An Integrated Multi-Agent Decision Making Framework for Robot Soccer


Abstract—Decision making is one of the most interesting issues to be solved in the RoboCup domain. This paper presents an integrated multi-agent decision making framework for robot soccer which addresses the problems of where to kick the ball to, and how to position in the field when the robot does not have the ball. The decision making framework is based on an MDP model of a soccer game, and it is integrated in the sense that the same criterions are used by the different robots for the selection of both the ball destination and the robot position. The proposed decision making framework is tested in simulated robot soccer matches, and decisions made in selected situations are shown.

Index Terms—Integrated Decision Making, Robot Soccer, Markov Decision Process

I. INTRODUCTION

This paper proposes a framework for multi-agent decision making in robot soccer. Soccer is a game of multiple players and at each moment every player in a team must make a decision. The decisions that all players in a team perform should be consistent. In other words, the set of the actions that they execute at any instant should be seen as a team action vector that intends, as a whole, to achieve the team objectives. In this paper, a soccer team is arbitrarily divided into three subsets: (i) the goalie, (ii) a robot which goes to the ball or is kicking it, and, (iii) the rest of the robots, which are positioning over the field. The goalie behavior is not addressed by this paper, and the behavior of going to the ball does not need further explanation. Thus, the paper focus in two kinds of decisions that robots must perform: where to kick the ball to, and where to position in the field when the robot does not have the ball. From now on the robot which goes to the ball or is kicking it will be called kicking robot and the rest of the robots, which are positioning over the field, excepting the goalie, will be called positioning robots. Analogously, the decisions of where to kick the ball to and where to position itself over the field will be called respectively the kicking decision and the positioning decision. The proposed decision making framework ensures that both decisions are performed in a consistent way even when they are performed by different robots that do not communicate their intentions each other. By using the proposed framework a team is able to perform from simple individual behaviors, like kicking to the goal when it is likely to score, to complex coordinated behaviors, like one positioning robot moving straight to the point where it will receive a pass, while the kicking robot goes to the ball and a defender tries to cover a dangerous zone.

Both the kicking and the positioning decisions must balance defensive and offensive considerations. In this paper this balance is achieved by using a Markov Decision Process (MDP) model of the game. The system is called multi-agent because no central coordination is performed between the robots, i.e. they have a distributed control. What makes them able to perform coordinated behaviors is that they share a common decision making criterion and they take into account what their partners are able to do.

This paper is organized as follows. In section II is presented some related work. The proposed framework is described in section III. In section IV, preliminary experimental results are presented. Finally, in section V some conclusions are drawn.

II. RELATED WORK

Several decision making frameworks have been proposed for the robot soccer problem. Most of them take care of one single decision, like the kicking decision or the positioning decision, or even subsets of them, like passing the ball or avoiding the opponents.

The methods presented in [5] and [6] are examples of positioning methods. They are based on Voronoi diagrams and dominant region diagrams, respectively. Voronoi diagrams partition the space in cells in which one robot is the closer in distance. Dominant region diagrams are similar to Voronoi diagrams but they change the meaning of closer, by considering the time that the robot will need to get to each point instead of the distance to it.

There are several methods for making the kicking decision. Some of them consider only a part of this decision. For example, in [7] a method for selecting a pass is proposed. This method uses pareto optimality for balancing gains, risks and costs. In [4], the probabilities of accomplishing some
prioritized objectives (passing, self-passing, shooting, and clearing) are estimated for each kick, and the feasible objective with the higher priority is selected. In [2], a generalized objective which takes into account simultaneously all the listed objectives considered in [4], plus other possible objectives, as for example leading passes.

Differing from the mentioned works, the here proposed approach is an integral decision making framework in the sense that it considers the kicking and positioning decisions and it makes them work together.

III. PROPOSED APPROACH

Our model of the game is based on Markov Decision Processes (MDP). A coarse MDP is used to approximate the short-term convenience of having the ball in any position of the field.

