Delta3D 2005 Team Description

Mohammad Reza Khojasteh, Mahdi Azhdary, Alireza Esaghi, Mohammad Reza Chehri, and Amir Massah

AI & Robotics Laboratory, Computer Engineering Department
Shahid Bahonar College, Shiraz, Iran
{mrkhojasteh, mazhdary, aresaghi, mrchehri, amassah}@persianrobotics.net

Abstract. This paper describes the main features of the Delta3D soccer simulation 3D team that is going to take part in RoboCup competitions this year. These include Delta3D agent's architecture, a technique called the "Best Corner in State Square" for generalizing the vast number of environmental states in such complex multi-agent domains, and at last, the use of Learning Automata as a machine learning method for strategy level planning.

1 Introduction

The ideas behind Delta3D 2005 team were started by one master student from Computer Engineering Department of Amirkabir University of Technology (Tehran Polytechnic) for his MS thesis [3][4][5][6]. They were first built on CMUnited98 [1] base code in order to have a 2D soccer simulation team at that time and now, this year, we have used the ideas in newer 3D simulations either. It is now a cooperative work with four of his students in Computer Engineering Department of Shahid Bahonar College in Shiraz, Iran.

3D soccer simulation is a new and more realistic part of RoboCup simulation league in comparison with 2D soccer simulation. 3D soccer server (rcssserver3D) is based on SPADES [8] simulation middle-ware system. SPADES is an event-based system, which can be distributed among several machines. It implements the basic structure to allow the interaction between agents. SPADES implements what it calls the sense-think-act cycle in which each agent receives sensations and replies with actions. The 3D soccer simulator runs upon the SPADES, and uses ODE [9] to calculate the physical interactions between the objects of the world.

Regarding the point that the 3D soccer server is new and still there are some problems in working with it, we had to start our work with developing agent's basic skills (moving, kicking, dribbling and so on). But as usual the main interest behind the Delta3D's effort in this domain is developing and using machine learning methods in such a complex domain. So, although it seems there is still much to be done in case of basics, we are building our high level strategy planning upon our base code too.

In this paper we are going to present the main features of Delta3D 2005 that include Delta3D agent's architecture, a technique called the "Best Corner in State Square" for generalizing the vast number of environmental states in such complex multi-agent

domains, and at last, the use of Learning Automata as a machine learning method for strategy level planning.

Learning automata [2] act in a stochastic environment and are able to update their action probabilities considering the inputs from their environment, so optimizing their functionality as a result.

Also, because of the large state space of a complex multi-agent domain, it is vital to have a method for environmental states' generalization. In this paper we have introduced and designed a new technique called the "Best Corner in State Square" for generalizing the vast number of states in agent's domain environment to a few number of states by building a virtual grid in that environment.

2 DeltaAgent Architecture

Our layered agent architecture consists of three layers including communication layer, skills layer, and Decision-Control layer and it can be shown as a whole in figure 1. We have seen the ideas in references like [7] and we have added our own ideas to design this layered architecture. We are going to explain the roles of some of our components in DeltaAgent in the following paragraphs briefly.

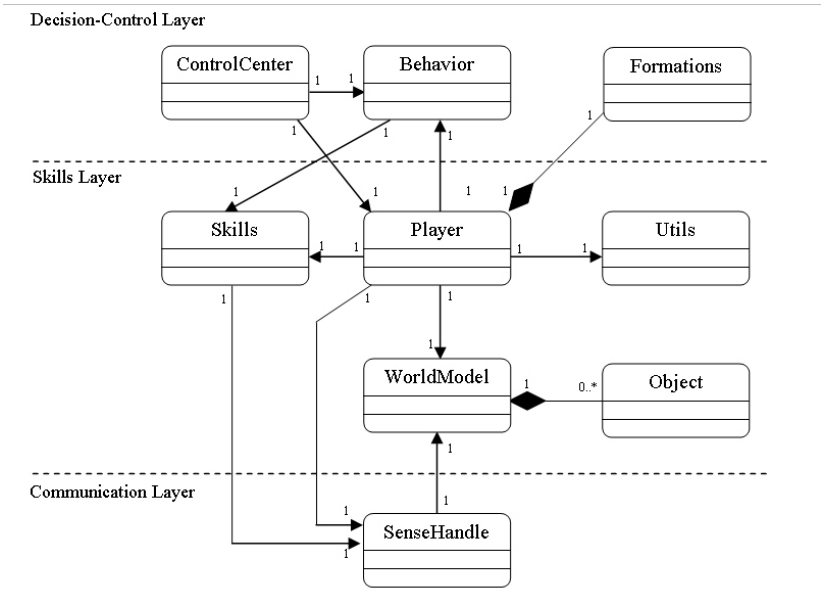

Fig. 1. DeltaAgent Architecture

SenseHandle: In this component, the agent waits for a message from the server and updates its world model according to it. In fact, this component processes the message

that the agent receives from the server. It handles the processing of these messages and sends extracted information to the WorldModel component.

