We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

4

Learning by Experience and by Imitation in Multi-Robot Systems

Dennis Barrios-Aranibar, Luiz M. G. Gonçalves and Pablo Javier Alsina
Universidade Federal do Rio Grande do Norte
Brazil

1. Introduction

With the increasing number of robots in industrial environments, scientists/technologists were often faced with issues on cooperation, coordination and collaboration among different robots and their self governance in the work space. This has led to the development of systems with several cooperative robotic agents. (Kim et al., 1997b). Generally, a system with several robotic agents (multi-robot system) is composed by two or more robots executing a task in a cooperative way (Arai and Ota, 1992).

Coordination, collaboration and cooperation are three terms used without distinction when working with multi-agent and multi-robot systems. In this work, we adopt a definition proposed by Noreils (Noreils, 1993) in which cooperation occurs when several agents (or robots) are gathered together so as to perform a global task. Coordination and collaboration are two forms of cooperation (Botelho and Alami, 2000). Coordination occurs when an entity coordinates its activity with another, or it synchronizes its action with respect to the other entity, by exchanging information, signals, etc. And, collaboration occurs when two or more agents decompose a global task in subtasks to be performed by each specific agent.

Generally, the solution for problems using multi-agent and multi-robot systems is divided into stages. When talking about autonomous robots, two of these stages are the task allocation stage and the task execution stage. Task allocation should be done so that all components (agents or robots) in the system are used and the problem is completely solved. The task execution stage itself should be performed so that the agents do not interfere to each other (coordination and/or collaboration) when solving the problem. Traditionally, both stages are carried out independently. In the task allocation stage, it is defined if the agents will collaborate to each other or if they will coordinate their activities. In the task execution stage, collaboration and/or coordination are effectively done.

In the literature, each stage is implemented using different techniques. The task allocation stage can be implemented using centralized or decentralized approaches (Le Pape, 1990). Centralized approaches can be implemented as an optimization problem. Decentralized approaches generally use marked based approaches like the contract-net protocol (CNP) (Ulam et al., 2007) or other approaches derived from it (Botelho and Alami, 1999).

The task execution stage can be implemented in many ways. It depends of the nature of interactions between agents (Collaboration or coordination) and if agents can modify or not their strategies (Static and dynamic strategies). For example it can be implemented using

probabilistic methods (Bruce et al., 2003), fussy logic (Wu and Lee, 2004; Chen and Liu, 2002), reinforcement learning (Salustowicz et al., 1998; Wiering et al., 1999), evolutionary algorithms (Zhang and Mackworth, 2002), neural networks (Kim et al. 1997a) and others.

A static strategy is defined during the design of the robotic system and after that, it is applied to the robots by choosing roles and actions of each robot depending of the a priori defined situation. A disadvantage of this kind of strategy is that it doesn't adapt automatically to changes in requirements and can lead to a low performance of the system if situations not envisaged by the designers occur.

On the other hand, a dynamic strategy adapts itself to the environment. This kind of strategy is generally implemented with artificial intelligence techniques. Dynamic strategies can be divided in two stages: the learning and the using stage. In the learning stage, the overall system is exposed to simulated environments, where, if necessary, opponents are programmed using static strategies. In the using stage, the system does not modify the strategy parameters. Both stages can be executed one after another during all useful life of the system.

This traditional way of implementing dynamic strategies requires a predefined set of possible situations and actions. Thus, because robots already know the kind of situations they can find in the environment and also they know what actions they can execute, we denominate these traditional approaches as "learning by experience".

We conjecture that, in real world and not in minimal applications, the robots can not completely know what kind of situations they can find and also what are all the actions that they can perform, in this case, consider an action as a set of consecutive low level signals to the actuators of the robots. In this sense, we propose to combine the imitation learning approach with learning by experience in order to construct robots that really adapt to environment changes and also evolve during their useful life. Imitation learning can help the robots to know new actions and situations where it can be applied and learning by experience can help robots to test the new actions in order to establish if they really work for the whole team.

It is important to note that the concepts, algorithms, and techniques proposed and evaluated in this work are focused in the task execution stage, specifically in dynamical strategies. Also, all the concepts are valid for multi-robot and multi-agent systems. The algorithms traditionally used for implementing learning by experience approaches are explained in section 2. Between them, we choose reinforcement learning algorithms for testing our model. In section 3 we explain our approach for implementing imitation learning and in section 4 we explain how the overall process of learning is implemented. Because there are several ways or paradigms for applying reinforcement learning to multi-robot systems , we compare them in section 5. Also, results obtained when the overall process was applied to a robot soccer problem are discussed in section 6. Finally conclusions of this work are explained in section 7.

2. Learning by Experience

The idea of using agents that can learn to solve problems became popular in the area of Artificial Intelligence, specifically in applications for machine learning techniques. When working with multi-agent or multi-robot systems, learning can be implemented in the two stages, as explained in the previous section. Thus, by developing agents that learn task allocation for the first stage and agents that learn to solve their task in the second stage, we are contributing significantly to this field.

We implement dynamic strategies, using machine learning techniques. These strategies require a predefined set of possible situations and actions. Thus, because robots already know the kind of situations they can find in the environment and also know what actions they can execute, we name those traditional approaches as learning by experience approaches. Learning by experience occurs when a robot or agent modifies the parameters used for choosing actions, from a set of them, to be executed in a set of known situations. Thus, in learning by experience, robots only modify the strategy using a set of actions prior defined by the system designer.

Almost all works found in the literature deal with robots learning by experience (Bruce et al., 2003; Wu and Lee, 2004; Chen and Liu, 2002; Wiering et al., 1999; Salustowicz et al., 1998; Zhang and Mackworth, 2002; Kim et al., 1997a).

Interactions in multi-agent and multi-robot systems can be collaborative, competitive or mixed. At this point, concepts of artificial intelligence and game theory come together to be able to explain those interactions and find ways to optimize them. In game theory, interactions between agents or robots are modeled as stochastic games. When the system is composed by a unique agent, the stochastic game is a Markov Decision Process. When the system is composed by several agents but with a unique state, it is a Repetitive Game. (Shoham et al., 2007). Stochastic games are also known as markov games and are classified in zero sum games, where agents are fully competitive, and in general sum games, where agents cooperate and/or compete to each other (Filar and Vrieze, 1997). Here, we address only those learning algorithms applied to general sum stochastic games.

2.1 Common Algorithms

Several machine learning algorithms are applied to learning in multi-agent and multi-robot cooperative systems (general sum stochastic games). In machine learning, the three most common paradigms for learning are supervised learning, unsupervised learning and the reward based learning (Panait and Luke, 2005). It is well known that these models are not easily applied for learning when using multi-agent and multi-robot systems mainly because of the complexity of these systems.

2.1.1 Supervised Learning

Supervised learning algorithms work with the assumption that it is possible to indicate the best response directly to the agent. But, when working with multi-agent and multi-robot systems, it is very difficult to indicate the best response for each agent or robot. This occurs because interactions are dynamic and there is a possibility that the teammates and/or opponents change their behavior during execution. In despite, we find some works that apply this kind of algorithm directly to multi-agent systems (Goldman and Rosenschein, 1996; Wang and Gasser, 2002; Śniezyński and Koźlak, 2006). Some hybrid systems that fuse this kind of algorithms with other machine learning algorithms can also be found. For example, Jolly et al. (2007) combine evolutionary algorithms and neural networks for decision making in multi-robot soccer systems.

2.1.2 Unsupervised Learning

Unsupervised learning algorithms try to cluster similar data, thus, they are widely used for pattern recognition and data visualisation. However, it is difficult to imagine how they can be applied to solve task allocation and execution in multi-robot, cooperative systems. In

despite, it is possible to find a few works that use this kind of learning, as the one of Li and Parker (2007) that uses the Fuzzy C-Means (FCM) clustering algorithm for detecting faults in a team of robots.

2.1.3 Reward Based Learning

The basic idea of reward based algorithms is not to indicate what the best response is, but the expected result. Thus, the robot or agent has to discover the best strategy in order to obtain the desired results. The most representative algorithms of this type are the evolutionary algorithms and the reinforcement learning algorithms. Evolutionary algorithms search the solution space in order to find the best results. This could also be done by assessing a fitness function. On the other hand, reinforcement learning algorithms estimate a value function for states or state-action pairs. This is done for defining the best policy that take advantage of these values. In this work, we focus specifically in reinforcement learning algorithms.

2.2 Requirements for Learning by Experience

In order to get a solution when using the learn by experience model in a single agent or robot we need to define: a set of pre defined states or situations where the agent can act; a set of possible actions that the agent can perform; a way of modifying parameters of action selection while actions reveal as good or bad for certain situations.

The same set is necessary, and enough, when using multi-agent or multi-robot systems. It is important to note that, although agents need a pre defined set of states or situations, abstraction is made in a way that any real situation in the environment is mapped into any pre defined state. Generally, all actions are tested in all situations during the training process in order to verify if it is good to use the action set or not.

2.3 Paradigms for Applying Reinforcement Learning Algorithms in Multi-Robot Systems

Reinforcement learning algorithms were first introduced for learning of single agents. They can also be applied in multi-agent and multi-robot systems there existing several ways of doing it. A first one is by modeling all agents or robots as a single agent. This is known as the team learning paradigm. A second model is by applying the algorithms without any modification to each agent or robot that compose the team. This is known as independent learning. The third and last traditional paradigm is by learning joint actions. It means that each agent learns how to execute an action, thinking to combining it with actions that other agents will execute.

We introduce here a new paradigm, called influence value learning, besides the three traditional paradigms explained above. In our new model, the agents learn independently, but, at the same time, each agent is influenced by the opinions that other agents have about its actions. This new approach and the three traditional ones are better explained in the following.

2.3.1 Team Learning

The paradigm of multi-agent and multi-robot systems where agents learn as a team is based in the idea of modeling the team as a single agent. The great advantage of this paradigm is that the algorithms do not need to be modified. But, in robotics applications, it can be

difficult to implement it because we need to have a centralized learning process and sensor information processing.

An example of this paradigm using reinforcement learning is the work of Kok and Vlasis (2004) that model the problem of collaboration in multi-agent systems as a Markov Decision Process. The main problem in their work and other similar works is that the applicability becomes impossible when the number of players increases because the number of states and actions increases exponentially.

The use of genetic algorithms is not much affected by the exponential growth of the number of states and actions (Sen and Sekaran, 1996; Salustowicz et al., 1998; Iba, 1999). However, the need of centralized processing and information, what is still a problem impossible to be solved in this paradigm, is one of its disadvantages.