A. Markov Decision Processes

A Markov Decision Process (MDP) [1] is a probabilistic model for a stochastic process that is influenced by the agent’s actions. The model considers the existence of a state space, $X$, and an action space, $A$. At each instant $k$, the state $x_k$ is sampled from a probability density function (pdf), known as the process model, conditioned on the past state $x_{k-1}$ and the current action $a_k$:

$$p(x_k|x_{k-1},a_k)$$

Additionally, every instant, the agent receives a reward $R(x_k)$ which is function of the current state. Then, a value function may be defined as the expected value of the discounted sum of the future rewards:

$$V(x_k) = E \left[ \sum_{i=k}^{\infty} \gamma^{i-k} R(x_i) \right]$$

Where $\gamma \in (0,1]$ is a discount factor that gives more preponderance to the closer-in-time rewards. A policy $\pi(x_k)$ is defined as a function of the state which tells the agent what is the most convenient action to execute. The problem then consists in finding an optimal policy $\pi^*(x_k)$ which maximizes the value function. It has been proven that an unique optimal policy exists and there is an associated optimal value function $V^*(x_k)$, and there are several methods for finding both of them [1].

B. Soccer Game Model

In the proposed approach the time is discretized in a non-uniform way that allows the definition of a one-to-one relationship between the considered instants and the kicks that the ball receives. More specifically, instant $k$ is defined as the moment just when the ball stops after receiving the $k^{th}$ kick from any robot. The state of the game is defined as a vector $x_k = (x_{k,r},x_{k,b})$, where $x_{k,b}$ is the position of the ball and $x_{k,r} = (x_{k,r,1},\ldots,x_{k,r,N_r},x'_{k,r,1},\ldots,x'_{k,r,N_r})$ is a vector containing the poses of all robots, being $x_{r,i}$ and $x'_{r,i}$ the poses of respectively the $i^{th}$ robot partner and $i^{th}$ robot opponent, and $N_r$ the number of robots per team. Following the argumentation in [2], we define a reward $R(x_k)$ for the state $x_k$ which is only non-zero when in $x_k$ the ball is inside any goal (a goal is scored). If the goal is for, $R(x_k) = 1$. If it is against, $R(x_k) = -1$.

C. Action Space

For a kicking robot, we will call its action space $A_{kick}$, and an action is defined as a pair:

$$a_k = (\theta_k,\kappa_k)$$

Where $\theta_k$ is the kicking angle of the robot, and $\kappa_k$ is a kick which is selected from a fixed finite set. For a positioning robot, an action is defined as the pose that the robot intends to reach in the remaining time up to the next instant.

For a positioning robot, we will call its action space $A_{pos}$, and an action is defined as an absolute desired destination pose.

D. Approximated MDP

It is theoretically possible to solve offline the MDP resulting from the model described in the previous subsections and to store the resulting optimal policy function for online operation. However, the huge number of states resulting from discretizing the state space, even with a very coarse discretization, makes this solution impractical. Thus, the proposed framework relies on a strong approximation from the original MDP. This approximation is based on two facts: (i) the only part of the state which influences the reward is the position of the ball, and (ii) the ball moves much faster than the robots. The approximation consists of assuming only the position of the ball as a part of the state and supposing that the poses of the robot will not change in time. Thus, the resulting MDP could be seen as a short term MDP which will not be valid for a long period of time. As will be shown later this approximated MDP is able to be calculated online, and yields enough information to make both positioning and kicking decisions.

In order to allow a definition of the transition probabilities for this reduced MDP a single binary variable is added to the state: team ID of the robot that will kick the ball next time. This binary variable is called the controlling team at instant $k$ and is noted $c_k$. The resulting approximated MDP and its reduced state will be called respectively the ball-control MDP and the ball-control state. The ball-control state is then defined as:
\[ \bar{x}_k = (x_{b,k}, c_k) \]  

(4)

The ball position is discretized using a coarse grid to allow online computation. The action space is also dramatically approximated. Since the robots are considered to be static, no positioning decision is necessary for the resolution of the ball-control MDP and only the kicking decision is discretized. The only relevant action is that of the kicking robot whose team owns the control. Then, in the ball-control MDP, an action is defined as the desired destination cell of the ball, relative to the source cell. This action space is limited by the maximum reachable distance given the available set of kicks. Figure 1 shows the ball-control state space and the ball-control MDP action space.