*Object***:** In this component we have the information about all the objects engaged in the system. Objects in this system can be static or dynamic. The static object subclass contains information about static and fixed objects in the system such as flags, lines, and goals. The dynamic object subclass contains information about moving objects such as player and ball.

*WorldModel***:** This component contains the current representation of the world as observed by the agent. In fact, the WorldModel provides the agent with all information that it wants to know, e.g. the position and velocities of all players and the ball. It is also responsible for information concerning the current play mode, time, and score.

Skills: The behavior of an agent depends on the individual skills this agent can perform. A skill can be regarded as the ability to execute a certain action. In general, these skills can be divided into simple skills that correspond to basic actions and more advanced skills that use the simple skills are parts of more complex behavior. The execution of each skill depends on the current state of the world which is represented by the agent world model. The agent in this simulated environment is an entity that can sense, think, and act. It has some simple and complex skills, which are used to fulfill the goal of the team. The low level skills can be specified in terms of basic action commands such as, moving to a specified position and kicking the ball with a certain speed. For the agent localization, we have used the distance of each agent from the flags in the field. Thus, each time an agent senses, it gets the relative position of itself and flags. For player's motion control and stopping an agent in a specified position, we have used a complicated movement equation. High level skills are based on the low level skills and execution of them depends on the arguments which are supplied to them. The behavior of an agent is the result of selecting an appropriate skill in a given situation. In general, our agents have the ability to shoot, intercept the ball, pass, and dribble. Also we have used a simple neural net that has been trained for dribbling skills of our agents. But the main interest behind the Delta3D's effort in this paper is developing and using machine learning and LA techniques for pass selection that we will explain in the later sections.

The skills component contains the low level skills and provides the Decision-Control layer with high level commands, such as 'dash to a certain position', 'kick the ball in a certain direction with a specified speed', 'shooting to a certain point', 'dribble to some target position with a certain speed', and so forth. The components in Decision-Control layer should decide to take an action to get the highest profit. This is where we have used Learning Automata in order to improve our passing skills for our agents.

Behavior: Our Decision-Control layer consists of Behavior, Formations, and ControlCenter components. The most important modules in these components are positionMaker module for the player to position on the field depending on the game state, ballHandler module for handling the ball for intercepting, passing, and dribbling, shooting module for shooting the ball and so forth.

Player: This component consists of all the necessary information for performing the agent's individual skills such as intercepting the ball or kicking the ball to a desired position on the field.

Formations: This component consists of information about team formation and a method for determining a strategic position.

3 Learning Automata

As a model for learning, learning automata act in a stochastic environment and are able to update their action probabilities considering the inputs from their environment, so optimizing their functionality as a result.

In fact it's the first time that Learning Automata has been used in such a complex domain. We have had simulations with fixed structure learning automata and variable structure learning automata [3][4][5][6], but for our present team we have used Lrp learning automata. We give a brief description of variable structure learning automata and Lrp here:

Variable Structure Automata: A variable structure automaton is represented by a sextuple $\langle \underline{\alpha}, \underline{\beta}, \underline{\Phi}, \underline{P}, G, T \rangle$, where $\underline{\beta}$ is a set of inputs, $\underline{\Phi}$ is a set of internal states, $\underline{\alpha}$ is a set of outputs, $\underline{P}$ denotes the state probability vector governing the choice of the state at each stage k, G is the output mapping, and T is the learning algorithm. The learning algorithm is a recurrence relation and is used to modify the state probability vector. It is evident that the crucial factor affecting the performance of the variable structure learning automata, is learning algorithm for updating the action probabilities. Various learning algorithms have been reported in the literature [2]. An Example of the variable structure type is L_{rp} automata that we summarize in the following paragraphs.

In *linear reward penalty algorithm (Lrp)* scheme, the recurrence equation for updating P is defined as:

$$p_j(k) = \begin{cases} p_j(k) + a\left(1 - p_j(k)\right) & \text{if } i = j \\ p_j(k)(1-a) & \text{if } i \neq j \end{cases} \quad (1)$$

if $\beta(k) = 0$ (i.e. reward received from environment) and

$$p_j(k) = \begin{cases} p_j(k)\left(1 - b\right) & \text{if } i = j \\ \frac{b}{r-1} + (1-b)p_j(k) & \text{if } i \neq j \end{cases} \quad (2)$$

if $\beta(k) = 1$ (i.e. penalty received from environment).