2.3.2 Independent Learning

The problems reported in learning as a team can be solved by implementing the learning algorithms independently in each agent. Several papers show promising results when applying this paradigm (Sen et al., 1994; Kapetanakis and Kudenko, 2002; Tumer et al., 2002). However, Claus and Boutilier (1998) explored the use of independent learners in repetitive games, empirically showing that the proposal is able to achieve only sub-optimal results. The above results are important when analyzed regarding the nature of the used algorithms. It may be noted that the reinforcement learning algorithms aim to take the agent to perform a set of actions that will provide the greatest utility (greater rewards). Below that, in problems involving several agents, it is possible that the combination of optimal individual strategies not necessarily represents an optimal team strategy. In an attempt to solve this problem, many studies have been developed. An example is the one of Kapetanakis and Kudenko (2002) which proposes a new heuristic for computing the reward values for actions based on the frequency that each action has maximum reward. They have shown empirically that their approach converges to an optimal strategy in repetitive games of two agents. Also, they test it in repetitive games with four agents, where, only one agent uses the proposed heuristic, showing that the probability of convergence to optimal strategies increases but is not guaranteed (2004). Another study (Tumer et al., 2002) explores modifications for choosing rewards. The problem of giving correct rewards in independent learning is studied. The proposed algorithm uses collective intelligence concepts for obtaining better results than by applying algorithms without any modification and learning as a team. Even achieving good results in simple problems such as repetitive games or stochastic games with few agents, another problem in this paradigm, which occurs as the number of agents increase, is that traditional algorithms are designed for problems where the environment does not change, that is, the reward is static. However, in multi-agents systems, the rewards may change over time, as the actions of other agents will influence them.

In the current work, this paradigm is analyzed in comparison with joint action learning and with our approach, the influence value learning. Q-Learning is known to be the best reinforcement learning algorithm (Sutton and Barto, 1998), so, the algorithms are going to be based on it. The Q-Learning algorithm for Independent Learning (IQ-Learning) is defined by equation 1.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(s_t, a_t)) \qquad (1)$$

where $Q(s_t,a_t)$ is the value of the action a_t in the state s_t, α is the learning rate $(0 \leq \alpha \leq 1)$, γ is the discount rate $(0 \leq \gamma \leq 1)$, s_{t+1} is the resultant state after executing the action a_t. And, r_{t+1} is the instantaneous reward obtained by executing the action a_t.

2.3.3 Joint Action Learning

One way for solving the problem of the independent learning model is learn the best response to the actions of other agents. In this context, the joint action learning paradigm appears. Each agent should learn what the value of executing their actions in combination with the actions of others (joint action) is. By intuits a model for other agents, it must calculate the best action for actions supposed to be executed by colleagues and/or adversaries (Kapetanakis et al., 2003; Guo et al., 2007). Claus and Bouitilier explored the use of this paradigm in repetitive games (Claus and Boutilier, 1998) showing that the basic form of this paradigm does not guarantee convergence to optimal solutions. However, the authors indicate that, unlike the independent learning algorithms, this paradigm can be improved if models of other agents are improved.

Other examples include the work of Suematsu and Hayashi that guarantee convergence to optimal solutions (Suematsu and Hayashi, 2002). The work of Banerjee and Sen (Banerjee and Sen, 2007) that proposes a conditional algorithm for learning joint actions, where agents learn the conditional probability of an action be executed by an opponent be optimal. Then, agents use these probabilities for choosing their future actions. The main problem with this paradigm is the number of combinations of states and actions that grows exponentially as the number of states, actions and/or agents grows.

As said before, in the current work, algorithms will be based on Q-Learning. A modified version of the traditional Q-Learning, for joint action learning, the so called JAQ-Learning algorithm, is defined by the equation 2:

$$
\begin{aligned}
Q_i(s_t, a1_t,...,aN_t) &\leftarrow Q_i(s_t, a1_t,...,aN_t) + \\
\alpha(r_{t+1} + \gamma \max_{a1,...,aN} &Q_i(s_{t+1}, a1,...,aN) - Q_i(s_t, a1_t,...,aN_t))
\end{aligned}
\tag{2}
$$

where ai_t is the action performed by the agent i at time t; N is number of agents, $Q_i(s_t,a1_t,...,aN_t)$ is the value of the joint action $(a1_t,...,aN_t)$ for agent i in the state s_t. r_{t+1} is the reward obtained by agent i as it executes action ai_t and as other agents execute actions $a1_t,...,ai\text{-}1_t,ai\text{+}1_t,...,aN_t$ respectively, α is the learning rate $(0 \leq \alpha \leq 1)$, γ is the discount rate $(0 \leq \gamma \leq 1)$.

An agent has to decide between its actions and not between joint actions. For this decision, it uses the expected value of its actions. The expected value includes information about the joint actions and the current beliefs about other agent that is given by (Equation 3):

$$
EV(s_t, ai) \leftarrow \sum_{a_{-i} \in A_{-i}} Q(s_t, a_{-i} \cup ai) * \prod_{j \neq i} \Pr_t(a_{-i}j)
\tag{3}
$$

where ai is an action of agent i, $EV(s_t,ai)$ is the expected value of action ai in state s_t, a_{-i} is a joint action formed only by actions of other agents, A_{-i} is the set of joint actions of other agents excluding agent i, $Q(s_t,a_{-i} \cup ai)$ is the value of a joint action formed by the union of the joint action a_{-i} of all agents excluding i with action ai of agent i in state s_t and $Pr_t(a_{-i}j)$ is the probability of agent j performs action aj that is part of joint action a_{-i} in state s_t.

2.3.4 Influence Value Learning

The learning by influence value paradigm that we propose (Barrios-Aranibar and Gonçalves, 2007a; Barrios-Aranibar and Gonçalves, 2007b) is based on the idea of influencing the behaviour of each agent according to the opinion of others. Since this proposal is developed in the context of learning through rewards, then the value of the proposed solutions will be the reward of each agent and the opinion that the other players have on the solution that the agent gave individually. This opinion should be a function of reward obtained by the agents. That is, if an agent affects the other players pushing their reward below than the previously received, they have a negative opinion for the actions done by the first agent.

From the theoretical point of view, the model proposed does not have the problems related to the paradigms of team learning and joint action learning about the number of actors, actions and states. Finally, when talking about possible changes of rewards during the learning process and that the agent must be aware that the rewards may change because of the existence of other agents, authors conjecture that this does not represent a problem for this paradigm, based on experiments conducted until now.

This paradigm is based on social interactions of people. Some theories about the social interaction can be seen in theoretical work in the area of education and psychology, such as the work of Levi Vygotsky (Oliveira and Bazzan, 2006; Jars et al., 2004).

Based on these preliminary studies on the influence of social interactions in learning, we conjecture that when people interact, they communicate each other what they think about the actions of the other, either through direct criticism or praise. This means that if person A does not like the action executed by the person B, then A will protest against B. If the person B continue to perform the same action, then A can become angry and protest angrily. We abstract this phenomenon and determined that the strength of protests may be proportional to the number of times the bad action is repeated.

Moreover, if person A likes the action executed by the person B, then A must praise B. Even if the action performed is considered as very good, then A must praise B vehemently. But if B continues to perform the same action, then A will get accustomed, and over time it will cease to praise B. The former can be abstracted by making the power of praise to be inversely proportional to the number of times the good action is repeated.

More importantly, we observe that protests and praises of others can influence the behavior of a person. Therefore, when other people protest, a person tries to avoid actions that caused these protests and when other people praise, a person tries to repeat actions more times.

Inspired in this fact, in this paradigm, agents estimate the values of their actions based on the reward obtained and a numerical value called influence value. The influence value for an agent i in a group of N agents is defined by equation 4.

$$IV_i \leftarrow \sum_{j \in (1:N), j \neq i} \beta_i(j) * OP_j(i)$$

(4)

Where $\beta_i(j)$ is the influence rate of agent j over agent i, $OP_j(i)$ is the opinion of agent j about the action executed by agent i.

The influence rate β determine whether or not an agent will be influenced by opinions of other agents. OP is a numerical value which is calculated on the basis of the rewards that an agent has been received. Because in reinforcement learning the value of states or state-action pairs is directly related to rewards obtained in the past, then the opinion of an agent will be calculated according to this value and reward obtained at the time of evaluation (Equation 5).

$$OP_j(i) \leftarrow \begin{cases} RV_j * Pe(s(t), a_i(t)) & If \quad RV_j \leq 0 \\ RV_j * (1 - Pe(s(t), a_i(t))) & In \quad other \quad case \end{cases} \tag{5}$$

Where

$$RV_j \leftarrow r_j + \max_{a_j \in A_j} Q(s(t+1), a_j) - Q(s(t), a_j(t)) \tag{6}$$

For the case to be learning the values of state-action pairs. $Pe(s(t),a_i(t))$ is the occurrence index (times action a_i is executed by agent i in state $s(t)$ over times agent i have been in state $s(t)$). $Q(s(t),a_j(t))$ is the value of the state-action pair of the agent j at time t. And, A_j is the set of all actions agent j can execute. Thus, in the IVQ-learning algorithm based on Q-Learning, the state-action pair value for an agent i is modified using the equation 7.

$$Q(s(t), a_i(t)) \leftarrow Q(s(t), a_i(t)) + \alpha(r(t+1) +$$
$$\gamma \max_{a_i \in A_i} Q(s(t+1), a_i) - Q(s(t), a_i(t)) + IV_i) \tag{7}$$

where $Q(s(t),a_i(t))$ is the value of action $a_i(t)$ executed by agent i, α is the learning rate ($0 \leq \alpha \leq 1$), γ is the discount rate ($0 \leq \gamma \leq 1$). And, $r(t+1)$ is the instantaneous reward obtained by agent i.

2.4 Robotics and Reinforcement Learning

For the implementation of reinforcement learning, it is important to define the model of the environment, the reward function, the action selection policy and the learning algorithm to be used. When applying reinforcement learning algorithms to robotics, developers have to consider that the state space and the action space are infinite.

There are several proposals to implement reinforcement learning in this kind of problem. These proposals may involve a state abstraction, function approximation and hierarchical decomposition (Morales and Sammut, 2004). In the current paper, we use state abstraction for modeling the robot soccer application where this approach was tested.

Another problem is that reinforcement learning algorithms require every action to be tested in all states. And, in robotics not all actions can be used in all states. Eventually there can exist an action that must not be used in some states. Also, if it is implemented in multi-robot systems, it is important to note that eventually some states into the model of the environment are impossible to exist in real applications. We propose a solution that can be implemented by modifying reinforcement learning algorithms in order to test actions only in certain states (for example, states pre-defined by designers or where some specialist robotic system used before)

Another problem when applying reinforcement learning to robotics is that learning algorithms require so much computational time in comparison with sampling rates needed in robotics. For solving this problem, authors propose to use off-line algorithms. The last means that algorithms will be executed during the rest time of the team of robots (e.g. After finishing to execute some tasks).