Two terminal states are added to the ball-control MDP. These terminal states correspond to the scoring by respectively the own team and the other team. Consequently with the definition of the original MDP, the terminal states are the only ones which have a non zero reward, and their reward is also \( \pm 1 \). These states are also the only ones which have no control team because since the MDP is not continued from this instant on, it is irrelevant who will kick the ball next.

![Fig. 1. Ball-control MDP. In this example the soccer field is discretized in a 5x3 grid. For each cell of that grid there are 2 states one for the control of each team. The red circle represents the maximum kick distance supposing that the current state is in the central grid. The red arrows represent the ball-control MDP actions from the previous definitions. In this case there are 8 actions and each of them represents a displacement of the ball to any of the 8 cell neighbors. There are two terminal and non-zero-reward states. Each of them corresponds to the space inside each of the goals.](image)

The ball-control MDP policy may be divided into two sub-policies: the own policy and the opponent policy. The selected sub-policy is determined by the controlling team. The own team policy intends to maximize the value function, while the opponent policy intends to minimize it.

In the ball-control MDP, the transition probabilities are defined as follows. Once an action \( \bar{a}_k \) is executed, the ball position is supposed to reach its objective, given the past state and the executed action:

\[ x_{k,b} = x_{k-1,b} + \delta(\bar{a}_k) \]  

(5)

Where \( \delta(\bar{a}_k) \) is the displacement of the ball resulting from action \( \bar{a}_k \). Then only two states are possible, depending on which team will get the control. The controlling team \( c_k \) is sampled from an heuristic pdf which depends on the full state, \( x_k \) (\( c_k = 0 \) means the own team controls the ball):

\[
p(c_k|x_k) = \begin{cases}  
\frac{d'_k}{d'_k + d_k} & c_k = 0 \\
\frac{d_k}{d'_k + d_k} & c_k = 1
\end{cases}
\]  

(6)

Where \( d_k \) and \( d'_k \) are the minimum distances to the ball from respectively the partners and the opponents. Put in equations,

\[ d_k = \min_i \|x_{k,b} - x_{k,r,i}\|; \quad d'_k = \min_i \|x_{k,b} - x'_{k,r,i}\| \]  

(7)

As a result, the ball-control MDP is an approximate model of the ball flow in the next few steps. It is not directly used for making decisions, neither kicking nor positioning. Instead, the resulting value function is used as a base for more precise calculations that will be detailed below.

E. Kicking Decision

The kicking decision consists of selecting the best kicking action given the current state. For that purpose, a discretization of the full action space is made. Since the number of available kicks is finite, only the kicking angle must be discretized. For any available action \( a'_k \), a resulting position of the ball according to an ideal execution of the action\(^1\) is calculated:

\[ x'_{k,b} = f_{kick}(x_{k-1,b}, a'_k); \quad a'_k \in A_{kick} \]  

(8)

Here, \( k - \in (k-1,k] \) represents the current instant, which probably does not belong to the time discretization, and \( f_{kick}(x_{k-1,b}, a'_k) \) is the ideal final position of the ball after being kicked from point \( x_{k-1,b} \) using the kick and angle contained in \( a'_k \). Given the current state \( x_{k-1} \), for any action \( a'_k \), it is possible to define its expected action value:

\[
O(x_{k-1}, a'_k) = E[V(x_k)]
\]  

(9)

The action value of every available action \( a'_k \) is calculated as the expected value of the state resulting from keeping the robot poses unaltered and only displacing the ball to \( x'_{k,b} \):

\(^1\) This implementation does not consider uncertainty on the action result but considering it is straightforward, for example using the method proposed in [4].
\[ Q(x_{k-1}, a^i_k) = E[V(x^i_k)] : \ x^i_k = (x^i_{k,b}, x^i_{k-R}) \]  

(10)

The value of this resulting situation is approximated using the ball-control MDP Value:

\[ E[V(x^i_k)] = E[V(x^i_{k-b}, c_k)] \]  

(11)

Since the ball-control state is three-dimensional, a linear interpolation from the 8 neighbors in the ball-control state is performed:

\[ \sum_{x_0 \in [10, 10]} \sum_{y_0 \in [10, 10]} \sum_{y_1 \in [10, 10]} V(x_{k,b}^i, c_k) \omega(x_{k,b}^i, c_k, x_{k,y,c}) \]  

