MDP Soccer Simulation using RDDL-based Monte Carlo Tree Search

Sana Arastehfar[1], Christian Muise[1]

School of Computing, Queen's University, Kingston, Canada

Abstract

In this paper, we discuss the use of Markov Decision Process (MDP) models in automated planning for modelling a soccer game. The MDP model captures the dynamic and uncertain nature of the game, enabling the generation of optimal plans for teams under different scenarios and conditions. The objective of our work is to find the optimal plan for a soccer team to score a goal, while in possession of the ball. Here, we model the players of a team as agents who can take different actions. However, we focus on devising the states such that reward signal evaluates the performance of the team rather than a single player. Moreover, we avoid to explicitly model the opponent and instead, we incorporate the effect of opponents through different lose-possession probabilities. Also, to more accurately model the game, we differentiate the games played in different regions of the pitch, by dividing the pitch to five different areas for which, the lose-possession probabilities are different. Finally, we implement our model using RDDL language and evaluate the performance of the model in a finite-horizon fashion in different scenarios. To effectively and efficiently find the plans, we also implement a Monte-Carlo Tree Search algorithm which employs the RDDL implementation to find a winning plan which ends in a goal for the team.

1. Introduction

Automated planning is concerned with designing and implementing algorithms that can automatically generate plans to achieve specific goals. One popular approach to planning is through the use of Markov Decision Process (MDP) models, which provide a flexible and powerful framework for modeling decision-making in dynamic environments with uncertain outcomes. In recent years, there has been growing interest in using MDP to model and plan strategies for real-world applications. One of the applications of MDPs is in the domain of sports such as soccer[1], ice hockey [2], basketball [3], where it can be used to plan strategies and tactics for teams. By modeling the game as an MDP, it is possible to account for the uncertainty and dynamic nature of the game. MDP-based planners can generate plans that take into account the strengths and weaknesses of the opposing team, as well as the abilities of individual players. Moreover, MDP-based planners can generate plans that are robust to changes in the environment.

Here we model a soccer game in a Markov Decision Process (MDP) formalism and players as planning agents. the MDP formalism allows us to capture the stochastic and uncertainty inherent in the game of soccer, and plan the optimal sequence of actions that the team should take to achieve its objective. To accomplish this, we propose a simulation model that can be used to generate plans for the team under different scenarios and conditions.

The objective of our work is to find an optimal plan for a soccer team to score a goal while in possession of the ball. We model the soccer game as an MDP, where the state of the game is represented as a vector containing the coordinates of the players, the id of the player who possesses the ball, a goal indicator which indicates whether the team has scored a goal, and the lost-possession indicator which demonstrates whether the team has lost the possession of the ball. We define the action set to be pass, shoot, move, and the transition probabilities to be encapsulated in a lose-possession probability, which could vary for different regions of the pitch. The reward function is defined as zero unless the team scores a goal, in which case the reward is one. Finally, we implement a customized Monte-Carlo Tree Search (MCTS) algorithm from scratch, which uses the RDDL description to simulate the domain and environment and finds a winning plan which ends with a goal for the team.

It is important to note that in this project, we do not explicitly model the opponent team, but rather, we implicitly incorporate the effect of an opponent team through the definition of lose-possession probabilities. The reason that we employ these scheme for modelling the game is to simplify the modelling and avoid min-max adversarial optimization. More precisely, in the presence of an opponent team, one has to modify the model that the home team minimizes the maximum benefit/goals of the opponent. In other words, the home team must find a strategy that beats the best strategy of the opponent. In case of ignoring such considerations, the planning model might fall into a wrong solution in which the opponent team is planned to behave against their own favor (for example, by scoring own goals).

The rest of this paper is structured as follows. The next section is the background 2 where MDP formalism will be discussed, Model Definition 3, in which we talk about state and action representation, transition probabilities and reward, Evaluation of the model will be presented in

CISC-813: Automated Planning – Final Project

section 4, related work and summary will be discussed in the sections 5 and 6, respectively.