3. Learning by Imitation

Learning by imitation is considered a method to acquire complex behaviors and a way of providing seeds for future learning (Kuniyoshi and Inoue, 1993; Miyamoto and Kawato,

1998; Schaal, 1999). This kind of learning was applied in several problems like learning in humanoid robots, air hockey and marble maze games, and in robot soccer, where was implemented using a Hidden Markov Model and by teaching the robots with a Q-Learning algorithm. (Barrios-Aranibar and Alsina, 2007).

3.1 Benefits of Using Imitation with Reinforcement Learning Algorithms

Benefits of using imitation learning with any algorithm of learning by experience are directly related to the requirements exposed in section 2.2. And, in the specific case of reinforcement learning, with the problems related in section 2.4. Thus, the principal benefits of using imitation learning could be:

1. Avoid the necessity of test every action in every state. For that, agents or robots have to test action only where other agents or robots previously done.
2. Diminishes the computational time needed of convergence. Because, the search space decrease as robots only use actions in states where they previously observe that actions were used.
3. Diminishes the number of state-action pairs values to storage. Thus, it means that the number of real state-action pairs will be less than the possible ones.
4. Finally, reinforcement learning algorithms will not need to have the states and actions defined a priori.

Also, it is important to note that the exposed benefits will be obtained independent of the paradigm used for implementing reinforcement learning in a multi-robot system.

3.2 Implementing Imitation Learning

For implementing the imitation learning, we propose to do it by observing other teams playing soccer for recognizing complex behaviors (Barrios-Aranibar and Alsina, 2007). First. all robots have to recognize what actions robots they are trying to imitate are performing. After that those actions have to be grouped into behaviors. And, finally by defining in which states observed robots used them, the recognized behaviors have to be included into the data base of learning robots. This approach is explained below.

3.2.1 Recognizing Actions of Other Robots

To understand this approach of learning by imitation, concepts like role, situation, action and behavior must be defined. A role is defined as the function a robot implements during it's useful life in a problem or in part of it. Each role is composed of a set of behaviors. Also, a behavior is a sequence of consecutive actions. And, an action is a basic interaction between a robot (actor) and an element in the environment, including other robots in the team. Finally, a situation is an observer description of a basic interaction between a robot and an element in the environment, including other robots in the team. This means that a situation is the abstraction that an observer makes about something other robot does.

For recognizing action of other robots, the learners have to really analyze and recognize situations. We propose to do this analysis over games previously saved. Because the application used for testing the overall approach is the robot soccer, in current work, the saved game must include positions and orientations of the robots of both teams, the ball position and scores at every sampling time.

The analysis of saved games is made with a fuzzy inference machine. Here, situations involving each robot in the analyzed team must be made. The fuzzy inference machine includes five fuzzy variables: Distance between two objects, orientation of an object with

respect to the orientation of another object, orientation of an object with respect to another object, playing time and velocity of an object with respect to the velocity of another object. For defining these fuzzy variables, values of time (sampling times), position, orientation, distance, velocity and direction were used. Figure 1 shows possible fuzzy values for each variable.
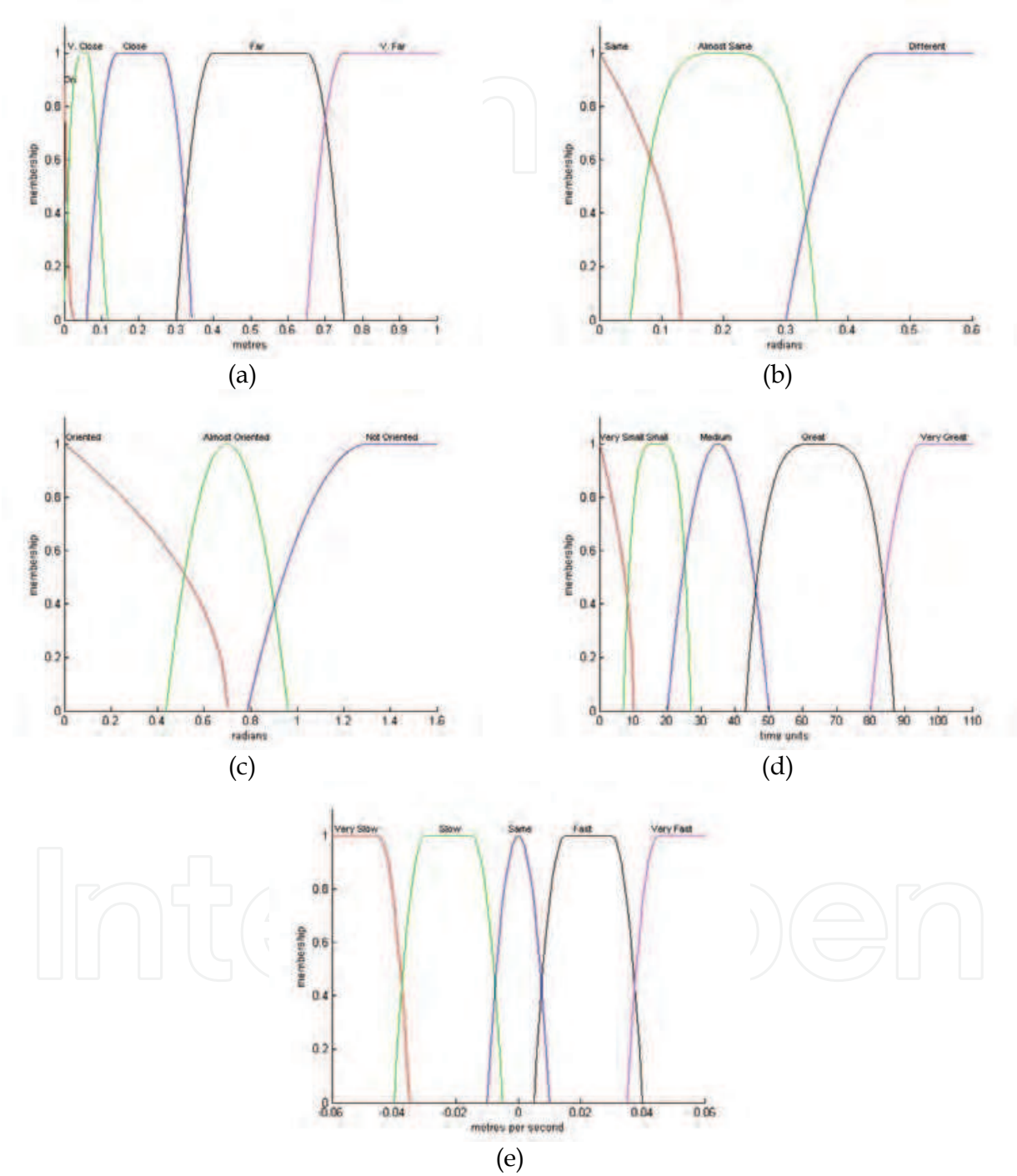

(a)

(b)

(c)

(d)

(e)

Figure 1. Fuzzy variables for recognizing situations: (a) Distance Between two Objects. (b) Orientation of an Object with Respect to Orientation of Another Object. (c) Orientation of an Object with Respect to Another Object. (d) Playing Time. (e) Velocity of an Object with Respect to the Velocity of Another Object

Distance between two objects fuzzy variable is based in the Euclidian distance between the objects positions. Orientation of an object with respect to orientation of another object fuzzy variable is based on the absolute value of the difference between the orientation angles of two objects. Orientation of an object with respect to another object fuzzy variable is based on the angular distance between two objects. Which is defined as the difference between the current orientation of the first object and the orientation needed to point to the second object. Playing time fuzzy variable is based on the sampled times of a game or a part of it. And, velocity of an object with respect to the velocity of another object fuzzy variable is based on the value of the difference between the velocity of two objects.

Code	Situation Name	Priority
001	Robot with the ball	1
011	Robot guides the ball	2
021	Robot kicks the ball	3
022	Robot loses the ball	4
023	Robot yields the ball	4
033	Robot leaves the ball	5
034	Robot moves away from the ball	10
044	Robot reaches the ball	6
045	Robot receives the ball	7
046	Robot approaches the ball	9
047	Ball hits the robot	8
057	Robot orients to the ball	11
067	Robot goes ball's X direction	12
068	Robot goes ball's Y direction	13
078	Robot orients to it's own goal	14
088	Robot approaches it's own goal	15
098	Robot moves away from it's own goal	16
108	Robot orients to adversary's goal	17
118	Robot approaches the adversary's goal	18
128	Robot moves away from adversary's goal	19
138	Robot approaches goalkeeper adversary	21
139	Robot approaches midfield adversary	21
140	Robot approaches striker adversary	21
150	Robot moves away from goalkeeper adversary	24
151	Robot moves away from midfield adversary	24
152	Robot moves away from striker adversary	24
162	Robot approaches role +1 teammate	23
163	Robot approaches role +2 teammate	23
173	Robot moves away from role+1 teammate	22
174	Robot moves away from role+2 teammate	22
184	Robot does not move	20
194	Robot moves randomly	

Table 1. Situations Defined in the Fuzzy Inference Machine

Thirty situations with fuzzy rules were defined. An additional situation called "Robot moving randomly" was defined to be used when there is a time interval that does not fulfill

restrictions of any situation. The situations are listed in table 1. Three roles were defined for the robots: Goalkeeper, midfield and striker, all of them depend on the robot position at each sampling time. Almost all situations are independent of the robot's role. The situations with codes 162, 163, 173 and 174 depend on the robot role when a situation starts. For that purpose, roles are arranged in a circular list (e.g. Goalkeeper, midfield, striker).

The recognition of a situation passes through two stages: Verification of possibility and verification of occurrence. The verification of possibility is made with a fuzzy rule called "Initial Condition". If at any time the game this rule is satisfied then the situation is marked as possible to happen. This means that in the future the rule for verification of occurrence should be evaluated ("Final Condition"). If the situation is scheduled as possible to happen, in every sampling time, in the future, the fuzzy rule "Final Condition" will be checked. The system has certain amount of time to verify this condition. If past the time limit this condition is not fulfilled, then the mark of possible to happen is deleted for that situation.

As shown in table 1, each situation has a priority for recognition on the fuzzy inference machine. Situations with priority 1 has the highest priority and situations with priority 24 have less priority.

