MULTI AGENT REINFORCEMENT LEARNING

by

Berk DAĞLI
Hüseyincan ERBAYRAKTAR

CSE4097 / CSE4098 Engineering Project report submitted to Faculty of Engineering
in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE

Supervised by:
Assoc. Prof. Borahan TÜMER

Marmara University, Faculty of Engineering

Computer Engineering Department

2019

MULTI AGENT REINFORCEMENT LEARNING

by

Berk DAĞLI
Hüseyincan ERBAYRAKTAR

CSE4097 / CSE4098 Engineering Project report submitted to Faculty of Engineering
in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE

Supervised by:
Assoc. Prof. Borahan TÜMER

Marmara University, Faculty of Engineering

Computer Engineering Department

2019

ABSTRACT

Reinforcement Learning (RL) is a learning approach which is similar to the learning in real life. In the nature, all living creatures take actions and learn within limits of the feedback from their environments. The classic RL algorithms, unlike in multi-agent systems, involve a single agent which interacts with an environment. In multi-agent systems, there are multiple agents affecting the environment. Therefore multi-agent reinforcement learning (MARL) has been an area of study attracting our attention.

In this work, we attempted to show the superiority of MARL approach to the single-agent reinforcement learning (SARL) approach in 2D soccer environment a suitable simulator for multi-agent systems. To do this, we have developed an algorithm based on MARL approach. Finally we compared this algorithm with the SARL algorithm in soccer environment.

ACKNOWLEDGEMENTS

First of all, we would like to thank our advisor Assoc. Prof. Borahan Tümer for his support, advises and providing of opportunities during our work.

Also we would like to thank Kutalmış Coşkun for his efforts in helping us during training our algorithm.

Also we would like to thank Stackoverflow community for answering questions about the topic.

Finally we would thank our families and friends for their support and encouragement during the project.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Reinforcement Learning (RL) is a learning approach similar to learning in real life. All mammals including humans take actions and learn from the feedback coming from their environments. Classic RL algorithms with a single agent proved promising in solving many complex problems where sequential decision making is involved. However, many tasks in the nature require teamwork and are mostly done under the adversary intentions of another team. This means that fulfilling such tasks require coming up with strategies involving cooperation of multiple agents within the same team and competition with another set of agents from an adversary team. Inspiring from such tasks in the nature solved by mammals, complex problems involving multiple agents necessitates devising solutions with multiple agents that both cooperate and compete with other agents, in respective order, from the same team and the other team. Multi-agent Reinforcement Learning (MARL) is one such paradigm that establishes the necessary framework. In MARL, there are multiple agents interacting with the environment. Those agents can also interact with each other, which means they can either be cooperative or competitive with each other.

1.1. Problem Description and Motivation

In RL, most of the time, the problem involves a single agent interacting with an static environment as seen in Figure 1. But in the real world problems, the environment is most of the time dynamic with multiple agents. Sometimes a single task can be too complex for a single agent to deal with. In that case they can handle it cooperatively. An example from nature is wolves hunting in packs. Wolves in a pack are individually same but their tasks during the hunt is different. By combining these different tasks, they can handle more difficult tasks. With this collaboration, they have become one of the most successful animals in hunting.

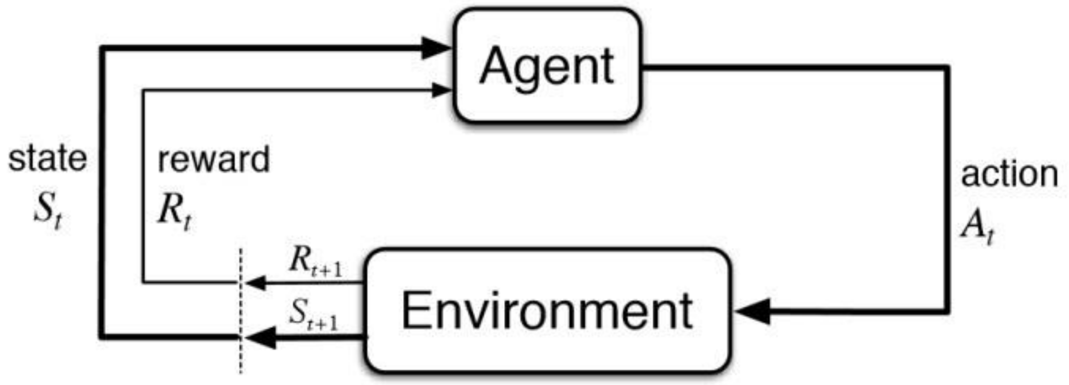


Figure 1: SARL Diagram

In multi-agent systems(MASs), it is considered that there are many agents interacting with the environment as seen in Figure 2. Therefore MASs contain agents that act cooperatively, competitively, independently or in a combination of these.

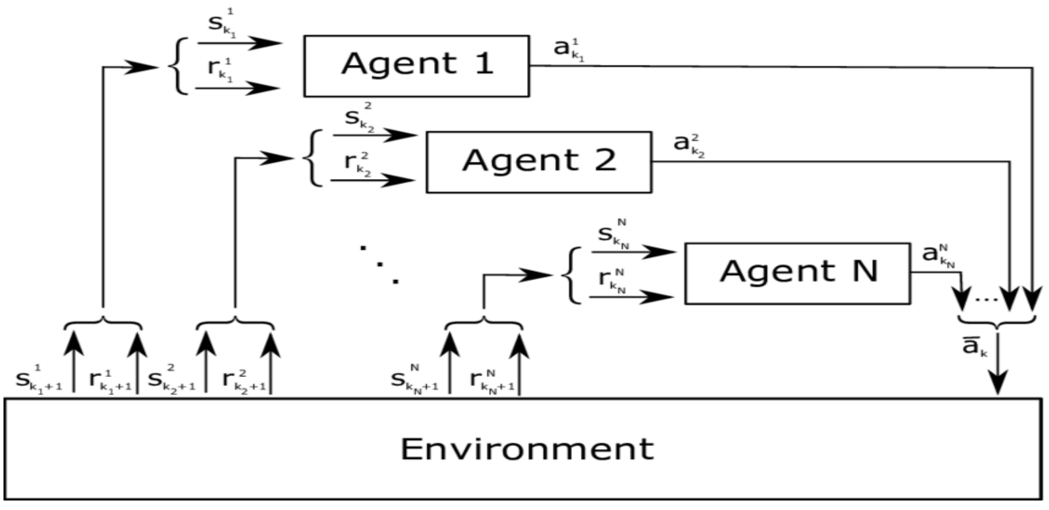


Figure 2: MARL Diagram

In order to apply RL approach to the real world problems, we developed appropriate RL algorithms for it. As we stated above, there are lots of real world MASs for example manufacturing automation systems, air traffic control, medical diagnosis, routers in the internet etc[15]. In these kind of systems, agents can be developed independently of each other and can be later used as components of new systems. For example in medical diagnosis systems, different agents would specialize in different domains and then can be used together with each other[15]. If we want to solve a problem in any MAS, using MARL would be more appropriate. Because MARL takes cooperative and competitive behaviours into consideration. In these systems, agents can benefit from game theory to select their actions. That means they consider the possible actions of the other agents in the environment.