2. Background

Automated planning is a well-established research area that seeks to develop algorithms and techniques for generating plans that satisfy complex goals and constraints. In recent years, there has been growing interest in using Markov Decision Processes (MDPs) as a formalism for automated planning. MDPs provide a flexible and powerful framework for modeling decision-making in dynamic environments with uncertain outcomes.

Markov Decision Processes (MDPs) are a popular formalism for modeling decision-making in dynamic environments. MDPs are a mathematical framework that provide a way to model decision-making under uncertainty. They consist of a set of states, a set of actions, a set of rewards, a set of transition probability functions, and (possibly) a discount factor, taking the effects of future decisions into account. The decision-maker (or agent) takes an action in a state and the environment transitions to a new state with a probability that depends on the current state and the action taken. The agent also receives a reward that depends on the current state and the action taken. The goal of the agent is to find the best action possible for each state (a policy) that maximizes the expected sum of rewards over time.

The formal definition of an MDP is as follows:

- $\mathcal{S}$: A set of states,
- $\mathcal{T}$: A state transition function, where $\mathcal{T}(s, a, s') = \mathbb{P}(s'|s, a)$ is the probability of transitioning to state s' from state s when action a is taken,
- $\mathcal{R}$: A reward function, where $\mathcal{R}(s, a, s')$ is the reward received when transitioning from state s to state s' when action a is taken,

The goal of an MDP is to find a policy π that maximizes the expected total reward obtained over a sequence of actions. The expected total reward of a policy starting from a state s is defined as:

$$V_\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1}|s_0 = s, a_t \sim \pi)] \quad (1)$$

where γ is the discount factor that determines the importance of immediate rewards relative to future rewards.

The use of MDPs as a formalism for automated planning offers several advantages over other approaches. MDPs can model complex decision-making problems in a flexible and transparent way, and provide a formal framework for reasoning about uncertainty and risk. Moreover, MDPs can be used to generate plans that are robust to changes in the environment, such as changes in resource availability or changes in the goals or constraints.

The objective of this study is to develop an MDP for soccer games. More precisely, we aim to appropriately define the state-action space of the MDP suitable for analysis of a soccer game. The planner will take as input the current state of the game, including the positions of the players and the location of the ball, and is supposed to generate a sequence of actions that the players should take to achieve the goal of scoring a goal. The ultimate goal is to find an optimal plan that maximizes the probability of scoring a goal.

Although this study is a course project and has no immediate practical significance, it has the potential to contribute to the growing body of research on using MDPs for soccer planning. The development of an MDP-based planner for soccer games could lead to improved game strategies and better player performance. Moreover, the use of MDPs in soccer planning could have applications in other areas, such as robotics and autonomous systems.

3. Model Definition

A Markov Decision Process model could be defined as a tuple of the form $(\mathcal{S}, \mathcal{A}, \mathbb{P}, r, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions, $\mathbb{P}$ is the transition probability function, $r : \mathcal{S} \times \mathcal{A} \longrightarrow \{0, 1\}$ is the reward function and $\gamma \in [0, 1]$ is the discount factor. Here, we intend to provide a sketch of how the state space, actions, transition probabilities and the reward function look like.

3.1. State Representations

Each state of the game can be modelled as a vector containing the coordinates of the players, the id of the player who possesses the ball, a goal indicator which indicates whether the team has scored a goal, and the lost-possession indicator which demonstrates whether the team has lost the possession of the ball. For example, for a team with three players where the second player possesses the ball, and they have not scored a goal yet, we can represent a state as

$$\forall s \in \mathcal{S},\ s = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ \text{ball} \\ \text{goal} \\ \text{lost-ball} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \\ 2 \\ 3 \\ 2 \\ 0 \\ 0 \end{pmatrix}. \quad (2)$$

The values that x_i, y_i can take are directly related to how we discretize the pitch into a grid; the refiner the grid is, the more number of values x_i, y_i can take, and the larger the state space will be.

In our simulation, we define the terminal states to be the ones in which either the team has scored a goal, or the team has lost the possession of the ball. Moreover, there are some invalid states that we handle them in our RDDL implementation. Such invalid states are those that the coordinates of two players overlap. Also, we assume that one of the players is the goalkeeper and cannot change his/her position.