3.2.2 Recognizing Behaviors of Other Robots

After recognizing all the situations for each of the robots of the analyzed time, the codes of these situations will be passed by self-organized maps (SOM), proposed by Kohonem. Both, the recognition of situations and the recognition of patterns of behaviours are done offline and after each training game.

At the beginning of the process, four groups of successive events are generated. They are considered all possible combinations of successive grouping of situations without changing their order, which means that each situation may be part of up to 4 groups (e.g. where it can be the first, second, third or fourth member of the group).

The groups formed are used to train a SOM neural network. After finishing the process of training, the neurons that were activated by at least 10% of the number of recognized situations are selected. Then, the groups who have activated the previously selected neurons are selected to form part of the knowledge base. To be part of knowledge base, each group or behavior pattern must have a value greater than a threshold.

The value of each group is calculated based on the final result obtained by the analyzed team. Final results could be: a goal of the analyzed team, goal of the opponent team, or end of the game. All situations recognized before a goal of the analyzed team receive a positive value α^t where $0<\alpha<1$ and t is the number of discrete times between the start of the situation and the final result. Each situation recognized before a goal of the opponent team, receives a negative value $-\alpha^t$. Finally, each situation recognized before the end of the game get the value of zero. The value of each group of situations is calculated using the arithmetic mean of the values of the situations.

After recognize the patterns of behaviours formed by four situations, groups of three situations are formed. The groups are formed with those situations that were not considered before (those that do not form part of any of the new behaviors entered in the knowledge base). It is important to form groups only with consecutive situations. It can not be formed groups of situations separated by some other situation. The process conducted for groups of four situations is then repeated, but this time will be considered the neurons that were activated by at least 8% of the number of recognized situations. After that, The process is

repeated again for groups of two and considering the neurons activated for at least the 6% of the number of recognized situations. Finally, those situations that were not considered in the three previous cases, form individually a behavior pattern. Again, the process is repeated considering the neurons activated for at least the 4% of the number of recognized situations. Since, to improve the learned by imitation strategy, each new behavior inserted in the knowledge base has to be tested by the learner. Each of these behaviors receive an optimistic value (e.g. The greatest value of behaviors that can be used in the state).

4. The Overall Learning Process

The process of learning becomes first by a stage of imitation for after try and see if the learned actions are valid for the robot is learning. Therefore, it is important to define the structure of the state and also what are the instant rewards in the application chosen (robot soccer). As said before, the number of states in robotic problems are infinite. To implement reinforcement learning authors opted up by state abstraction. The purpose of this abstraction is to get a set of finite states.

The state of a robot soccer game is constituted by the positions, orientations and velocities of the robots; the position, direction and velocity of the ball as well as scores of the game. The positions of the robots and the ball were abstracted in discrete variables of distance and position with reference to certain elements of the game. Orientations of the robots were abstracted in discrete variables of direction. Direction of the ball was abstracted in a discrete direction variable. The same was done with the velocity and scores. Finally, to recognize the terminal states of the game, there was set a final element in the state that is the situation of the game. Table 2 shows the configuration of a state.

Element	Quantity
Distance of robot to ball	6 (6 robots x 1 ball)
Distance of ball to goal	2 (1 ball x 2 goals)
Orientation of robot to ball	6 (6 robots x 1 ball)
Orientation of robot to goal	12 (6 robots x 2 goals)
Robot/ball position in relation to own goal	6 (6 robots)
Ball direction	1 (1 ball)
Ball velocity	1 (1 ball)
Game score	1 (1 game)
Game situation	1 (1 game)
TOTAL	36

Table 2. Abstraction of a state in a robot soccer game

To abstract the position of the robots, we consider that the important in the position of a robot is whether he is too close, near or far the ball and whether the ball is close, very close to or far from the goals. It is also important to know if the robot is before or after the ball. With these three discrete variables you can identify the position of the robot and its location within the soccer field in relation to all elements (the ball, goals and robots)

Besides recognize the location of the robot in relation to the ball, the goal and the other robots, it is important to know where he is oriented, so we can know whether he is ready to shoot the ball or go towards the own goal. Then, the orientation of the robot was abstracted in two discrete variables called orientation to the ball and orientation to the goal.

In the case of the direction of the ball, their speed and scores, there was defined discrete variables for each considering the characteristics of the soccer game. In the case of the ball direction, 8 discrete levels were defined (each level grouping 45°). Also, there were defined 3 speed levels. In the case of the scores, important is whether the team is winning, losing or drawing. Finally the situation of the game may be normal (not terminal state), fault or goal (terminal states).
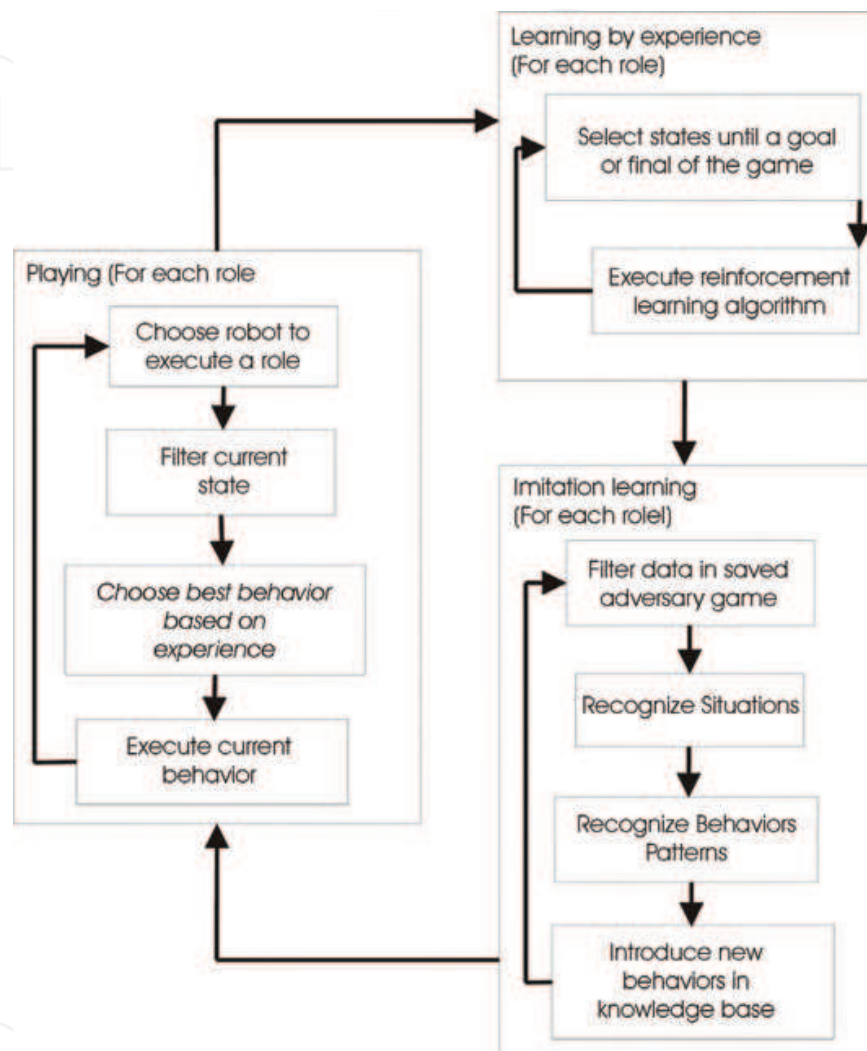

Figure 2. Training process of a team of robots learning by experience and by imitation

Considering all the possibilities of each element of the state we calculate that a robot soccer game would have 80621568 states. If we consider that the states where the situation of the game is fault or goal are only to aid in the implementation of algorithms, then we would have that the number of states decreases to 26873856. The previous value is the value of the theoretical maximum number of states in a robot soccer game. This theoretical value will be hardly achieved since not all combinations of the values of the components of the state will exist. An example of how the number of states will be reduced in practice is the case of states of the goalkeeper role. The goalkeeper role is attributed to the robot that is closest to its own goal. For states where the goalkeeper is after ball, it will be almost impossible for the other robots to be before it. Then the number of states is virtually reduced by two thirds.

Thus, considering only this case we see that the number of states for the goalkeeper reduced to 8957952.

In the case of actions in reinforcement learning, each behavior learned during imitation is considered an action in the model of reinforcement learning. Since learning by imitation is the seed for reinforcement learning, we have that reinforcement learning acts only on states and behaviors learned by the robot in previous games.

Figure 2 shows the simplified procedure of the overall training process of the control and coordination system for a robot soccer team.

Since learning of new formations, behaviors and actions, as well as the best choice for them at a particular moment, is defined by the quality of opponents teams, it is proposed that the training process is done through the interaction with human controlled teams of robots. The main advantages of this training are given by the ability of humans to learn and also by the large number of solutions that a team composed by humans could give to a particular situation.

5. What is the best Paradigm of Reinforcement Learning for Multi-Robot systems

Before assessing the paradigms of reinforcement learning for multi-robot or multi-agent systems, it is important to know that when talking about cooperative agents or robots, it is necessary that agents cooperate on equality and that all agents receive equitable rewards for solving the task. Is in this context that a different concept from the games theory appears in multi-agent systems. This is the concept of the Nash equilibrium.

Let be a multi-agent system formed by N agents. σ^*_i is defined as the strategy chosen by the agent i, σ_i as any strategy of the agent i, and Σ_i as the set of all possible strategies of i. It is said that the strategies $\sigma^*_i,...,$ σ^*_N constitute a Nash equilibrium, if inequality 8 is true for all $\sigma_i \in \Sigma_i$ and for all agents i.

$$r_i(\sigma_1^*,...,\sigma_{i-1}^*,\sigma_i,\sigma_{i+1}^*,...,\sigma_N^*) \leq r_i(\sigma_1^*,...,\sigma_{i-1}^*,\sigma_i^*,\sigma_{i+1}^*,...,\sigma_N^*) \tag{8}$$

Where r_i is the reward obtained by agent i.

The idea of Nash equilibrium, is that the strategy of each agent is the best response to the strategies of their colleagues and/or opponents (Kononen, 2004). Then, it is expected that learning algorithms can converge to a Nash equilibrium, and it is desired that can converge to the optimal Nash equilibrium, that is the one where the reward for all agents is the best.

We test and compare all paradigms using two repetitive games (The penalty problem and the climbing problem) and one stochastic game for two agents. The penalty problem, in which IQ-Learning, JAQ-Learning and IVQ-Learning can converge to the optimal equilibrium over certain conditions, is used for testing capability of those algorithms to converge to optimal equilibrium. And, the climbing problem, in which IQ-Learning, JAQ-Learning can not converge to optimal equilibrium was used to test if IVQ-Learning can do it. Also, a game called the grid world game was created for testing coordination between two agents. Here, both agents have to coordinate their actions in order to obtain positive rewards. Lack of coordination causes penalties. Figure 3 shows the three games used here.