		Pl 2	
		Left	Right
Pl 1	Left	4,2	6,2
	Right	3,3	2,5

Figure 3: Example of Game Theory

For example in Figure 3, its assumed that player 1 and player 2 are opponents. the numbers represent gain of player 1 and player 2 respectively. Player 1 assumes that player 2 would take right action because it gains 5 which is maximum for player 2 therefore player 1 will select left action to minimize player 2's gain and to maximize its own gain. This scenario is an example of competitive behaviour in a MAS.

In our work, we showed that MARL with interacting agents, which is our second approach, gives better results than the Single-agent RL where agents are independent, which is our first approach, in MAS. We studied these two approaches to RL using a soccer simulation as it is a good example where players in the same team behave cooperatively while opposing teams show competitive behaviour. Therefore a soccer simulation was a suitable way to show the superiority of MARL.

In our work we planned to make the agents cooperate to accomplish tasks that can not be effectively handled with independent agents.

1.2. Aims of the Project

- Implementation of 2D Soccer Simulation where teams consist of two players
- Implementation of SARL in which agents act selfish
- Implementation of MARL in which agents learn from the feedback of the coach
- Implementation of a heuristic for the reward function of the environment
- Integration of these implementations
- Performance comparison of MARL and SARL by speed of learning and playing strength

2. PROJECT DEFINITION

2.1. Scope of the Project

In our work we showed superiority of MARL approach only in soccer environment. Our work is focused on soccer environment. The environment is fully observable by all agents and deterministic, which means transition probability of an action is 1. Also the environment is dynamic for the reason that there are multiple agents changing the environment. This provides a dynamism.

The actions of the agents are same in both MARL and SARL approaches except for the pass action which is specific to MARL approach. The state space is continuous for both approaches because there are lots of state action pairs. In both approaches, the state-action pairs consist of state representation and action of the agent.

There is a decision maker "coach" in MARL which gets top N actions from each cooperative agent in the team and determines all possible actions of agents in opponent team. It evaluates all the combinations of actions of cooperative and competitive agents. Finally it determines cooperative agents' actions which maximizes the team reward.

The simulation that we are using is two dimensional. All the agents have identical features. The agents are represented with circles. In the simulation, agents can turn anywhere, and can kick the ball onward where they are facing. An episode ends when one of the team scores a goal.

In the both approaches, the number of agents are fixed and we do not guarantee that it is applicable to the number exceeding that.

2.2. Success Factors and Benefits

Success of our work is measured in two factors:

First one is the speed of learning. Both MARL and SARL approaches are trained for a certain number of episodes and the increase in the rewards are observed.

Second one is the playing strength. Both MARL and SARL approaches are trained until their average rewards per episode are converged. After that they are raced against each other 100 times. The team that scores more is considered more successful.

In case of a success in MARL approach, the MARL in soccer environment could also be applied to other multi agent systems just making small changes in state-action pairs and actions.

2.3. Professional Considerations

- During the development of the software part, we used Git as Revision Control System. For the programming language we used Python as it includes a powerful machine learning library PyTorch which we did use. Also during the development of the simulation we benefited from PyGame library which helped a lot in 2D graphics.
- There is no societal or ethical consideration that we need to take.

2.4. Literature Survey

In the work of Ming Tan [4], the author discussed that cooperative agents could give better results than the agents that behave independently. Also in his work, he answered what the cost could be of making the agents cooperate. He categorized them in three approaches which are (1) sharing sensation, (2) sharing episodes and (3) sharing learned policies. He stated that sharing sensation could be suitable if used effectively. He also stated that second and third approaches would speed up learning process at the cost of communication. Finally he analyzed that in multi agent systems the learning would be slow at the beginning.

In the work of Lauer and Riedmiller [5], the focus was on distributed RL in cooperative multi-agent decision process. They stated that agents in SARL do not have information about the behaviour of teammates but it should be exactly the opposite. Besides they proposed a model free distributed Q Learning algorithm on multi-agent decision processes. Moreover they showed that this algorithm can find optimal policy in deterministic environments. Also an interesting point in their work is that cooperating agents do not need any communication links.

In the work of Nagayuki et al. [6], authors stated that in multi-agent environments correctness of an agent’s behaviour depends on the behaviours of the other agents. They claim that in classical RL approaches only one agent changes the environment and the other agents’ actions are ignored. Therefore they studied a two-agent cooperation problem. In their work, one of the agents determines its action based on the prediction of the other agents action. In order to predict, the agent looks to the internal model of the other agent. Finally the authors claim that by their approach they achieved a cooperative behaviour that works as intended.

In the work of DeepMind AI research team [7], researchers have developed a multiagent environment where agents try to capture flag of the opponent team and get it to their own starting area. In their environment, teams are composed of two players. They have compared several situations like 2 agents vs 2 human players or 2 agents vs 1 agent and 1 human etc. After training about 180K games, the agents have become stronger than a strong human player and after 250-300K games, the strength between agents and strong human players have increased significantly.

In the work of Peter Stone [13], author studied about building artificially intelligent agents for complex multiagent control tasks. He chose robotic soccer as multiagent system. He tried to solve the problem with *layered learning* approach. Layered learning approach is a hierarchical *machine learning* paradigm. This approach consists of three interconnected layers. In the first layer, agents learn individual skills. In the second layer, agents learn multiagent behaviour. In the last layer, agents learn team behaviour.