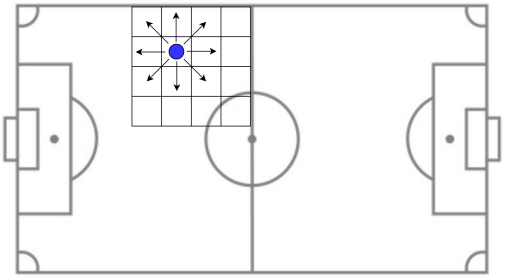

Figure 1: The allowed move actions for a single player in the pitch.

3.2. Action Representations

For this project, we define the action set to be $\{pass, shoot, move\}$. For simplicity, we assume that the end location of the pass is always the location of one of the players. In addition, we assume that shoot is always intended for a goal and if it is unsuccessful, the possession is lost. Also, we assume that the players' movement can always be in a vicinity of one cell in the grid (Figure (1).

3.3. Transition Probabilities

As mentioned in the Introduction section, explicitly modelling the opponent renders the problem as an adversary min-max problem. Therefore, to simplify the problem, we encapsulate the effect of the opponent in a *lose possession* probability, which could vary for different regions of the pitch. In this project, we aim to define five different lose-possession probabilities (Figure (2))

In fact, these lose-possession probabilities are directly used for modelling the transition probabilities. First, we should note that the transition probabilities are expressed as

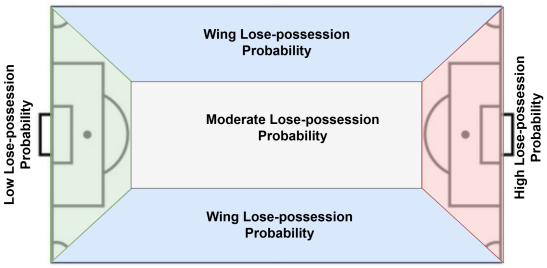

Figure 2: The partition of the pitch into five different regions with different lose-possession probabilities. We assume that our team moves and attacks from left to right.

$$\begin{aligned} &\mathbb{P}(s, pass, s') \\ &\mathbb{P}(s, shoot, s') \\ &\mathbb{P}(s, move, s'). \end{aligned} \tag{3}$$

The lose-possession probability shows itself in $\mathbb{P}(s, move, s')$, where s' is a state for which the lost-ball attribute is true (let's represent such a state as lost-possession). In this case, we can write

$$\mathbb{P}(s, move, \text{lost-possession}) = \text{loss-possession probability}. \tag{4}$$

Depending on the coordinate of the ball in the sate s we can determine which loss-possession probability we should use.

Furthermore, we define the transition probability of the action *shoot* only when the next state is a goal. In such a case, define a not-goal probability for the shoot and we can write

$$\mathbb{P}(s, shoot, goal) = 1 - \text{not-goal probability}. \tag{5}$$

Similarly, we can define an interception probability for the passes and we can write

$$\mathbb{P}(s, pass, s') = 1 - \text{interception probability}. \tag{6}$$

3.4. Reward

The reward function is defined as $r : \mathcal{S} \times \mathcal{A} \longrightarrow \{0, 1\}$. For this simulation, unless we have reached to a state for which the goal attribute is true (i.e., the team has scored a goal), the reward is zero. If the team scores a goal, we give a reward of 1 to the team.

3.5. Monte-Carlo Tree Search

In this research, we have also implemented the Monte-Carlo Tree Search (MCTS) method for planning, which is mounted on top of the RDDL simulator. Briefly, MCTS starts with an initial state and sets it as the root of a tree. Each node in the tree represents a state in the state space of the problem and a parent node is connected to a child node if there is an action taking us from the state of the parent node to the state of the child node. The important steps of the MCTS algorithm is as follows:

1. **Selection** The selection means that which child node should be selected from a parent node. Equivalently, selection demonstrates which action should be taken from a given state. MCTS algorithms usually use the *Upper Confidence Bound* formula for the selection phase which is defined as

$$UCB(i) = \frac{w_i}{n_i} + c.\sqrt{2\frac{\log(n)}{n_i}}, \qquad (7)$$

   where w_i denotes the number of wins starting from node i, n_i denotes the number of times node i has been visited, n denotes the total number of simulations, and c is a constant called *expansion parameter*. In our experiments, we have set $c = 1$ for each experiment.
2. **Expansion** When we cannot find the successor node using UCB (e.g, we have reached a leaf node), we need to expand the tree from that node. The expansion is carried out by appending all the successors of the node.
3. **Simulation** After expansion, a simulation step is followed. This simulation is usually carried out using a random rollout procedure. Rollout procedure starts by randomly selecting a feasible action, and performs a random walk until it reaches a terminal state.
4. **Backpropagation** Once a terminal state is reached the tree path is traversed in reverse to update the number of wins for all the intermediate nodes. In this way, we value of each state is updated and using UCB becomes viable again.

Since, the state space of a soccer game could exponentially grow, it is essential to use a sampling method to perform the planning. Monte-Carlo Tree Search has shown to be a strong and robust tool for problems with huge state space, and we use the same algorithm here to come up with a goal-ended planning in a soccer game.

4. Evaluation

4.1. Experimental Settings

To evaluate the proposed model, we define the soccer pitch as a 11×11 grid and defined 3 players in the pitch with certain initial positions. We use RDDL to define the planning domain and instance. Then, we use **pyRDDLGym** to simulate the RDDL domain/instance. This simulation is used in the heart of our MCTS implementation. More precisely, MCTS uses RDDL for simulations and state transitions. We explicitly specify in the planning domain that a player is allowed to shoot when they are in the offense area. This assumption is both realistic and makes the planning much simpler.

Also, to determine the different regions of the pitch, we specify the four corners of the midfield region as (-2, 2), (2, 2), (2, -2), and (-2, -2). Using this four corner points the planner determines the region the ball-possessing player is located. Subsequently, knowing the region, the designated pass-interception, shoot-success, and lose-possession probabilities can be applied.

For a given initial state we examine the performance of the model with different shoot success, pass interception, and lose possession probabilities. Here, we examine the performance of the model in terms of the number of successful outputs of the MCTS algorithm. More precisely, for each setting, we run the MCTS algorithm 20 times and count the number of times MCTS has been successful outputting a goal-ended plan.

4.2. Experiments

For the first series of the experiments, we fix the initial positioning of the players as illustrated in Figure (3). In this setting the players are located at coordinates (1, 0), (1, 1), (1, -1). In the initial state we assume that the ball is initially owned by the player at (1, 0).

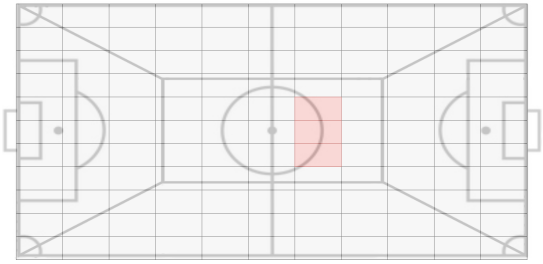

Figure 3: The initial positioning of the players. The location of players are shown by red cells.

Given the initial state illustrated in Figure (3), we set the shoot success probability to 1; i.e., if a player who possesses the ball resides in the offense area and decides

to shoot, it will score a goal with probability 1. From 20 times of running MCTS on this setting, MCTS could produce a winning plan 16 times (see Figure(4)).

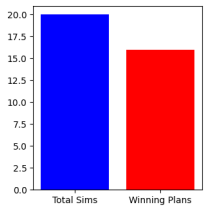

Figure 4: The number of successful plannings of MCTS with shoot success proability of 1.

Also, Figure (5) illustrates two of terminal states reached by the model which has ended in a goal.