The parameters a and b represent reward and penalty parameters, respectively. The parameter a(b) determines the amount of increase (decreases) of the action probabilities. For more information on learning automata the reader may refer to [2].

As we will explain in the next part we have generalized the environmental states for each agent into 8 states and in each state we have one Lrp learning automata that is going to have 8 actions. So, the total memory that our typical agent consumes is just 64 floats.

Our agents use their learning automata for their pass selection and their high level strategies and they do this by tuning the probabilities they have for their actions in each state. Also, in case that our agents don't have the ball, they simply position themselves again using the same learning automata.

4 The "Best Corner in State Square" Technique

The number of states in a simulated robotic soccer domain is very large and therefore, for an agent to consider all of them is impossible. In fact, the most important job in this regard, is to design a proper generalization of the environmental state space for the agent. If we call the set of agent's actions A, each agent will have |A| possible actions in each of |V| states and therefore, the set to be learned for the agent will have at most $|V| \times |A|$ elements. If we choose the sets V and A wisely, our agents can learn effectively in a complex and real-time environment using limited samples. In fact, the sets V and A should have the property that cover all states and actions as much as possible and they should be good mappings of the sets of all possible states and actions that exist in the domain of agents. For generalization of the environment, we divide the field of soccer into 150 equal squares each with side length 7. By doing so, at any moment of the simulation, each agent is in one of these squares. Considering the fact that each agent has a limited view of its environment, it can't count on the whole field. So, we have focused on 24 squares around the square that has the agent in it.

We have considered a numeral quantity for each of 8 squares (directions) around the agent; the north-west square, the north square, the north-east square, the east square, the south-east square, the south square, the south-west square, and the west square. Note that we have used north for the direction toward the opponent's goal.

If the agent located in a square sees one of its opponent agents in one of its north-west squares, it adds a negative quantity to its corresponding direction (in this case north-west) quantity. This would be vice versa in the case that the agent sees a teammate.

The quantity added to the direction quantities is conversely proportional to the distance of the mentioned agent (opponent or teammate) in that direction from the agent that has the ball. In fact the agents that are closer to the agent that has the ball (and are located in the 24 squares around its square) have more effect on its direction quantities than the agents that are far (and of course still located in the above-mentioned 24 squares) from it. Note that we haven't considered any effects on the agent's direction quantities for the agents that are outside the above 24 squares.

As a result, by computing the overall positions of the agents (opponents or teammates) that are located in the 24 squares around its square, each agent will have 8 numbers. By assigning these 8 numbers to the 8 directions around the agent and choosing the maximum of these values at any instant, we will have 8 states. In each state we will have 8 actions. Each action corresponds to sending the ball to the center of one of its immediate squares for the agent that possesses the ball (based on the learned information). By doing so, we have reduced the state space around the agent to 8 states that seem to cover all the states in that space. Our experiments [3][4] show that the similar states are mapped into a unique state by using this technique. It is ob-

vious that in any time instant *t*, there may be any number of teammates and agents in the 24 squares around the agent's square. If any of these agents are located in the same square that the mentioned agent is located in, the agent will change its numeral quantities depending on which of the 8 directions (relative to itself) these teammates and opponents are located in.

In fact, the agent that possesses the ball, calculates the "Best Corner in State Square" that is its state, by finding the maximum value of the quantities for each of its square's 8 corners gained by the above-mentioned process.

We have also used these state squares for determining different formations for the team and for the positioning of each player on the field.

Note that selecting a state (or corner) by the agent that possesses the ball in the above case doesn't mean that this agent should send the ball to the center of the immediate square (or toward the corner) that is associated with that state. This state just shows one of the 8 states that the ball possessor can be in it. It is clear that in each state, the agent can choose one of its 8 possible actions (sending the ball to the center of each of its immediate squares) independently of the state selected and based on the learned information (experience) in that state. Although our definition of the state for the possessor of the ball may lead us think that the direction for sending the ball should be to the center of the immediate square corresponding to the selected (corner or) state, but in a lot of cases, this won't be the optimum action for our agent [3][4]. So, our states are independent of the best possible action that can be done in them. Therefore, each agent has at most 64 elements to be learned (8 possible actions in each of 8 possible states). In fact, in each state, each agent chooses its optimal action (in that state) using the corresponding learning automata.

The overall scenario for each agent in our simulations is like this: if the possessor of the ball sees any player in its way toward the opponent's goal, it determines its state and then use the corresponding learning automata (for that state) to select its optimal action that seems to be effective in order to achieve the common goal of the team. Also, the player without the ball, goes to the center of the immediate square that finds it proper for receiving the ball from its teammate that has the ball, using the same mentioned technique. Therefore in our simulations, the player without the ball can have its own states and its own actions for moving to one of the 8 directions in order to receive the ball.