In the work of DeepMind AI research team [17], researchers have presented a distributed architecture for deep reinforcement learning. Their achitecture uses four main components: parallel actors that generate new behaviour, parallel learners that learn from the stored experience, a distributed neural network that represents the value function or behaviour policy and distributed store of experience. They made multiple agents running in parallel that interact with the multiple instances of the same environment in order to generate more data by utilizing multiple CPU cores.

In the work of Google Research Brain Team [18], researchers have announced the release of the Google Research Football Environment, a novel RL environment where agents aim to master the world’s most popular sport—football. Modeled after popular football video games, the Football Environment provides a physics based 3D football simulation where agents control either one or all football players on their team, learn how to pass between them, and manage to overcome their opponent’s defense in order to score goals.

3. SYSTEM DESIGN AND SOFTWARE ARCHITECTURE

3.1. System Design

3.1.1. System Model

In our project, we assume that every team consists of two players. One of the teams agents learn to play soccer using SARL approach, other ones agents learn using MARL approach.

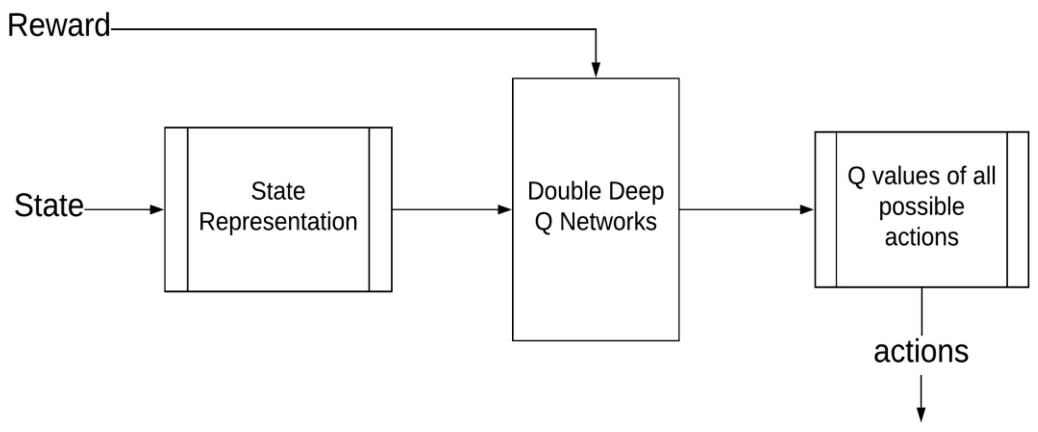


Figure 4: Internal structure of an agent

As seen in Figure 4, SARL agents get reward and state directly from the environment and independently select their actions according to highest Q value. However MARL agents also get state and reward information from the environment like SARL agents. In MARL team, there is a decision maker. It takes N actions of MARL agents which have the highest Q values. Also it takes state information from the environment to determine final actions of MARL agents.

Also for MARL team, there is a Decision Maker module as seen in Figure 5 to provide cooperative and competitive behaviors.