(12)

\[ \omega(x_{k,b}^i, c_k, x_{k,y,c}) = \left( 1 - \frac{x - x_{k,b}^i}{x_0 - x_1} \right) \left( 1 - \frac{y - y_{k,b}^i}{y_0 - y_1} \right) p(e|x^i_k) \]  

(13)

Where \( x^i_{k,b} = (x^i_{k,b}, y^i_{k,b}) \), and \( x_0, x_1, y_0, \) and \( y_1 \) are the spatial coordinates of the neighbors of \( (x^i_{k,b}, y^i_{k,b}) \) in the ball-control state space grid.

Finally, the action yielding the maximum expected value is selected.

**F. Role Assignment and Positioning**

Role assignment and positioning are performed in a fashion that is coherent with the kicking decision method. The problem to solve is the following: two kicking robots are going to the ball, one belonging two each team. There is a remaining time until the next instant and the positioning robot is able to move in any direction up to this moment. The proposed method is applicable to the current scale of the RoboCup Standard Platform League (SPL) (3 or 4 robots per team), and it allows: (i) dynamic role assignment and (ii) dynamic number of attackers or defenders, depending on the situation.

For the SPL scale, given that there is a goalie and one partner kicking robot, only 1 or 2 robots are positioning. Also, the ball could move along the whole field length (~6m) with only two kicks. Then, the positioning robots may safely consider only positioning objectives where they can reach the ball in the next instant. Only one of the kicking robots will succeed to kick the ball, and depending on which kicking robot will, the decision of the positioning robot may have radically different results. If the succeeding kicking robot is a partner, then it will kick the ball to the available destination that maximizes the value (attacking situation). In this case, the positioning robot best choice would be to approximate to that point to be able to catch the ball from there. On the other hand, if the succeeding kicking robot is an opponent, then it will kick the ball to the available destination that minimizes the value (defending situation). In this case, the positioning robot best choice would be to approximate to the available minimum to have a chance of intercepting the opposite team pass or shut.

Since the positioning robot does not know a priori which robot will succeed kicking the ball, it cannot be sure of which is the best decision. Then, two scenarios for the next ball position are considered, depending on who will kick the ball:

\[ x_{k,b} = \begin{cases} \text{fkick} & \text{if the succeeding kicking robot is an opponent} \\ \text{at} & \text{if the kicking robot is a partner} \end{cases} \]

(14)

\[ \begin{cases} i_{\max} = \arg \max \ _i Q(x_{k-1}, a^i_k) \quad c_{k-1} = 0 \\ i_{\min} = \arg \min \ _i Q(x_{k-1}, a^i_k) \quad c_{k-1} = 1 \end{cases} : a^i_k \in A_{kick} \]

(15)

We will call \( x_{k,b,max} \) and \( x_{k,b,min} \) the resulting ball positions from selecting respectively \( a^i_{k,max} \) and \( a^i_{k,min} \). For every positioning action \( a^i_k \in A_{pos} \), the ideal resulting pose of the robot is calculated:

\[ x^j_{k,R} = \begin{cases} \text{fpos} & \text{if the resulting ball position is to the right} \\ \text{left} & \text{if the resulting ball position is to the left} \end{cases} \]

(16)

Wore \( me \) is the index of the positioning robot, and \( t_a \) is the estimated positioning remaining time, given the current distance of the kicking robots to the ball. All the other robot poses are considered to remain unaltered\(^2\), i.e.:

\[ x^j_k = \begin{cases} x^j_{k,R} & \text{if the resulting ball position is to the right} \\ x^j_{k,R,me} & \text{if the resulting ball position is to the middle} \\ x^j_{k,R,N_k} & \text{if the resulting ball position is to the left} \end{cases} \]

(17)

Then, for each positioning action \( a^j_k \), the resulting next state \( x^j_k \), has two possible values:

\[ x^j_k = \begin{cases} x^j_{k,max} = (x^j_{k,b,max}, x^j_{k,R}) \quad c_{k-1} = 0 \\ x^j_{k,min} = (x^j_{k,b,min}, x^j_{k,R}) \quad c_{k-1} = 1 \end{cases} \]

(18)

And we can calculate the expected action value of \( a^j_k \):

\[ Q(x_{k-1}, a^j_k) = E[V(x^j_{k,max})]p(c_{k-1} = 0|x_{k-1}) + E[V(x^j_{k,min})]p(c_{k-1} = 1|x_{k-1}) \]

(19)

Where \( E[V(x^j_{k,max})] \) and \( E[V(x^j_{k,min})] \) can be approximated from the ball-control MDP value function following the procedure shown in equations (11)-(13).