In penalty game, k < 0 is a penalty. In this game, there exist three Nash equilibriums ((a0,b0), (a1,b1) and (a2,b2)), but only two of them are optimal Nash equilibrums ((a0, b0)

and (a3, b3)). When k = 0 (no penalty for any action in the game), the three algorithms (IQ-Learning, JAQ-Learning and IVQ-Learning) converge to the optimal equilibrium with probability one. However, as k decrease, this probability also decrease.
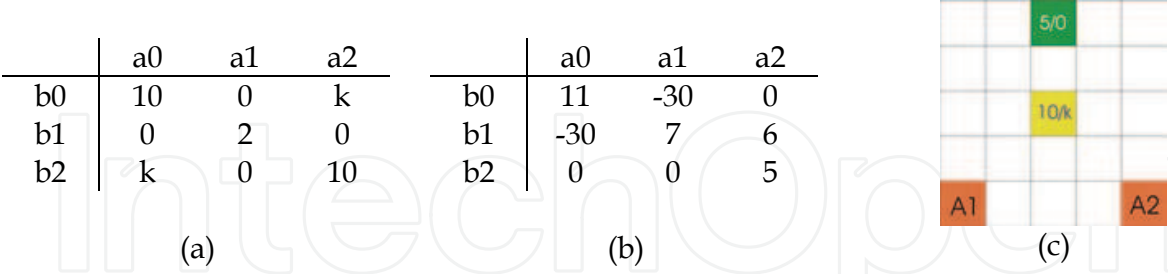
	a0	a1	a2		a0	a1	a2
b0	10	0	k	b0	11	-30	0
b1	0	2	0	b1	-30	7	6
b2	k	0	10	b2	0	0	5

(a)   (b)   (c)

Figure 3. Games used for testing performance of paradigms for applying reinforcement learning in multi-agent systems: (a) Penalty game, (b) Climbing game, (c) Grid world game

Figure 4 compiles results obtained by these three algorithms, all of them was executed with the same conditions: A Boltzman action selection strategy with initial temperature $T = 16$, $\lambda = 0.1$ and in the case of IVQ-Learning $\beta = 0.05$. Also, a varying decaying rate for T was defined and each algorithm was executed 100 times for each decaying rate.

In this problem JAQ-Learning has the best perform. But, it is important to note also that for values of k near to zero, IVQ-Learning and IQ-Learning performs better than the JAQ-Learning, and for those values the IVQ-Learning algorithm has the best probability to converge to the optimal equilibrium.

The climbing game problem is specially difficult for reinforcement learning algorithms because action a2 has the maximum total reward for agent A and action b1 has the maximum total reward for agent B. Independent learning approaches and joint action learning was showed to converge in the best case only to the (a1, b1) action pair (Claus and Boutilier, 1998). Again, each algorithm was executed 100 times in the same conditions: A Boltzman action selection strategy with initial temperature $T = 16$, $\lambda = 0.1$ and in the case of IVQ-Learning $\beta = 0.1$ and a varying temperature decaying rate.

In relation to the IQ-Learning and the JAQ-Learning, obtained results confirm that these two algorithms can not converge to optimal equilibrium. IVQ-Learning is the unique algorithm that has a probability different to zero for converging to the optimal Nash equilibrium, but this probability depends on the temperature decaying rate of the Boltzman action selection strategy (figure 5). In experiments, the best temperature decaying rate founded was 0.9997 on which probability to convergence to optimal equilibrium (a0, b0) is near to 0.7.

The grid world game starts with the agent one (A1) in position (5; 1) and agent two (A2) in position (5; 5). The idea is to reach positions (1; 3) and (3; 3) at the same time in order to finish the game. If they reach these final positions at the same time, they obtain a positive reward (5 and 10 points respectively). However, if only one of them reaches the position (3; 3) they are punished with a penalty value k. In the other hand, if only one of them reaches position (1; 3) they are not punished.

This game has several Nash equilibrium solutions, the policies that lead agents to obtain 5 points and 10 points, however, optimal Nash equilibrium solutions are those that lead agents to obtain 10 points in four steps.

The first tested algorithm (Independent Learning A) considers that the state for each agent is the position of the agent, thus, the state space does not consider the position of the other agent. The second version of this algorithm (Independent Learning B) considers that the

state space is the position of both agents. The third one is the JAQ-Learning algorithm and the last one is the IVQ-Learning.
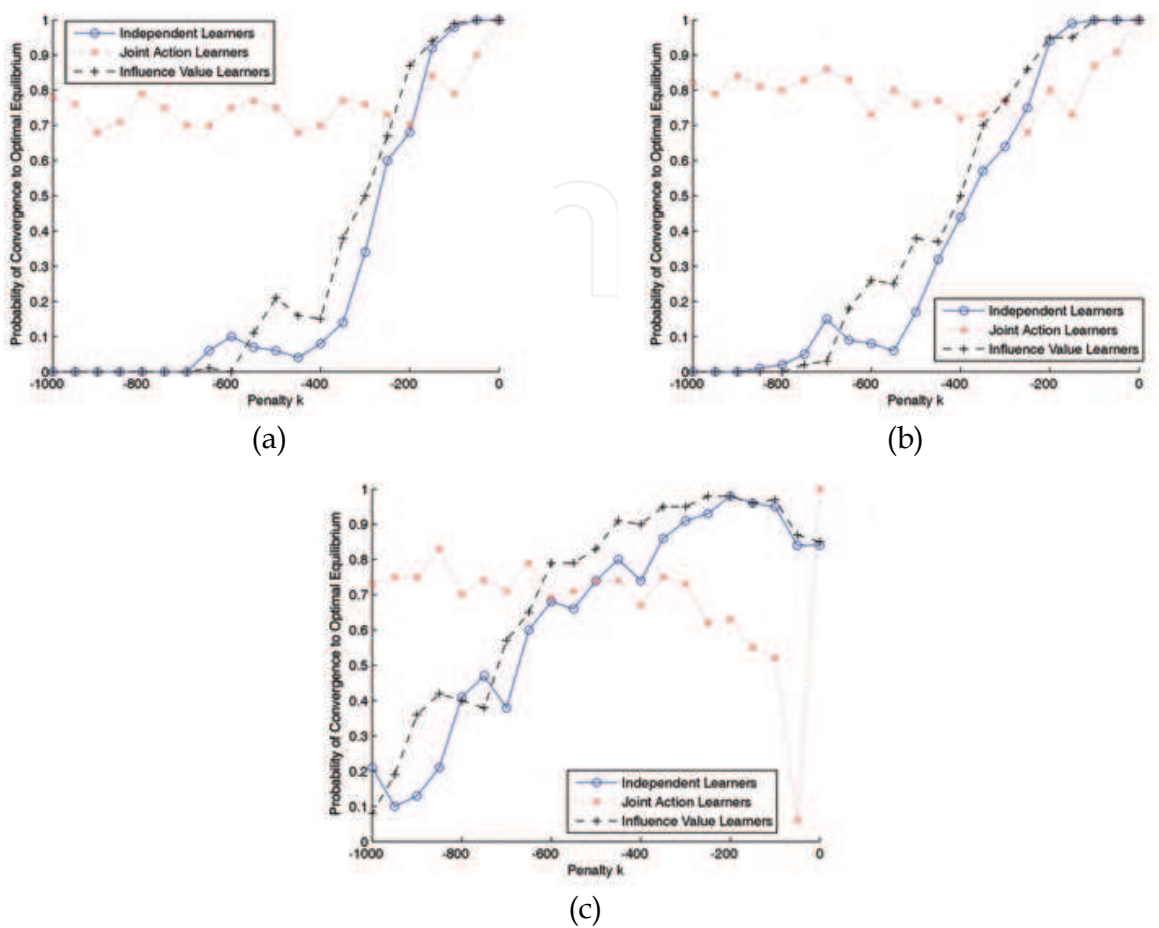

(a)

(b)

(c)

Figure 4. Probability of convergence to optimal equilibrium in the penalty game for $\lambda = 0.1$, $\beta = 0.05$ and (a) $T = 0.998^t * 16$, (b) $T = 0.999^t * 16$, and (c) $T = 0.9999^t * 16$
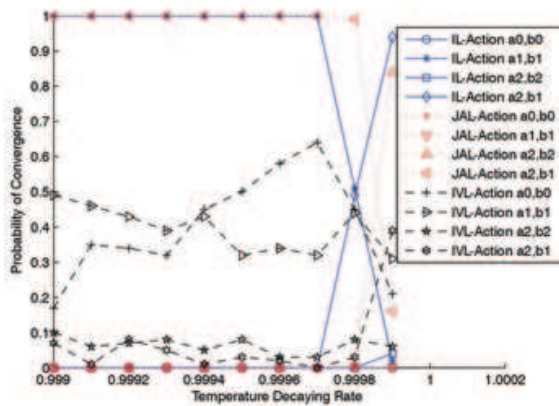

Figure 5. Probability of Convergence in Climbing Game with $\lambda = 0.1$, $\beta = 0.1$ and Variable Temperature Decaying Rate

In the tests, each learning algorithm was executed three times for each value of penalty k ($0 \leq k \leq 15$) and using five different decreasing rates of temperature T for the softmax policy

(0:99t; 0:995t; 0:999t; 0:9995t; 0:9999t). Each resulting policy (960 policies, 3 for each algorithm with penalty k and a certain decreasing rate of T) was tested 1000 of times.