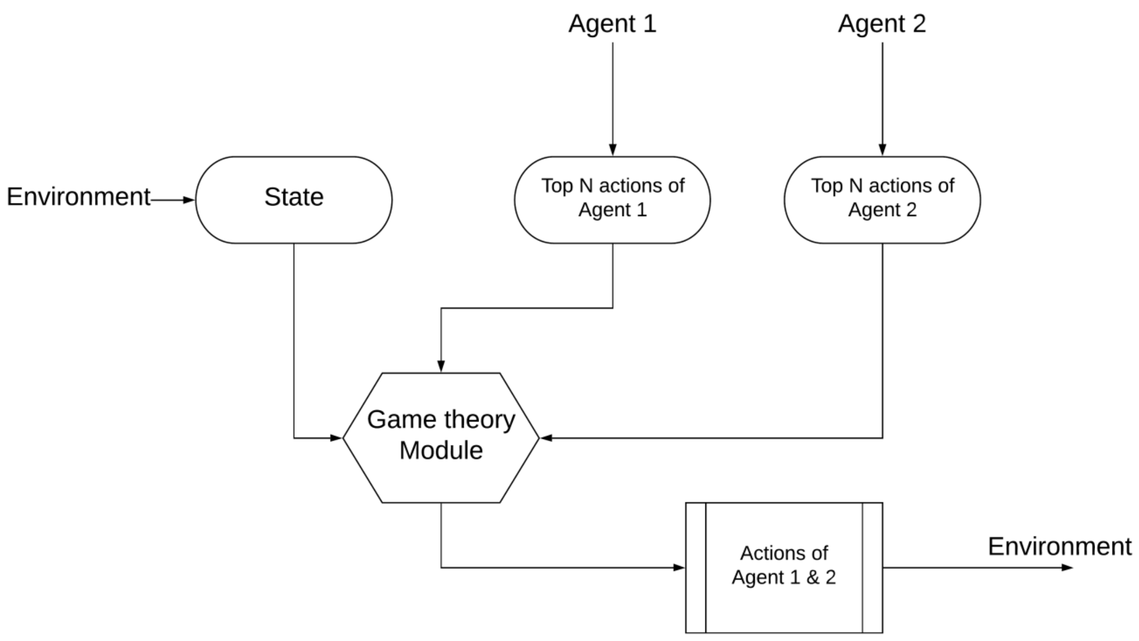


Figure 5: Internal structure of a decision maker

3.1.2. Flowchart of Proposed Algorithms

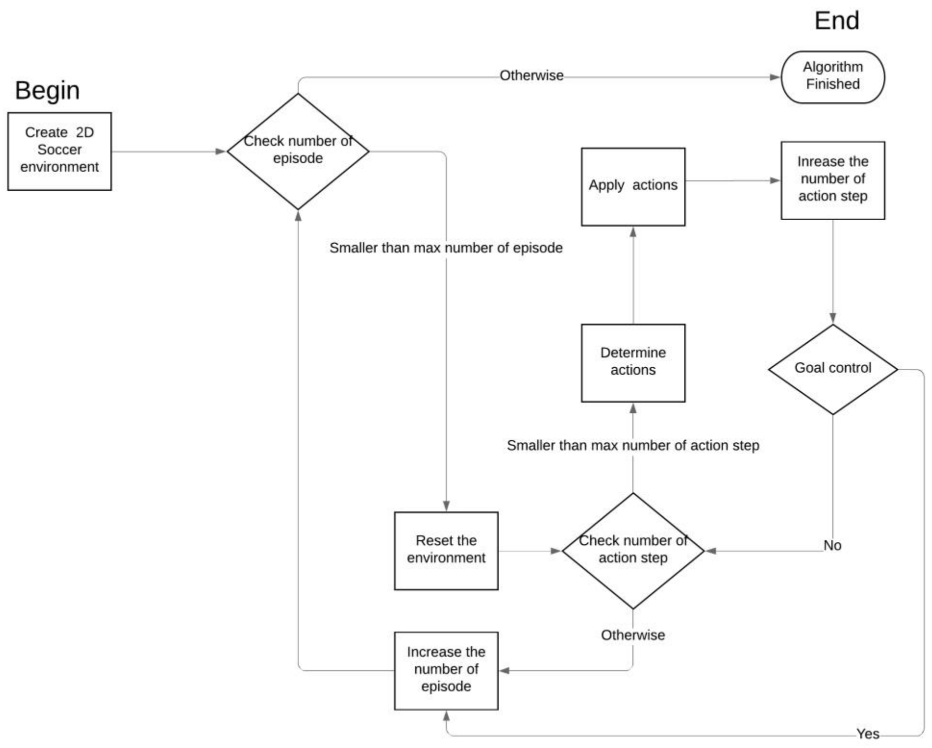

End
Begin
Otherwise
Algorithm Finished
Create 2D Soccer environment
Check number of episode
Apply actions
Inrease the number of action step
Smaller than max number of episode
Determine actions
Goal control
Smaller than max number of action step
Reset the environment
Check number of action step
No
Otherwise
Increase the number of episode
Yes

Figure 6: Flowchart of our algorithm

We explained all steps of our algorithm shown in Figure 6 in depth in section 4.3.2.

3.1.3. Comparison metrics

We compared MARL and SARL approaches in testing phase. This phase consists of two cases. First case is about making 100 matches between MARL and SARL team after training our algorithm through 10K episodes. Thus, we compare learning speeds of both approaches with respect to results. In the results, the team that has higher win percentage is considered as better in terms of learning speed.

Second case is about making 100 matches between MARL and SARL team after training our algorithm through 70K episodes. Thus, if a team increases its winning percentage with respect to the previous cases results, we consider that learning is still happening for that team.

Also we compared two MARL teams with different evaluation functions which we explained in detail in sections 4.3.3.1 and 4.3.3.2 with respect to the cases mentioned above. Thus, we compared effects of different evaluation functions on learning.

3.1.4. Data sets or benchmarks

We will not use any external data sets, because in RL approach data are generated by the interaction of agents with the environment. Also we did not use any benchmarks because there does not exist any benchmarks for 2D soccer environment.

3.2. System Architecture

Our project consists of two parts. One of them is comparing SARL and MARL approaches and other one is comparing MARL and MARL.

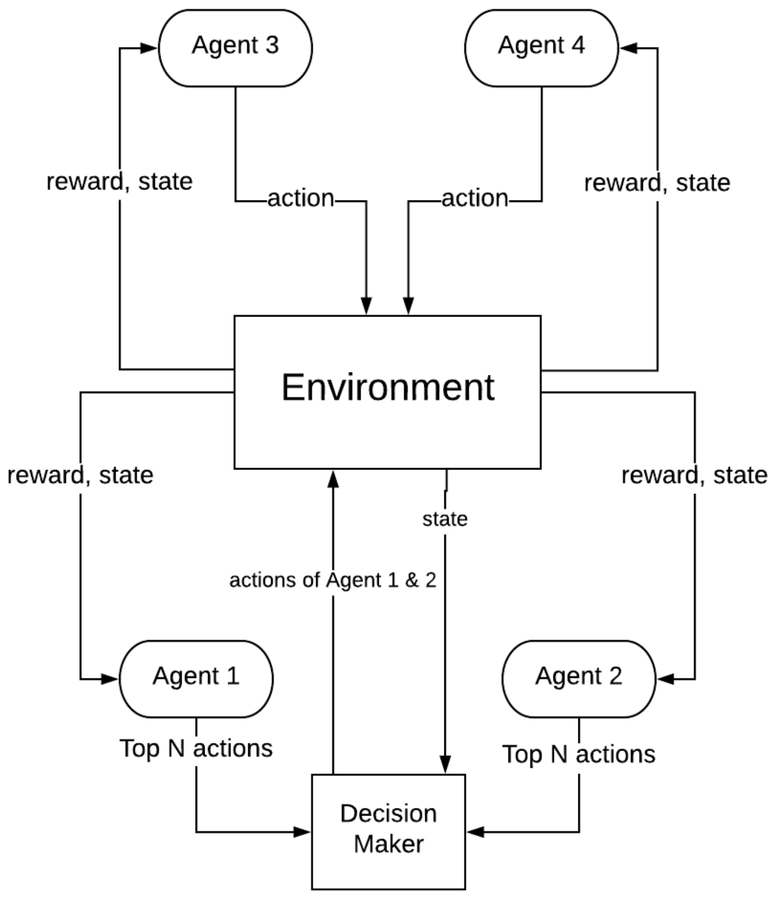


Figure 7: SARL vs MARL system diagram

In Figure 7, agents 1 & 2 are MARL agents and agents 3 & 4 are SARL agents. MARL agents do not directly interact with environment. They communicate with environment via decision maker by sending their top N actions with respect to highest Q values from their DQNs. Thus, decision maker calculates nash equilibrium of actions of MARL agents and determines optimal actions for MARL agents with respect to its evaluation function. SARL agents communicate directly with the environment by sending their highest Q valued actions.