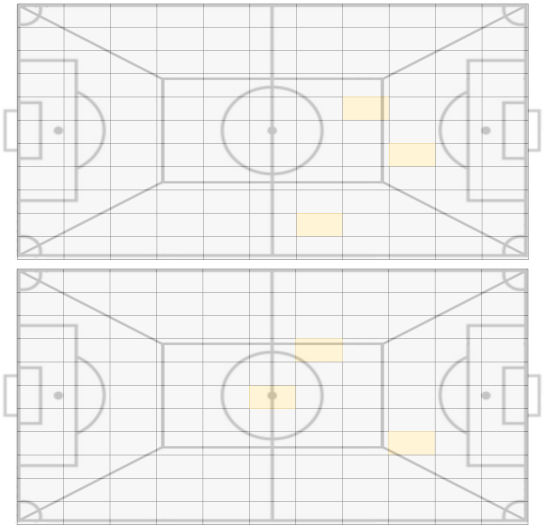

Figure 5: Example of the planning result when the shoot success probability is 1. Players locations are shown by yellow cells.

The results of the planning suggest that in this setting the planner tries to make a shot whenever the player resides in the offense area. This is intuitively correct since the shoot success probability is 1. We repeat the same experiment, with shoot success probabilities 0.8, 0.6, 0.4, where the corresponding rate of generating a successful plan is 14, 7, and 7 times out of 20 tries (Figure (6)). This demonstrates that when the shoot success probability gets lower (and indication of a stronger opponent), it also becomes harder to score a goal.

Figures (7) and (8) show examples of final configuration of players for a successful plan when the shoot success probability is 0.6 and 0.4, respectively. We can see that in such cases, the planner tries to move a player deep into the offense area for scoring a goal.

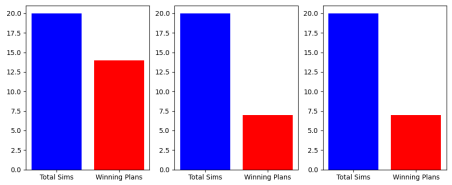

Figure 6: The success rate when the shoot success probability is 0.8 (left), 0.6 (middle), and 0.4 (right).

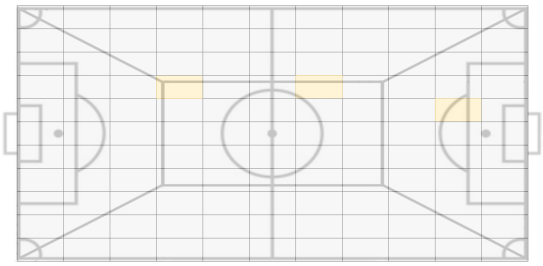

Figure 7: Example of the planning result when the shoot success probability is 0.6. Players locations are shown by yellow cells.

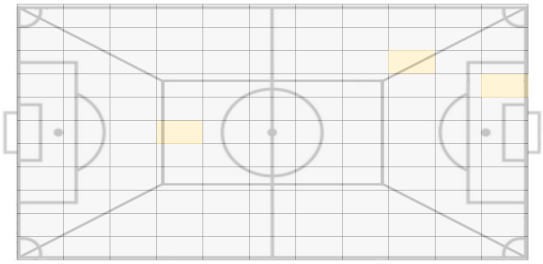

Figure 8: Example of the planning result when the shoot success probability is 0.4. Players locations are shown by yellow cells.

We follow the same procedure to examine the effect of pass interception probability. To this end, we revert the shoot success probability to 1, and use the same initial state as in Figure (3). Here, we study the effect of pass interception probability for two different probabilities, 0.2, and 0.4. Our studies reveal that the models is highly sensitive to the probability of a pass being intercepted (Figure (9)).

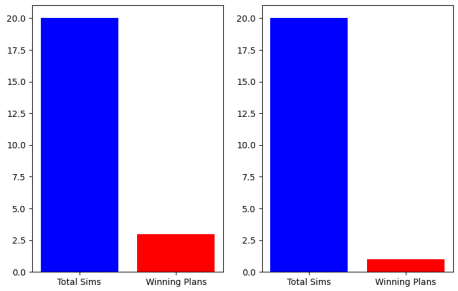

Figure 9: The success rate when the pass interception probability is 0.2 (left), and 0.4 (right).