Figure 6 shows the probability of reaching the position (3; 3) with α=1, λ=0.1, β=0:1 and T = 0:99t. In this figure, was observed that in this problem the joint action learning algorithm has the smaller probability of convergence to the (3; 3) position. This behavior is repeated for the other temperature decreasing rates. From the experiments, we note that the Independent Learning B and our approach have had almost the same behavior. But, when the exploration rate increases, the probability of convergence to the optimal equilibrium decreases for the Independent Learners and increase for our paradigm.
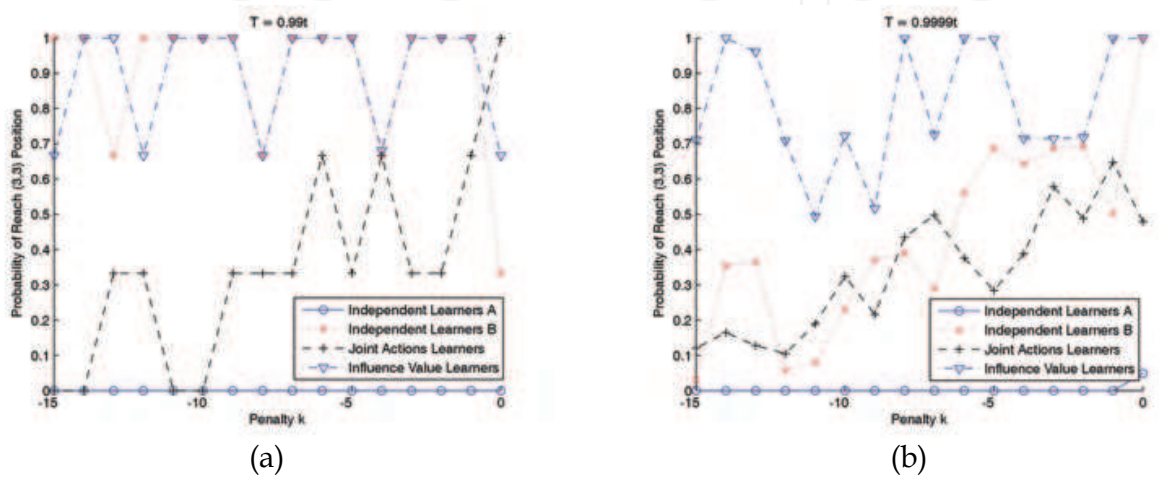

Figure 6. Probability of reaching (3,3) position for (a) T = 0.99t and (b) T = 0.9999t
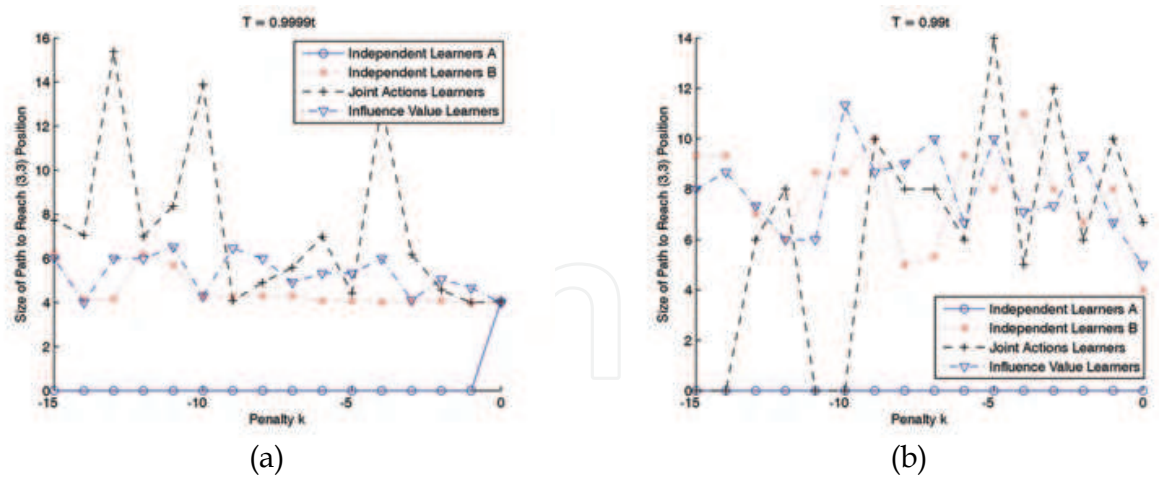

Figure 7. Size of path for reaching (3,3) position for (a) T = 0.99t and (b) T = 0.9999t

As shown in figure 7, as more exploratory the action selection policy is, smaller is the size of the path for reaching (3; 3) position. Then, it can concluded that when exploration increases, the probability of the algorithms to reach the optimal equilibrium increases too. It is important to note that our paradigm has the best probability of convergence to the optimal equilibrium. It can be concluded by joining the probability of convergence to the position (3; 3) and the mean size of the path for reaching this position.

6. How Performs the Proposed Learning Process

For testing the overall proposed approach, a basic reinforcement learning mechanism was chosen. This mechanism (eligibility traces) is considered as a bridge between Monte Carlo and temporal differences algorithms. Then, because the best algorithm that uses eligibility traces is the Sarsa(λ), it was used in the current work. Also, because the simplest paradigm for applying reinforcement learning to multi-robot systems is the independent learning, then, it was used for testing the overall approach. Finally, we conjecture that results obtained here validate this approach and also by using better techniques, like influence value reinforcement learning, results could be also improved.

A robot soccer simulator constructed by Adelardo Medeiros was used for training and testing our approach. Also, the system was trained in successive games of approximately eight minutes against a team of robots controlled by joystick by humans, and against a team using the static strategy developed by Yamamoto (Yamamoto, 2005).

An analysis of the amount of new actions executed in each game was conducted. Both training process (games against humans and games against the strategy of Yamamoto) were analyzed. Table 3 shows results of this analysis.

As can be seen in this table, the team that played against humans had a positive and continue evolution in the number of new actions executed during the first ten games, after these games the progress was slower, but with a tendency to increase the amount of not random actions. But the team that played against the static strategy developed by Yamamoto failed to evolve in the first ten games. This team only began to evolve from the game number eleven, where the initial state of the game was changed to a one with very positive scores (e.g. The apprentice team started winning).

Results shown in this table represent an advantage of the training against a team controlled by humans over the training against a team controlled with a static strategy. Moreover, it is important to note that both teams have a low rate of use of new actions during the game, this is due to the fact that initially, all the new states have the action "moving randomly". Also, exists a possibility that the apprentice is also learning random actions. Finally, it is important to note that this learning approach is very slow due to robots will have to test many times each action, including the random action. In this approach the knowledge base of each of the three roles starts empty and robots start using the action "moves randomly", then it is important to know how many new states and actions robots will learn after each game.

Figure 8 shows the number of states for each one of the three roles defined in this game. It compares both learning processes (against humans and against Yamamoto's strategy). Also, figure 9 shows the number of actions of each role. It is important to note that number of states increases during playing and during training. And, number of actions increases only during training.

As could be observed in these figures, the number of states and actions of the team trained against Yamamoto's strategy is greater than the other one. Also, it is important to note that despite this condition, the number of new actions executed by the team trained against humans is greater and increase over time.

In a soccer game, goals could be effect of: direct action of a player, indirect action, or error of the adversary team. An analysis of effectively made goals was conducted for knowing if learner teams really learn how to make goals (the main goal of playing soccer). For this

propose, a program developed for commenting games was used (Barrios-Aranibar and Alsina, 2007).

Game	Against Humans				Against Yamamoto's Strategy			
	Total	Random	Other	%	Total	Random	Other	%
1	3017	3017	0	0.00	3748	3748	0	0.00
2	4151	4115	36	0.87	3667	3667	0	0.00
3	4212	4168	44	1.04	3132	3132	0	0.00
4	2972	2948	24	0.81	3480	3480	0	0.00
5	2997	2973	24	0.80	3465	3465	0	0.00
6	2728	2657	71	2.60	3529	3529	0	0.00
7	3234	3162	72	2.23	4145	4145	0	0.00
8	2662	2570	92	3.46	3430	3430	0	0.00
9	3058	2886	172	5.62	3969	3969	0	0.00
10	2576	2427	149	5.78	5230	5239	0	0.00
11	3035	2812	223	7.35	4295	4198	97	2.26
12	3448	3447	1	0.03	4212	4149	63	1.50
13	3619	3464	155	4.28	2953	2842	111	3.76
14	3687	3587	100	2.71	4021	3874	147	3.66
15	3157	3071	86	2.72	4025	3942	83	2.06
16	4427	4293	134	3.03	4351	4168	183	4.21
17	3835	3701	134	3.49	4468	4273	195	4.36
18	3615	3453	162	4.48	3741	3598	143	3.82
19	4624	4497	127	2.75	4379	4173	206	4.70
20	4441	4441	0	0.00	3765	3548	217	5.76
21	4587	4422	165	3.60	4171	4047	124	2.97
22	4115	4115	0	0.00	4484	4329	155	3.46
23	4369	4369	0	0.00	4289	4178	111	2.59
24	3920	3920	0	0.00	3819	3646	173	4.53
25	3601	3447	154	4.28	4074	4074	0	0.00
26	4269	4269	0	0.00	4125	4125	0	0.00
27	4517	4347	170	3.76	3967	3967	0	0.00
28	6445	6195	250	3.88	3899	3745	154	3.95
29	3437	3346	91	2.65	4280	4081	199	4.65
30	3819	3686	133	3.48	3756	3704	52	1.38
31	4779	4779	0	0.00	3446	3294	152	4.41
32	4710	4546	164	3.48	4557	4370	187	4.10
33	3439	3285	154	4.48	4413	4203	210	4.76
34	4085	3928	157	3.84	3909	3742	167	4.27
35	3537	3454	83	2.35	3953	3776	177	4.48