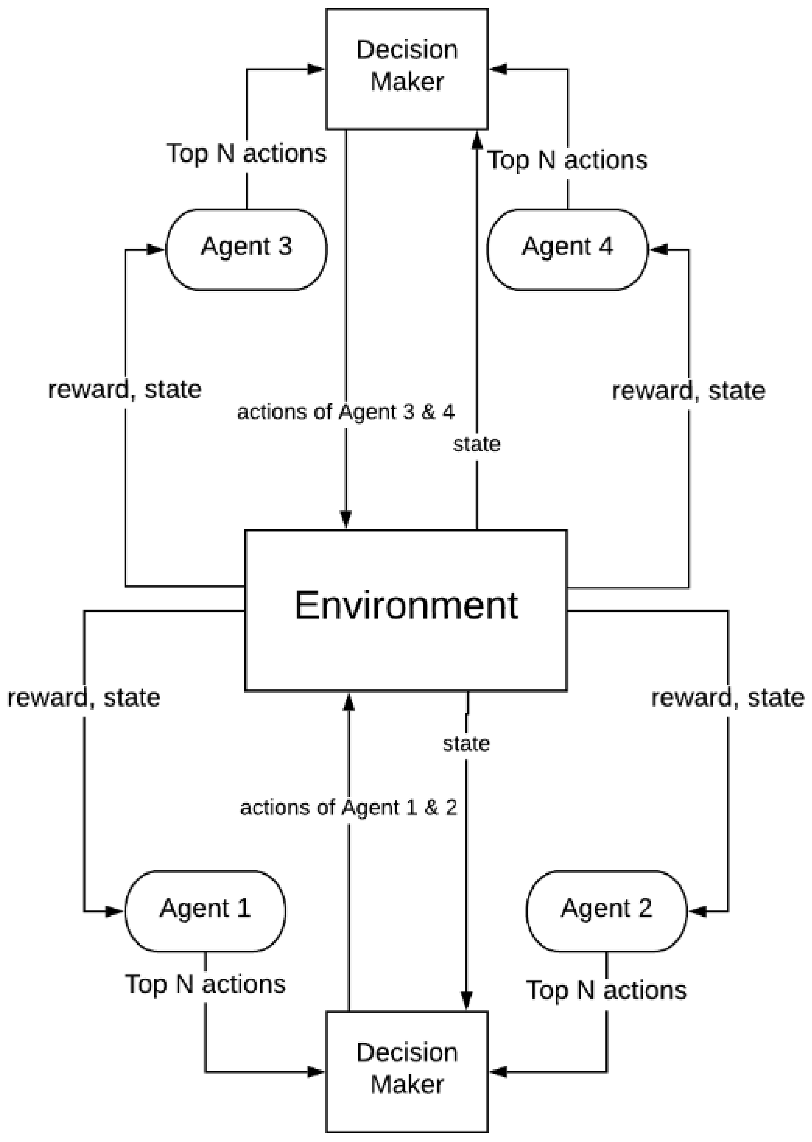


Figure 8: MARL vs MARL system diagram

In Figure 8, there are two MARL teams that have two different decision makers. The difference between these decision makers are the evaluation functions. In this system, both teams do not directly interact with the environment. They communicate with the environment via their decision makers like MARL team we mentioned in previous paragraph. One of the team has a simple evaluation function explained in section 4.3.3.2 and the other one has a more complex evaluation function explained in section 4.3.3.1.

The details of the algorithms used in these systems are discussed in chapter 4.

4. TECHNICAL APPROACH AND IMPLEMENTATION DETAILS

As we stated before, we have developed our own algorithm which learns to play soccer in a 2D environment using RL approach. Also, we have implemented 2D soccer environment which can be integrated with our algorithm and two evaluation functions for 2D soccer. Before explaining these algorithms, we define hardware and software requirements of our project.

4.1. Hardware Requirements

We need only a PC for our project. Optionally, if the PC has CUDA support which version is bigger than 9.0, our algorithm can run on GPU faster than CPU.

4.2. Software Requirements

Python 3.6.x must be installed in a PC which will run our project. Also, PyGame, PyTorch and Numpy libraries must be installed on the PC. We used Python because it contains lots of powerful machine learning and computer graphics libraries. We used PyGame library to implement 2D soccer environment because it is one of the best Python library to write video games. We used PyTorch library to implement DQN because we have experience on using PyTorch library. Therefore, we have implemented DQN easily. We used Numpy library for numerical operations because Python is slow at numerical operations such as matrix multiplication because of it is interpreted language. Therefore, we have made numerical operations fastly with Numpy library.

Lastly, our project can be run on both Windows and Unix based operating systems. However, we prefer to run our project on Unix based operating system because PyGame library runs on Unix based operating system more stable than Windows operating system.

4.3. Implementation Details

4.3.1. 2D Soccer Environment

We have implemented 2D soccer environment with PyGame library. This environment consists of seven main classes which are “***player***”, “***enemy***”, “***coach***”, “***enemy_coach***”, “***brain***”, “***replayBuffer***” and “***game***”. We will explain all of them briefly.

Instance of class ***player*** represents MARL agent. MARL agent has some properties. Speed and angle properties are for defining direction of movement of the agent. Size and mass properties represent its volume and weight. Drag property shows how the agent is effected from the frictional force. Also, there are learning related properties besides these properties. There are two Q networks which are “***qnetwork_local***” and “***qnetwork_target***”. These networks represent brain of the MARL agent. Memory property is for experience replay. State size and action size determine input and output size of these networks, respectively. On the other hand, MARL agent has some methods. ***Move***, ***bounce*** and ***update*** methods are for movement and taking action. ***Act*** and ***learn*** methods are for learning to play 2D soccer.

Instance of class ***enemy*** represents SARL agent. This class is almost the same with the class ***player***. However, the only difference between these classes is class ***player*** has ten actions, eight of them being ***movement action***, ***shoot*** and ***pass*** but the class ***enemy*** does not have the action ***pass***.

Instance of the class ***coach*** represents the decision maker of a MARL team. In soccer, coach is the decision maker of a team. The ***coach*** has some properties, “player1”, “player2”, “enemy1” and “enemy2” properties are for determining which agents are in its own team and which agents are in its opponents team. Also, the ***coach*** has some methods. Evaluation function of the ***coach*** is implemented in ***get_rewards*** method. The ***coach*** determines action with respect to its ***determine_action*** method.
The class ***enemy_coach*** is almost same with the class ***coach***. They both represent the coach of a MARL team. However, they have different evaluation functions. In other words, they have different ***get_rewards*** methods.

Instance of class ***brain*** represents Q network. This class includes the implementation details of the Q network. Our Q network consists of 3 layers. In other words, the Q network has only one hidden layer which consists of 64 nodes. Also, we used relu as activation function in the forwarding step of training.

Instance of class ***replayBuffer*** represents experience replay memory of an agent. Experience replay memory is used to store experience tuples. Experience tuple consists of state, action, reward and next state informations.