Also in our method, the player is not forced to move (pass or shoot) forward in all states and it may (as the result of one of its 8 actions) passes back, dribbles, sends ball to out, or shoots the ball to an empty space in the field if it finds it more appropriate (i.e. if it is probable that a teammate or itself in some cycles later chases the ball).

5 Cooperation in our team using learning automata

We used learning automata for cooperation among Delta3D 2005 team members. By now, various machine learning methods such as Q-learning, genetic algorithms, decision trees, behavioral learning, to mention a few, have been used for training the soccer player agents. To our knowledge, this is the first attempt to use learning automata in cooperation in multi-agent systems.

We use 8 learning automata for each agent (one automata for each corner in the "Best Corner in State Square") and 8 actions for each learning automata; sending the ball to the center of one of agent's 8 immediate squares as defined in the "Best Corner in State Square" technique.

In our team, the agents determine their current state by the "Best Corner in State Square" technique. Then the agent that possesses the ball performs the action that is advised by the corresponding automata in its state. The agent then percepts its action's result, and gives itself a reward or a penalty depending on that result.

In fact, our agent simply gives itself a reward if the ball has gone toward opponent's goal and one of its teammates has chased the ball (as the result of its action). Similarly, the agent gives itself a penalty if the ball has gone toward its team's goal and one of the opponent players has chased the ball (again as the result of its action). In all other cases, the agent does not give itself any reward or penalty and leaves its learned values unchanged.

Note that we have simulated our agents to learn from zero (i.e. without any previous knowledge of the environment before starting the simulation). Also, we have used the agent itself for the judgement about its action's results and this let us have what we call "distributed judgement", again a multi-agent approach.

We should point that we have used ($a = b = 0.1$) for L_{rp} we used. The parameters a and b are reward and penalty parameters respectively [2].

Our simulations [3][4][5][6] showed us that learning automata perform well in order to have a cooperative team of agents in a complex multi-agent domain.

6 Conclusion and Future Directions

Our goal was to use learning automata for successful production of a series of actions for agents that were members of a team, such that the resulting team can act well in multi-agent, adversarial, noisy, real-time, and most important collaborative environments.

In this paper we also introduced the "Best Corner in State Square" technique for proper generalization of the environmental states which maps similar states into one unique state. The most important characteristic of this technique is to map a continuous state space to a discrete and finite state space by forming a virtual grid on the agent's environment and at the same time to maintain the important roles of some aspects such as distance and angle in agent's state. The results of our simulations, show that the "Best Corner in State Square" technique in which the agent's decision is based on its local space is very effective when one wants to use local cooperation in order to achieve a global team goal in cooperative and complex multi-agent domains [3][4][5][6].

Most of Dealt3D's effort has been devoted to developing and improving agent's skills, especially low level ones. But we think even for future, it's necessary to work more on our team low level skills and to use some techniques for opponent modeling.

7 Acknowledgement

At last, we would like to thank the 3D soccer server developer's team for supporting this new server and for providing such a lovely test bed for MAS researches.

References

1. Stone P., Layered Learning in Multi_Agent Systems, PhD thesis, School of Computer Science, Carnegie Mellon University, December 1998.
2. Narendra K.S. and Thathachar M.A.L., Learning Automata: An Introduction, Prentice Hall, Inc., 1989.
3. Khojasteh M. R. and Meybodi M. R., The technique "Best Corner in State Square" for generalization of environmental states in a cooperative multi-agent domain, Proceedings of the 8th annual CSI computer conference (CSICC' 2003), pages 446-455, Mashhad, Iran, Feb. 25-27, 2003.
4. Khojasteh M. R., Cooperation in multi-agent systems using learning automata, M.Sc. thesis, Computer Engineering Faculty, Amirkabir University of Technology, May 2002.
5. Khojasteh M. R. and Meybodi M. R., The evaluation of learning automata in cooperation between agents in a complex multi-agent system, Proceedings of the 5th Conference on Intelligent Systems (CIS2003), Ferdowsi University of Mashhad, Iran, Oct. 14-16, 2003.
6. Khojasteh M. R. and Meybodi M. R., Learning automata as a model for cooperation in a team of agents, Proceedings of the 8th annual CSI computer conference (CSICC' 2003), pages 116-125, Mashhad, Iran, Feb. 25-27, 2003.
7. Kok J. R., Vlassis N., and Groen F., UvA Trilearn 2003 Team Description, Faculty of Science, University of Amsterdam.
8. Riley, P., SPADES (System for Parallel Agent Discrete Event Simulation), 2003.
9. Smith, and Russell, Open Dynamics Engine v0.039 User Guide, 2003.