Table 3. Analysis of new actions executed by learner teams using the proposed approach
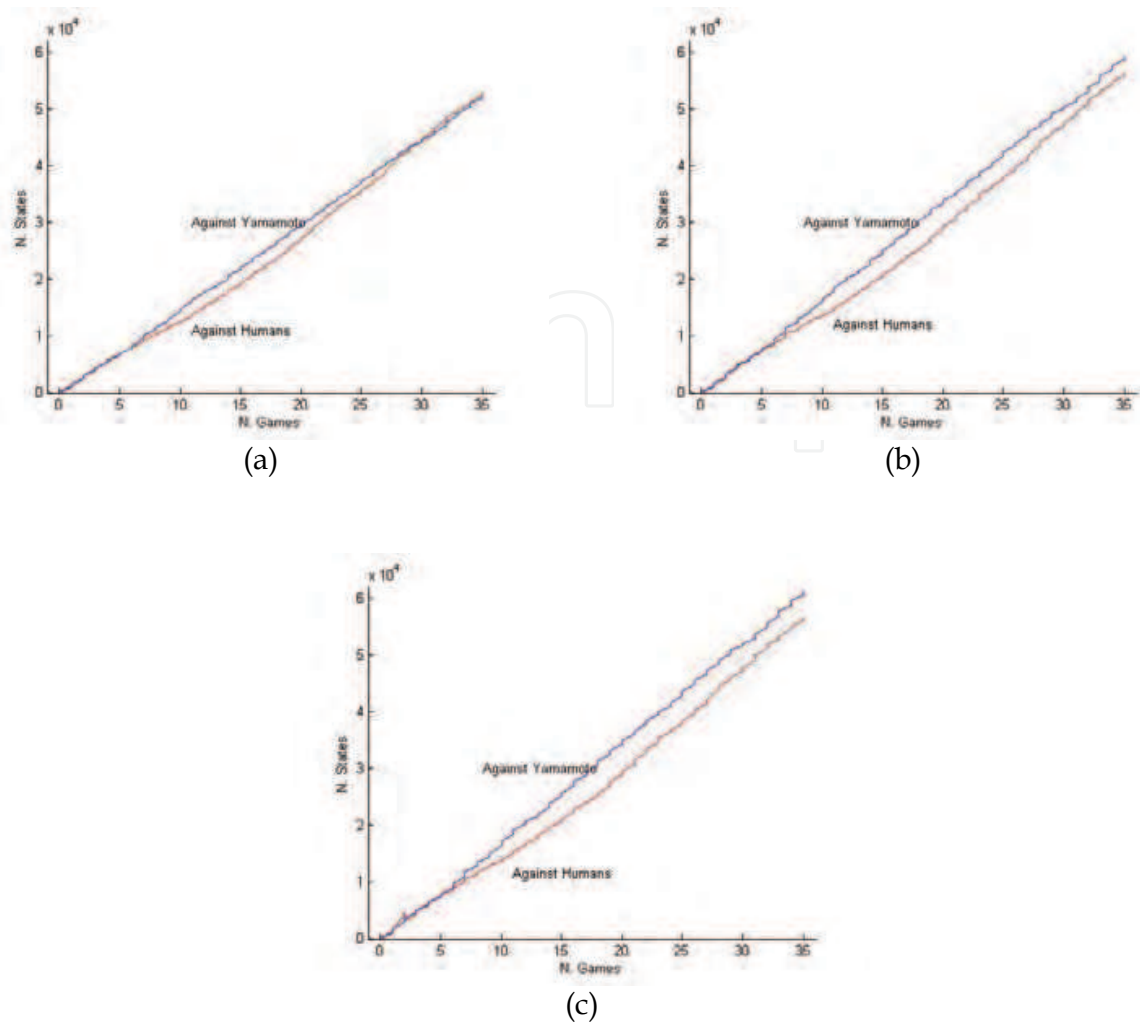
(a)

(b)

(c)

Figure 8. Number of states for roles (a) Goalkeeper, (b) Midfield, and (c) Striker

In figure 10 could be observed a comparison between the number of goals effectively made by both learners. In general, the team trained against humans achieved greater quantity of goals that the team trained against the Yamamoto's static strategy.

Although the learning process of this approach is too slow, it is important to say that learning by observation and analysis of visual information was (by our knowledge) first explored and was feasible to be implemented in robot soccer matches of robots.

7. Conclusion

One way to learn to make decisions using artificial intelligence in robotics is by leaving the learning for the rest time of the robotic system. This means, run the algorithms of learning in batches. Also, when talking about reinforcement learning in multi-robot and multi-agent systems, the paradigm proposed by authors showed better results than the traditional paradigms on the problems chosen for testing. Since these results encourage to continue this research using the model proposed in new problems and comparing it with previous proposals.

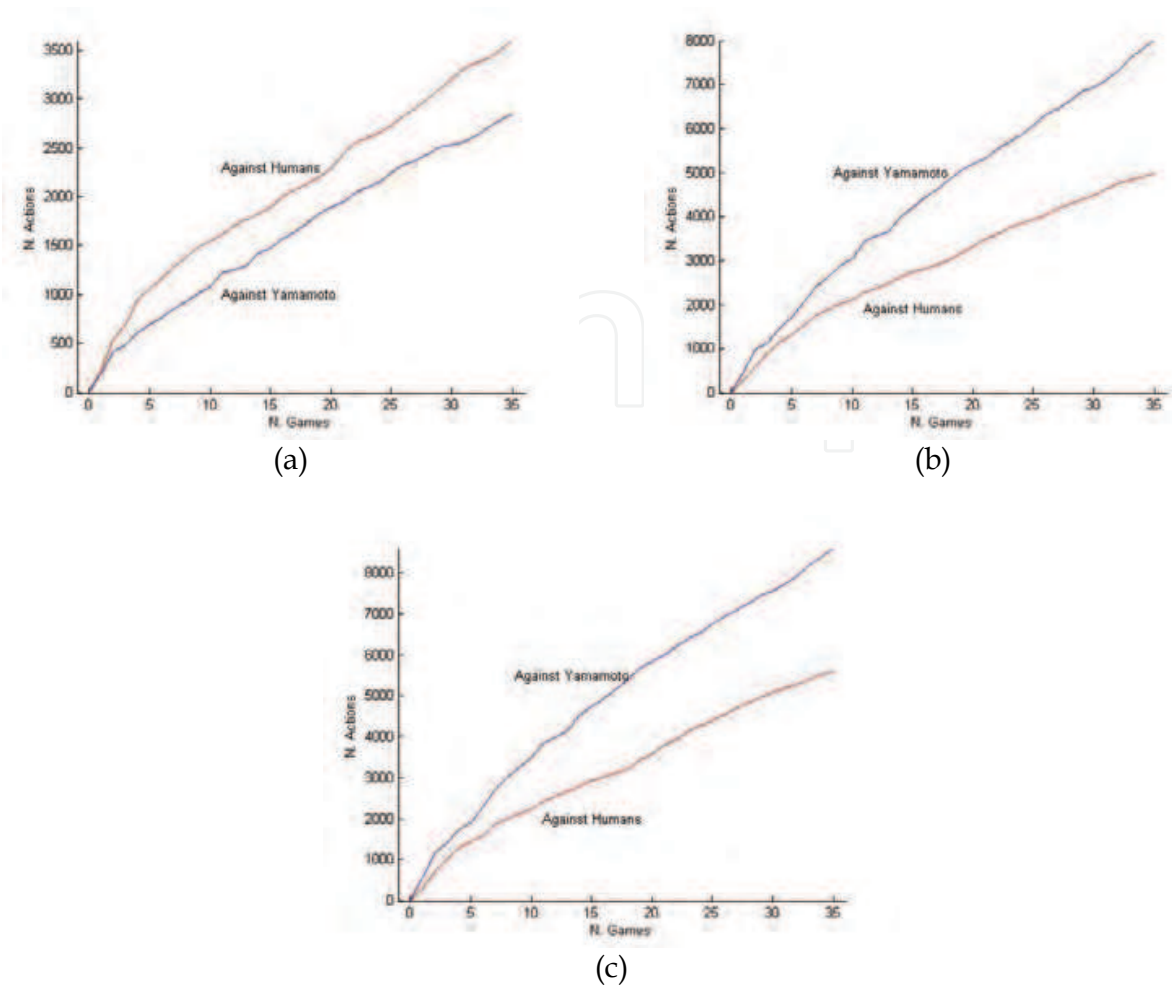(a)   (b)

(c)

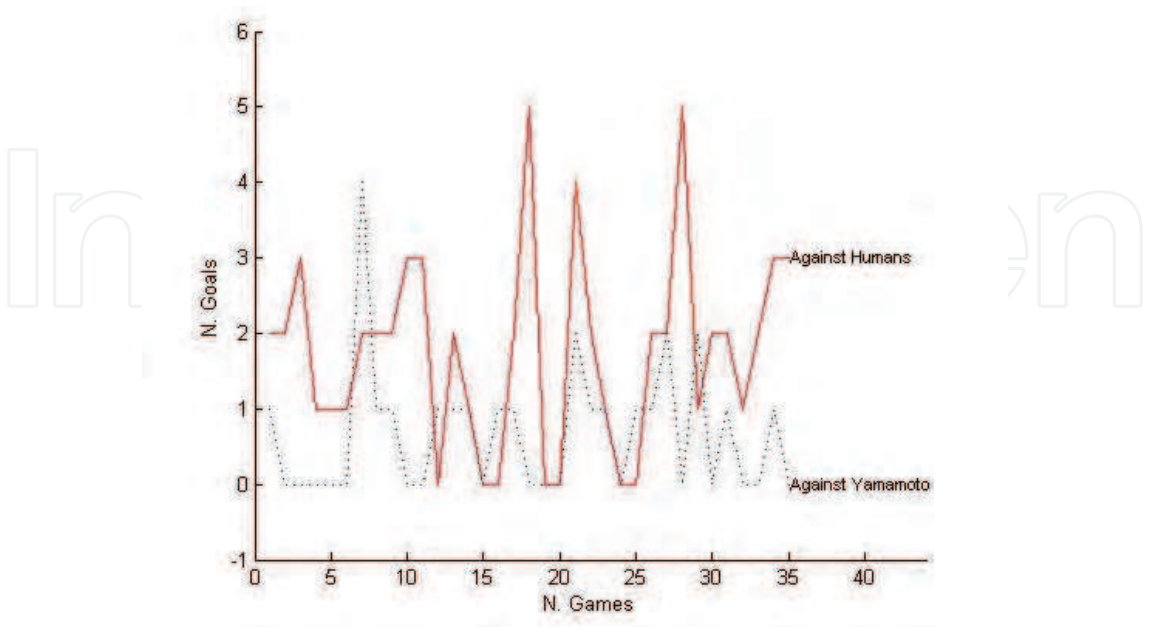Figure 9. Number of actions for roles (a) Goalkeeper, (b) Midfield, and (c) Striker

Figure 10. Number of goals effectively made by learners using this approach

During this work it was observed that learning by imitation is an appropriate technique to assist learning of techniques used in artificial intelligence, such as reinforcement learning. Thus, it is possible to overcome the limitations of these techniques when applied independently in complex problems, such as robot soccer. Limitations could appear either by the large number of states or by the large amount of available actions. Thus, learning by imitation can be used as a basis to carry out reinforcement learning in this kind of problems.

8. References

Arai, T. and Ota, J. (1992). Motion planing of multiple mobile robots. In Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1992, volume 3, pages 1761-1768.

Banerjee, D. and Sen, S. (2007). Reaching pareto-optimality in prisoner's dilemma using conditional joint action learning. Autonomous Agents and Multi-Agent Systems 15(1), 91–108.

Barrios-Aranibar, D. and Alsina, P. J. (2007). Imitation learning: An application in a micro robot soccer game. Studies in Computational Intelligence series, Mobile Robots: The Evolutionary Approach. Volume 50, 2007, Pages 201-219 .

Barrios-Aranibar, D. and Gonçalves, L. M. G. (2007a). Learning Coordination in Multi-Agent Systems using Influence Value Reinforcement Learning. In: 7th International Conference on Intelligent Systems Design and Applications (ISDA 07), 2007, Rio de Janeiro. Pages: 471-478.