The class ***game*** is the main class of 2D soccer environment. When the instance of this class is created; agents, coach/coaches and ball are created. Also, this class have responsibilities of visualising the game, displaying score and checking whether is goal or not. The state informations are returned to coaches and players with "***get_state***" method. The rewards of all players can be retrieved with ***get_rewards*** method. ***Act*** and ***step*** methods are for learning process and movement of players. We explain these two methods in detailed when we explain our algorithm in next section. Our soccer environment is briefly like that we see in Figure 9.

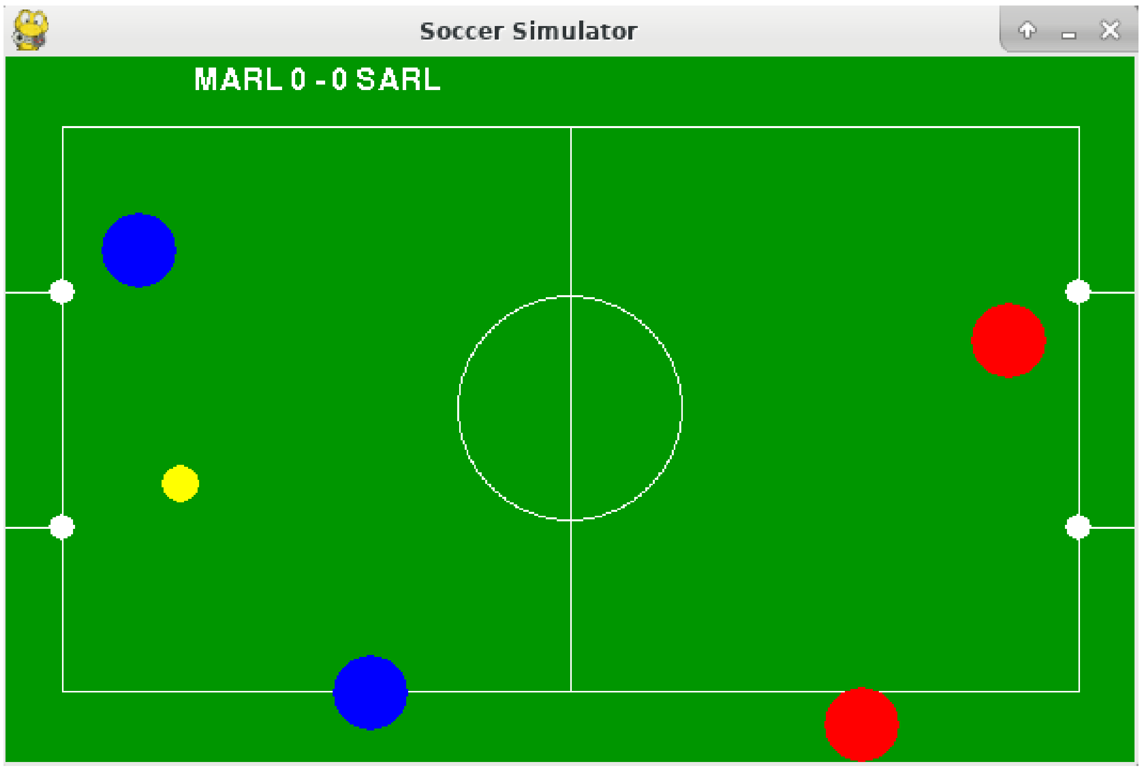
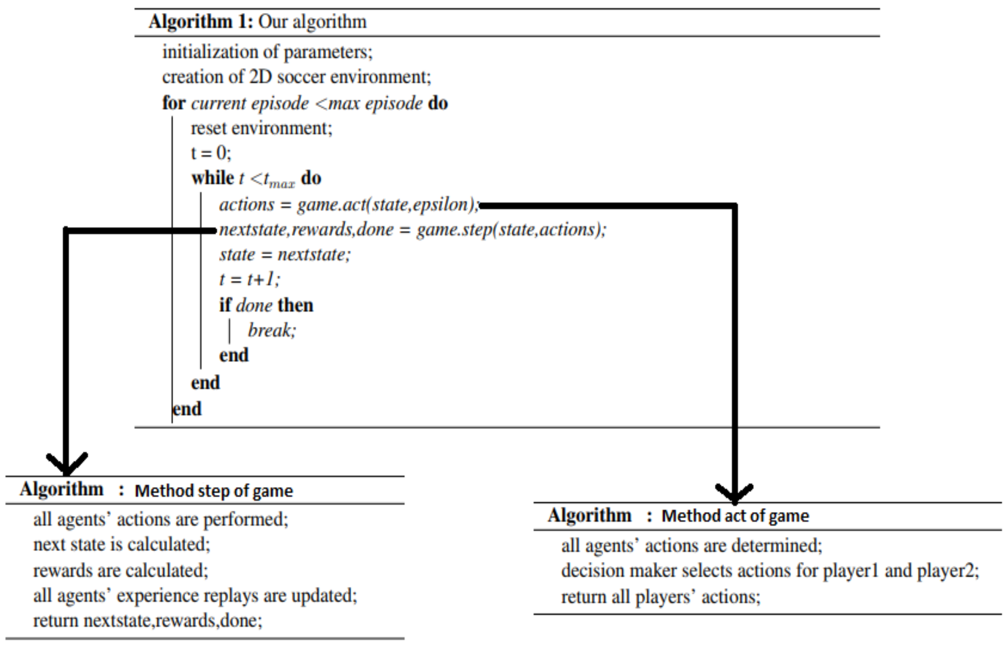


Figure 9: 2D Soccer Environment

4.3.2. Our Algorithm

We have implemented two different versions of our algorithm. In the first one is one team consists of MARL agents and the other team of SARL agents. In the other one both teams are made up of MARL agents and have different coaches. It means different evaluation functions.

4.3.2.1. First Version (MARL vs SARL)



```
Algorithm 1: Our algorithm
  initialization of parameters;
  creation of 2D soccer environment;
  for current episode <max episode do
      reset environment;
      t = 0;
      while $t < t_{max}$ do
          actions = game.act(state,epsilon);
          nextstate,rewards,done = game.step(state,actions);
          state = nextstate;
          t = t+1;
          if done then
              break;
          end
      end
  end
```

```
Algorithm : Method step of game
  all agents' actions are performed;
  next state is calculated;
  rewards are calculated;
  all agents' experience replays are updated;
  return nextstate,rewards,done;
```

```
Algorithm : Method act of game
  all agents' actions are determined;
  decision maker selects actions for player1 and player2;
  return all players' actions;
```

Figure 10: Pseudo code of our algorithm

The details of the pseudo code in Figure 10 are explained below in detail.