**IV. Results**

We have implemented the proposed approach and tested it in simulated soccer matches. Preliminary results show that the proposed approach allows a team of robots to work coherently

---

\(^2\) This positioning decision reasoning could consider a vector of positioning actions where all the positioning robots are included. However, that approach would involve the handling of a high dimensional action space which is beyond the scope of this paper.
and to be efficient in balancing the offensive and defensive tasks.

With the purpose of testing our world modeling and decision making modules, we have developed a 2D simulator in which vision, actuation, and optionally world modeling, are replaced by perfect versions of them which are noise free or have controlled noise. Every robot decision making module works in a different process. The graphic interface of the simulator and its engine (2D dynamic library, game flow, refereeing, etc.) run in a single process which communicates to the robot processes via sockets (allowing the robot clients to run in different computers). By having this parallelism, the simulator is able to take into account the time that the robot expends in processing. We have performed several simulated games and the proposed method seems to make, in real time, reasonable decisions almost all the time.

In the presented experiments, the robots have 4 available kicks that can move the ball to 4 different distances. The angle of the kick is discretized using 24 angle intervals (of 15°). Then, the kicking robot has 96 possible actions and each of them are evaluated. Each action is represented as a small square around the ball in the ideal destination position of the ball.

Figures 2 and 3 show some screenshots, from the mentioned simulator, of game situations and the consequent decisions. In the shown screenshots the analyzed team is the red one. The robots are represented with a white rectangle and two colored circles, having the color of the robot’s team. An arrow shows the forward orientation of each robot. One of the red robots, marked with a “K” letter is the kicking robot. The other red robot, marked with a “P” letter is the positioning robot.

The gray intensity of the square represents the expected value of the kicking action. So, the kicking robot should select the action represented by the lighter square. Regarding the positioning decision, the selected maximum is marked with a big red square, while the selected minimum is marked with a big blue square. A big arrow marks the selected destination of the robot.

Figure 2 shows a situation where attacking positioning is selected. First, the red kicking robot selects a pass in advance to its partner. The objective of this pass (marked with a red square) seems to make both desired events to have a high probability: (i) that the ball will be caught by the red positioning robot, and (ii) then a goal will be scored from that position. Second, from the point of view of the positioning robot, there are two choices: making a defensive move towards the blue square or an offensive move looking for the pass in advance from the kicking robot. Since the red kicking robot is closer to the ball than the blue one, the first term in equation (19) is stronger than the second one. Then, the positioning robot selects a position close to the red square. Figure 3 shows two situations where the positioning robot decides to defend. In both situations, the blue positioning robot is closer to the ball than the red one. In those cases, the second term in equation (19) becomes more preponderant than the first one. In both cases, the defensive choice seems to be the right one since it is more likely than the opponent will catch the ball first.

V. CONCLUSIONS

An integrated multi-agent decision making framework for
robot soccer has been proposed. The current implementation of the proposed method does not consider uncertainty of any kind, however it is straightforward to include this feature using sampling and perhaps simple methods for solving POMDPs like QMDP [3].

We are planning to improve the system by letting it learn several parts of the model which are now selected arbitrarily. For example, the heuristic transition pdf could be replaced by a learned one.

Other directions in which the method should be improved is efficiency. In that direction we are planning to parallelize the computation of the ball-control MDP and make the stronger calculations when other threads are not so charged. Another interesting research direction would be to find out how the efficiency and accuracy of the ball-control and full MDPs could be improved by allowing the different robots of a team to calculate them cooperatively.

Finally, a desirably extension of the proposed positioning method would be making it capable of dealing with teams having more robots. A possible way of extending it toward this direction would be to allow the positioning robots to move to the local maxima and minima of the whole ball-control state space and not only to its reachable subspace.

REFERENCES