Barrios-Aranibar, D. and Gonçalves, L. M. G. (2007b). Learning to Reach Optimal Equilibrium by Influence of Other Agents Opinion. In: Hybrid Intelligent Systems, 2007. HIS 2007. 7th International Conference on, 2007, Kaiserslautern. pp. 198-203.

Botelho, S. and Alami, R. (1999). M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. In Proceedings of 1999 IEEE International Conference on Robotics and Automation ICRA, pp. 1234–1239.

Botelho, S. and Alami, R. (2000). Robots that cooperatively enhance their plans, Proc. of 5th International Symposium on Distributed Autonomous Robotic Systems (DAR 2000). Lecture Notes in Computer Science, Springer Verlag.

Bruce, J., Bowling, M., Browning, B. and Veloso, M. (2003). Multi-robot team response to a multi-robot opponent team. In IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03, volume 2, pages 2281 – 2286, Taipei, Taiwan.

Chen, K.-Y. and Liu, A. (2002). A design method for incorporating multidisciplinary requirements for developing a robot soccer player. In Fourth international Symposium on Multimedia Software Engineering, 2002. Proceedings, pages 25-32, New-port Beach, California, USA.

Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In Proceedings of the 15th National Conference on Artificial Intelligence -AAAI-98. AAAI Press., Menlo Park, CA, pp. 746–752.

Filar, J. and Vrieze, K. (1997). Competitive Markov Decision Processes. Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY 10010, USA.

Goldman, C. V. and Rosenschein, J. S. (1996). Mutually supervised learning in multiagent systems. In G.Weiß and S.Sen, eds., Adaptation and Learning in Multi–Agent Systems. Springer-Verlag: Heidelberg, Germany, Berlin, pp. 85–96.

Guo, R., Wu, M., Peng, J., Peng, J. and Cao, W. (2007). New q learning algorithm for multi-agent systems, Zidonghua Xuebao/Acta Automatica Sinica, 33(4), 367–372.

Iba, H. (1999). Evolving multiple agents by genetic programming. In U.-M.L. Spector, W. Langdom & P.Angeline, eds., Advances in Genetic Programming. Vol. 3, The MIT Press, Cambridge, MA, pp. 447–466.

Jars, I., Kabachi, N. and Lamure, M. (2004). Proposal for a vygotsky's theory based approach for learning in MAS. In AOTP: The AAAI-04 Workshop on Agent Organizations: Theory and Practice. San Jose, California. http://www.cs.uu.nl/virginia/aotp/papers/ AOTP04IJars.Pdf.

Jolly, K. G., Ravindran, K. P., Vijayakumar, R. and Kumar, R. S. (2007). Intelligent decision making in multi-agent robot soccer system through compounded artificial neural networks. Robotics and Autonomous Systems. Volume 55, Issue 7, 31 July 2007, Pages 589-596.

Kapetanakis, S. and Kudenko, D. (2002). Reinforcement learning of coordination in cooperative multi-agent systems. In Proceedings of the National Conference on Artificial Intelligence. pp. 326–331.

Kapetanakis, S. and Kudenko, D. (2004). Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2004. Vol. 3, pp. 1258–1259.

Kapetanakis, S., Kudenko, D. and Strens, M. J. A. (2003). Reinforcement learning approaches to coordination in cooperative multi-agent systems. Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science) 2636, 18–32.

Kim, H.-S., Shim, H.-S., Jung, M.-J., and Kim, J.-H. (1997a). Action selection mechanism for soccer robot. In IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings, pages 390-395.

Kim, J.-H., Shim, H.-S., Kim, H.-S., Jung, M.-J., and Vadakkepat, P. (1997b). Action selection and strategies in robot soccer systems. In Proceedings of the 40th Midwest Symposium on Circuits and Systems, 1997, volume 1, pages 518-521.

Kok, J. R. and Vlassis, N. (2004). Sparse cooperative q-learning. In Proceedings of the twenty-first international conference on Machine Learning. Banff, Alberta, Canada, p. 61.

Kononen, V. (2004). Asymmetric multiagent reinforcement learning. Web Intelligence and Agent System 2(2), 105 – 121.

Kuniyoshi, Y. and Inoue, H. (1993). Qualitative recognition of ongoing human action sequences. In IJCAI93 Proceedings, Chambery, France, pages 1600–1609, 1993.

Le Pape, C. (1990). A combination of centralized and distributed methods for multi-agent planning and scheduling. In Proceedings of 1990 IEEE International Conference on Robotics and Automation ICRA, pp. 488–493.

Li, X. and Parker, L. E. (2007). Sensor Analysis for Fault Detection in Tightly-Coupled Multi-Robot Team Tasks. In 2007 IEEE International Conference on Robotics and Automation. Roma, Italy, 10-14 April 2007, pp. 3269-3276.

Miyamoto, H. and Kawato, M. (1998). A tennis serve and upswing learning robot based on bidirectional theory. Neural Networks, 11:1331–1344, 1998.

Noreils, F. R. (1993). Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment, The International Journal of Robotics Research 12(2): 79-98.

Oliveira, D. De and Bazzan, A. L. C. (2006). Traffic lights control with adaptive group formation based on swarm intelligence. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4150 LNCS, 520–521.

Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. Autonomous Agents and Multi-Agent Systems. 11(3), 387–434.

Salustowicz, R. P., Wiering , M. A. and Schmidhuber, J. (1998). Learning team strategies: Soccer case studies. Machine Learning, 33(2): 263-282.

Schaal, S. (1999). Is imitation learning the route to humanoid robots?. Trends in Cognitive Sciences, 3(6):233–242, 6 1999.

Sen, S. and Sekaran, M. (1996). Multiagent coordination with learning classifier systems. In G.WeiB & S.Sen, eds., Proceedings of the IJCAI Workshop on Adaption and Learning in Multi-Agent Systems. Vol. 1042, Springer Verlag, pp. 218–233.

Sen, S., Sekaran, M. and Hale, J. (1994). Learning to coordinate without sharing information. In Proceedings of the National Conference on Artificial Intelligence. Vol. 1, pp. 426–431.

Shoham, Y., Powers, R. and Grenager, T. (2007). If multi-agent learning is the answer, what is the question?. Artificial Intelligence. 171(7), 365–377.

Ŝniezyński, B. and Koźlak, J. (2006). Learning in a multi-agent system as a mean for effective resource management. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics. 3993 LNCS - III, 703–710.

Suematsu, N. and Hayashi, A. (2002), A multiagent reinforcement learning algorithm using extended optimal response. In Proceedings of the International Conference on Autonomous Agents. number 2, pp. 370–377.

Sutton, R. and Barto, A. (1998), Reinforcement learning: an introduction. MIT Press, Cambridge, MA, 1998.

Tumer, K., Agogino, A. K. and Wolpert, D. H. (2002). Learning sequences of actions in collectives of autonomous agents. In Proceedings of the International Conference on Autonomous Agents. número 2, pp. 378–385.

Ulam, P., Endo, Y., Wagner, A. and Arkin, R. (2007). Integrated mission specification and task allocation for robot teams - design and implementation. In 2007 IEEE International Conference on Robotics and Automation. Roma, Italy, 10-14 April 2007, pages 4428-4435.

Wang, J. and Gasser, L. (2002). Mutual online concept learning for multiple agents. In Proceedings of the International Conference on Autonomous Agents. (2), 362–369.

Wiering, M., Salustowicz, R. and Schmidhuber, J. (1999). Reinforcement learning soccer teams with incomplete world models. Autonomous Robots, 7(1): 77-88.

Wu, C.-J. and Lee, T.-L. (2004). A fuzzy mechanism for action selection of soccer robots. Journal of Intelligent and Robotic Systems, 39(1): 57-70.

Yamamoto, M. M. (2005). Planejamento cooperativo de tarefas em um ambiente de futebol de robôs. Master's thesis in electrical engineering. Federal University of Rio Grande do Norte. Brazil.

Zhang, Y. and Mackworth, A. K. (2002). A constraint-based robotic soccer team. Constraints, 7(1): 7-28.
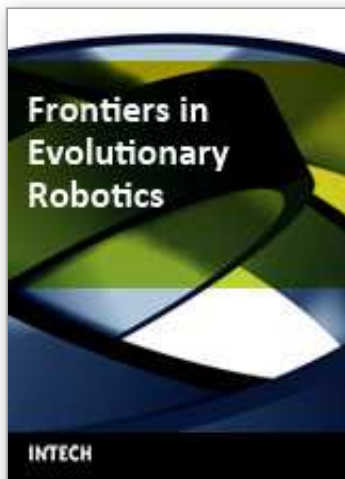
Frontiers in Evolutionary Robotics

Edited by Hitoshi Iba

ISBN 978-3-902613-19-6

Hard cover, 596 pages

Publisher I-Tech Education and Publishing

Published online 01, April, 2008

Published in print edition April, 2008

This book presented techniques and experimental results which have been pursued for the purpose of evolutionary robotics. Evolutionary robotics is a new method for the automatic creation of autonomous robots. When executing tasks by autonomous robots, we can make the robot learn what to do so as to complete the task from interactions with its environment, but not manually pre-program for all situations. Many researchers have been studying the techniques for evolutionary robotics by using Evolutionary Computation (EC), such as Genetic Algorithms (GA) or Genetic Programming (GP). Their goal is to clarify the applicability of the evolutionary approach to the real-robot learning, especially, in view of the adaptive robot behavior as well as the robustness to noisy and dynamic environments. For this purpose, authors in this book explain a variety of real robots in different fields. For instance, in a multi-robot system, several robots simultaneously work to achieve a common goal via interaction; their behaviors can only emerge as a result of evolution and interaction. How to learn such behaviors is a central issue of Distributed Artificial Intelligence (DAI), which has recently attracted much attention. This book addresses the issue in the context of a multi-robot system, in which multiple robots are evolved using EC to solve a cooperative task. Since directly using EC to generate a program of complex behaviors is often very difficult, a number of extensions to basic EC are proposed in this book so as to solve these control problems of the robot.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Dennis Barrios-Aranibar, Luiz M. G. Goncalves and Pablo Javier Alsina (2008). Learning by Experience and by Imitation in Multi-Robot Systems, Frontiers in Evolutionary Robotics, Hitoshi Iba (Ed.), ISBN: 978-3-902613-19-6, InTech, Available from:
http://www.intechopen.com/books/frontiers_in_evolutionary_robotics/learning_by_experience_and_by_imitation_in_multi-robot_systems

INTECH
open science | open minds

www.intechopen.com

Fax: +385 (51) 686 166   Fax: +86-21-62489821
www.intechopen.com