Step 1: We create 2D soccer environment. We determine some parameters such as number of episodes, number of maximum action steps etc.

Step 2: We check the current episode whether it is smaller than the number of total episodes. If it is smaller we can go to Step 3 else the program finishes.

Step 3: We reset the environment and reset the action step count.

Step 4: If action step is smaller than maximum action step, then go to Step 5. The reason of this control is, if a goal does not occur in the episode, the episode is finished when the step number for an episode exceeds a predetermined maximum number of steps.

Step 5: We determine actions of SARL and MARL agents with environments method ***act***.

> Step 5.1: In the environments method ***act***, first, each team of SARL and MARL agents call their own methods ***act***.
>
> Step 5.2: These methods ***act*** take state and epsilon values as parameter and their Q networks determine actions with respect to state information. Then, they generate a random number, if random number is smaller than epsilon value, then they return array which consists of random actions. Otherwise, they return output of the Q networks.
>
> Step 5.3: After determining each actions of MARL and SARL agents, the coach of MARL team is informed with current state informations of MARL and SARL agents.
>
> Step 5.4: The coach determines actions with respect to top 4 actions of MARL agents and state information. The coach, firstly, finds the action of enemy1 which returns enemy1 the highest reward with respect to its evaluation function. Then the action of enemy2 is determined in the same way.
>
> Step 5.5: Actions of MARL agents are determined with considering the actions of enemy1 and enemy2 in the previous step by testing each action pairs of MARL agents which gives highest team reward. The team reward becomes sum of player1 reward and player2 reward. Finally, we determined all agents actions.

Step 6: We determine next state, MARL rewards, SARL rewards and whether a goal occur or not with environments method ***step***. This method takes state information and actions of all agents which were determined in Step 5.

> Step 6.1: Method ***step***, firstly, calculates all agents rewards in current state with ***get_rewards*** method. Method ***get_rewards***, returns seperate rewards to all agents. Rewards of the SARL agents are determined by the environments

evaluation function but rewards of the MARL agents are determined by evaluation function of MARL teams coach.

Step 6.2: Each agents take actions which were determined in Step 5 and next state is calculated.

Step 6.3: Rewards of the next state is calculated in the same way described in Step 6.1.

Step 6.4: Real rewards of the all agents are calculated by subtracting the current state rewards from the next state rewards.

Step 6.5: All agents call their own methods ***step*** to learn the affect of their actions. One of the parameters of the method ***step*** is reward of the agent. MARL agents send sum of team reward and individual reward which they get from their coach as reward parameter to method ***step***. SARL agents send individual reward which they get from the environment as reward parameter to method ***step***.

Step 6.6: State, action, next state and done informations are added to their own experience replay memories.

Step 6.7: Every 4 executions of method ***step***, sample experiences are taken from their own memories and their local Q networks are trained with these sample experiences with respect to their target Q network.

Step 6.8: After some training, every agents' target Q networks' weight and bias values are synchronized with its local Q network.

Step 6.9: If next state is goal state. The agent that scores is rewarded with 10000 and it executes its method ***step*** additionaly. The agent that assists is rewarded with 5000 and it executes its method ***step*** additionaly. Therefore, actions which results in goal are taken more frequently.

Step 7: Increase the action step count.

Step 8: If goal does not occur and action step count is smaller than max action step count, then go to Step 5, otherwise Step 9.

Step 9: Write Q networks' parameters of all agents in every ten episode.

<u>Step 10:</u> Increase the episode number and go to Step 2.

4.3.2.2. <u>Second Version (MARL vs MARL)</u>

This versions steps are also same with the flowchart in Figure 6 in section 3.1.2. Only differences with first version are implementations of Step 5 and Step 6. Only the differentiated steps will be covered.

<u>Step 5:</u> We determine actions of MARL agents of two teams with environments method ***act***.

> <u>Step 5.1:</u> In the environments method ***act***, firstly, each agent call their methods ***act***.
>
> <u>Step 5.2:</u> These methods ***act*** take state information and epsilon values as parameter and their local Q networks determine actions with respect to state. Then, they generate a random number, if random number is smaller than epsilon value, then they return an array which consists of random actions. Otherwise, they return output of their local Q networks.
>
> <u>Step 5.3:</u> After determining each action of all agents, the coaches of MARL teams are informed with state informations of all agents.
>
> <u>Step 5.4:</u> Coaches determine their actions by their teams players top 4 actions and state informations. Both coaches first determine the opponent teams actions that maximizes the team reward of opponent team. Then using these actions, they determine action of their own players that maximizes their own team reward. The actions that the coaches determine for their own team players will be selected as the actions taken in corresponding state.

<u>Step 6:</u>

> ...
>
> <u>Step 6.5:</u> All agents call their methods ***step*** to learn the affect of the actions. These methods ***step*** take reward as an parameter. Both teams send sum of team rewards and individual rewards to methods ***step***.

4.3.3. Evaluation Functions

We have implemented two different evaluation functions for 2D soccer which are coaches evaluation functions and enemies coaches evaluation functions.

4.3.3.1. Coaches Evaluation Function

The rewards of its MARL agents are determined with some conditions. These conditions are players horizontal and vertical positions and balls horizontal and vertical positions. Also, rewards consist of some component. These components are players' distance to opponents goal line, own goal line, upper line of the field, bottom line of the field and ball. Reward increases proportionally with the distance to own goal line while decreases proportionally with the distance to the opponents goal line and ball. The reward is composed of certain components depending on the conditions of the players. This heuristic is in the shape of a decision tree. Rewards are calculated as shown in Figure 11.