When the pass interception probability is 0.2, only 3 times out of 20 tries lead to a successful plan. This number becomes 1 when the pass intercept probability increases to 0.4. For higher probabilities of pass interception the planner fails to find a successful winning plane. This matches our belief in reality; since, a football game is heavily reliant on passing, if a team's passes are intercepted frequently, the chances of that team winning would drop significantly.

The procedure is the same as above for examining the effect of lose possession probabilities. We reset the initial state to as in Figure (3), set the pass interception probabilities to 0.0, set the shoot success probability to 1, and evaluate the performance of the model using two different probability values 0.2 and 0.4. For these probability values the planner can find successful plans 5 and 4 times out of 20 tries, respectively (Figure (10)).

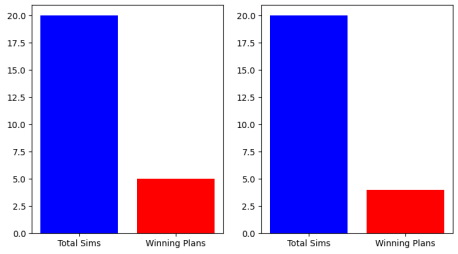

Figure 10: The success rate when the lose-possession probability is 0.2 (left), and 0.4 (right).

The empirical studies show that the model is still highly sensitive to the lose-possession probabilities. However, this sensitivity seems to be smaller than with respect to pass interception probability.

In the final step of our experiments, we investigate the effect of initial state or the initial positioning of the player. To this end we examine two other initial states by changing the x-coordinates of the players. In the first new initial state (Figure (11)), the coordinates of the players are (2, 0), (2, 1), (2, -1) and the players are closer to the opponent's goal. In the second new initial state (Figure (12)), the coordinates of the players are (-1, 0), (-1, 1), (-1, -1), and the players are farther from the opponent's goal.

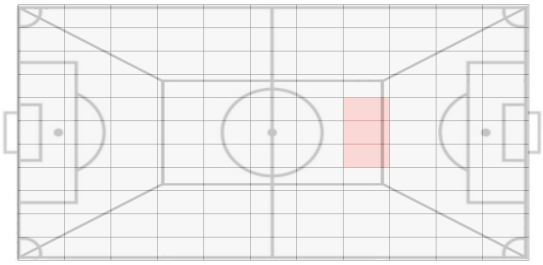

Figure 11: The initial positioning of the players at (2, 0), (2, 1), (2, -1). The location of players are shown by red cells.

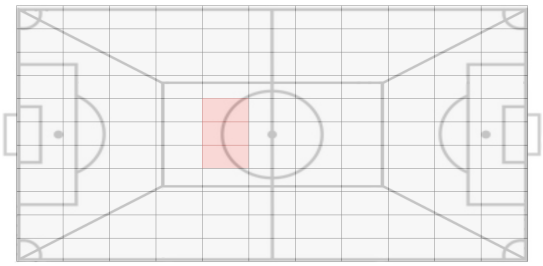

Figure 12: The initial positioning of the players at (-1, 0), (-1, 1), (-1, -1).

In the initial state where the players are located at coordinates (2, 0), (2, 1), (2, -1), we expect that the planning is more successful since the configuration is closer to the goal. Also, when the initial state dictates the initial coordinates of the player at (-1, 0), (-1, 1), (-1, -1), we expect that the planner fails more often since the configuration has become more distant to the goal. Our experiments in fact corroborate this intuition; for the closer configuration the planner gets 19 successful plans out of 20 tries, and for the farther configuration the planner can get only 11 successful plans (Figure (13)).

5. Related Work

Recent studies have demonstrated considerable improvements in the application of automated planning methods and MDP models for cooperative multi-agent systems.

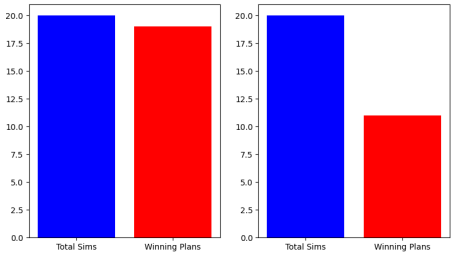

Figure 13: The success rate when the initial state is (2, 0), (2, 1), (2, -1) (left), and (-1, 0), (-1, 1), (-1, -1) (right).

