference on Multi-Agent Systems. (1998)