These techniques have been used to solve planning problems in various domains, including robotics, transportation, and gaming, and have also been applied to sports such as soccer.

The authors in [4] present a Markov Decision Process (MDP) to model professional soccer matches to aid in-game decision making. The paper uses streaming data of data from the 17/18 and 18/19 English Premier League seasons. The paper addresses the challenges posed by working with such data, including the observational nature of the data, sparsity issues, and difficulties in evaluating and validating the learned models. Van Roy et. al. [4], propose an outline of a framework for learning the MDP model, which includes a predictive model to predict the intended end locations of players' actions (e.g., passes and shoots), a hierarchical Bayesian approach to learn the transition model, and an approach to evaluate and validate the final model. They provide three illustrative use cases using the English Premier League data, which demonstrate how the MDP model can be used to gain insights into shooting behavior, assess changes to in-game decision making, and rate players based on their decision making.

Furthermore, the authors in [5] introduced a novel and real-world application for verifying properties of learned models in the context of sports, specifically soccer. They proposed a framework for using automated planning approaches to model and simulate soccer matches from a defensive perspective. The authors suggest that Markov Decision Processes (MDPs) can be used to model the dynamics of soccer matches and schedule player actions. The paper focuses on using probabilistic model checking techniques to reason about the goal-directed policies of opponents, providing insights into areas such as predicting match outcomes, evaluating decisions, and rating players. The authors propose a two-step process that involves learning a team-specific MDP from data and using probabilistic model checking to reason about the learned policy. The paper also provides a number of illustrative examples of tactical insights about teams in the English Premier League dataset.

Ahmadi et. al. [6] discuss the challenges of planning on-board robotic agents and introduce a new approach for robot action planning in dynamic environments with uncertain action effects. Specifically, the authors introduce an instance-based action model that can capture arbitrary distributions of action effects, and an efficient online incremental re-planning method that can modify the transition model to account for the effects of other agents and then re-plan only for the affected states. The paper also compares the performance of the instance-based action model with the more popular parametric action models on a physical robot using a goal scoring task from the 4-legged RoboCup competitions. The main contributions of the paper are a dynamic re-planning approach based on an instance-based representation of action effects and an empirical comparison of instance-based and parametric action models. The paper provides detailed descriptions of the proposed methods and experimental results to demonstrate their effectiveness. In [7] proposes a multi-agent learning approach for generating offensive plays in the Rush 2008 football simulator. The authors demonstrate how to automatically identify subgroups from historical play data based on mutual information and ball work flow. They then use these subgroups to focus the search for new plays using the UCT algorithm, which has shown promising results in games with large search spaces. The paper highlights the importance of considering the coordinated efforts of different subgroups within a team to successfully execute plays in American football. The authors in [8] reviews 77 different methods for evaluating collective tactical behaviors and highlights the importance of ball possession football. Although extensively investigated, the relationship between ball possession and the outcome of soccer matches remains uncertain. The paper suggests that simulation-based approaches offer a useful tool to study the complexity of tactical behavior in soccer. Mathematical approaches like Monte Carlo simulation and game theory can be used to generate optimal strategic patterns. The paper proposes the mediation of player group-specific tactical variations as a practical approach to studying tactical behavior in soccer.

6. Summary

In this paper, we proposed to simulate a soccer match using the Markov Decision Process (MDP) model, where the state of the game is defined as a vector including the player positions, the player holding the ball, goal and lost-possession indications. We also defined the action set of the MDP to be "pass", "shoot", and "move," where "pass" refers to an action that is always intended for one

of the players and "shoot" refers to an action that is always intended to score a goal. The players' movements are always permitted to be close to one grid cell. The definition of the transition probability function is based on five distinct lose-possession probabilities, which differ for various areas of the pitch. The effect of the opponent is encapsulated in such lose-possession probabilities to simplify the problem. If the team achieves a goal, the reward function is defined as 1; otherwise, it is defined as 0. The MDP model might be utilized to address the issue of determining an effective team strategy in a football game.

The reward function can be used to assess the effectiveness of the team's strategy, and the proposed MDP model can be used to simulate soccer games. By resolving the MDP problem, the MDP model can also be used to determine a team's best course of action during a soccer game. To do this, one must identify the best course of action that maximizes expected total benefit over an infinite horizon; however, we conducted our experiments here in finite horizons. To come up with a plan, we implemented a Monte-Carlo Tree Search algorithm on top of the RDDL simulator to effectively explore the state space of the MDP and find a winning plan for the team which ends with a goal for the team. The presented MDP model can be extended by including more complex actions, such as dribble or tackle, and by considering more advanced opponent modelling techniques.

6.1. Future Work

As mentioned in the Model Definition section, the players have three different actions, namely, move, pass, and shoot. However, in a real soccer game, there are other actions such as cross and dribble as well. For the first step of our future works, we intend to extend the set of actions of the players to make the simulation more realistically.

As demonstrated in the experiments, the model is sensitive to the values of pass-interception and lose-possession probabilities. This indicates that a good value of such probabilities is crucial for a reliable planning. Therefore, a possible thread of research could be to incorporate neural networks to learn these probabilities from available games. In this way, one could even extend the delineation of the pitch to define more fine-detailed regions and learn their corresponding pass-interception and lose-possession probabilities using the data of available games.

Even though assumption of distilling the opponent's effect into pass-interception and lose-possession probabilities facilitates the modelling of the game and might be useful for rudimentary modelling of a soccer game, it might be a naive approach since the actual lose-possession probabilities are highly dependent to other factors such as the positions of opponent players (which change throughout the game). More precisely, if there are opponents close to a player carrying the ball, the chance of that player losing the ball or having his/her pass intercepted increases. Hence, the biggest future step for this paper is to explicitly model the opponents and simultaneously learn the best strategy for the opponent as well as the team we are analyzing(i.e., solving the min-max problem to find the best strategy of our team).

References

[1] N. Hirotsu, M. Wright, Using a markov process model of an association football match to determine the optimal timing of substitution and tactical decisions, Journal of the Operational Research Society 53 (2002) 88–96.

[2] O. Schulte, M. Khademi, S. Gholami, Z. Zhao, M. Javan, P. Desaulniers, A markov game model for valuing actions, locations, and team performance in ice hockey, Data Mining and Knowledge Discovery 31 (2017) 1735–1757.

[3] D. Cervone, A. D'Amour, L. Bornn, K. Goldsberry, A multiresolution stochastic process model for predicting basketball possession outcomes, Journal of the American Statistical Association 111 (2016) 585–599.

[4] M. Van Roy, P. Robberechts, W.-C. Yang, L. De Raedt, J. Davis, Learning a markov model for evaluating soccer decision making, in: Reinforcement Learning for Real Life (RL4RealLife) Workshop at ICML 2021, 2021.

[5] M. Van Roy, W.-C. Yang, L. De Raedt, J. Davis, Analyzing learned markov decision processes using model checking for providing tactical advice in professional soccer, in: AI for Sports Analytics (AISA) Workshop at IJCAI 2021, 2021.

[6] M. Ahmadi, P. Stone, Instance-based action models for fast action planning, in: RoboCup 2007: Robot Soccer World Cup XI 11, Springer, 2008, pp. 1–16.

[7] M. Arbatzat, S. Freitag, M. Fricke, R. Hafner, C. Heermann, K. Hegelich, A. Krause, J. Krüger, M. Lauer, M. Lewandowski, et al., Creating a robot soccer team from scratch: the brainstormers tribots, Proc. of Robocup 2003, Padua (2003).

[8] J. Perl, J. Imkamp, D. Memmert, Strictness vs. flexibility: Simulation-based recognition of strategies and its success in soccer, International Journal of Computer Science in Sport 20 (2021) 43–54. URL: https://doi.org/10.2478/ijcss-2021-0003. doi:doi:10.2478/ijcss-2021-0003.