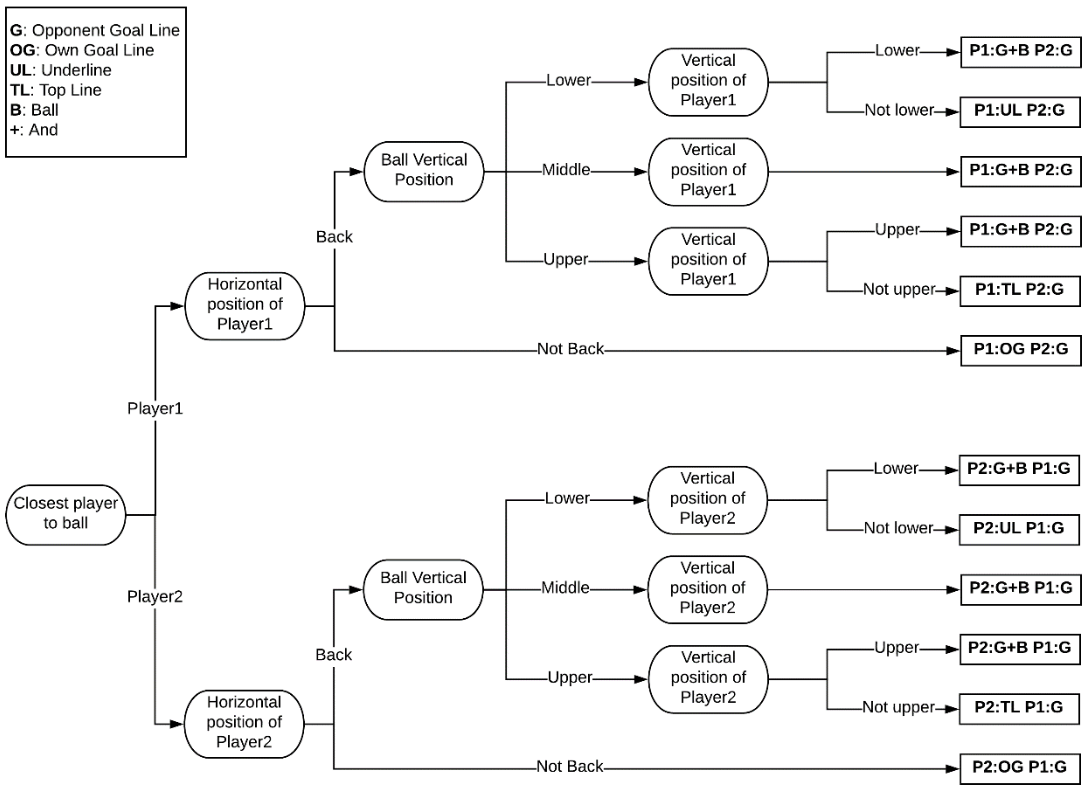


Figure 11: Decision tree of the coaches reward heuristic function

4.3.3.2. Opponents Coaches Evaluation Function

This evaluation function is simpler than the previous one. It gives a reward with inverse proportion to the distance of the players to the ball and opponents' goal line. This function results in oscillation when players are between the ball and the opponents' goal line because when the player moves closer to the ball it can not take action to move ball to the opponents' goal line. The reason for this is that the player is in front of the ball.

5. EXPERIMENTAL STUDY

We ran our tests on a workstation with Unix based OS and 64-core Intel i9 CPU. Our test cases are explained below:

We compared teams in two cases by making them play 100 matches having trained 10K and 70K episodes. We applied our tests both for MARL vs SARL and MARL vs MARL versions.

Our tests are based on matches between two teams. Each match consists of five episodes. Episodes end when a goal occurs or 1000 actions are performed. At the end of 5 episodes, the team with a higher number of goals wins that match. If both teams' goal counts are equal, match ends in a draw.

5.1. Test Results

5.1.1. Test Results of MARL vs SARL

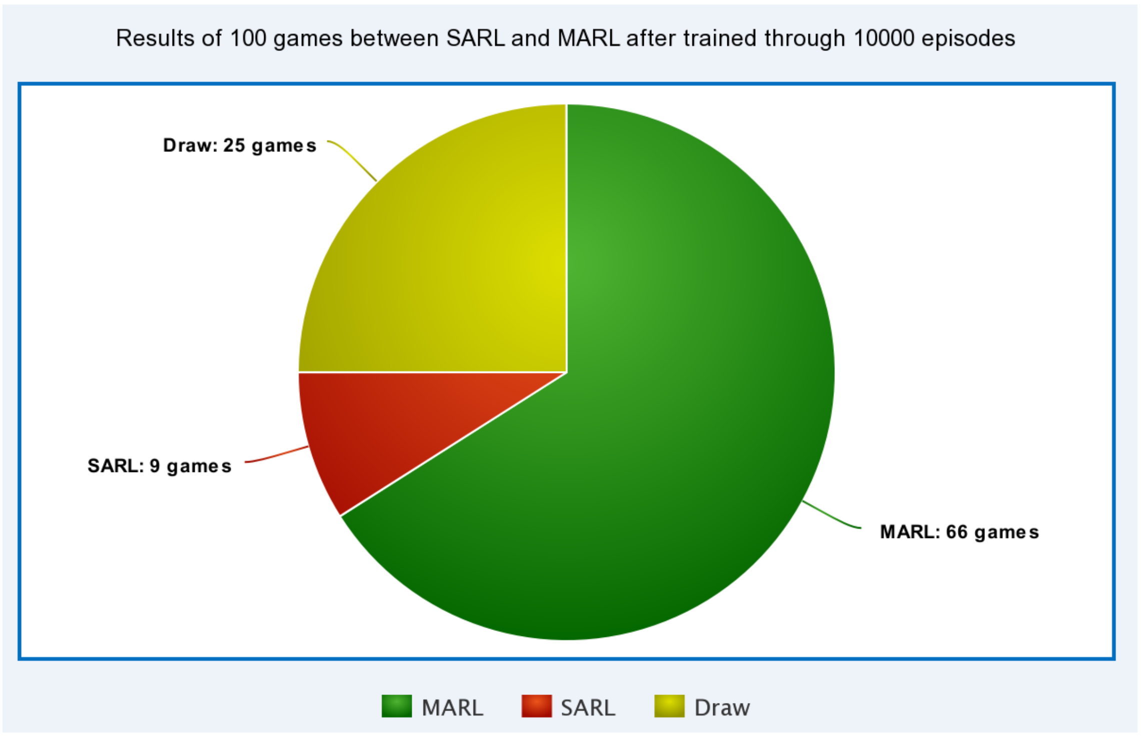


Figure 12: SARL vs MARL results after 10K episodes

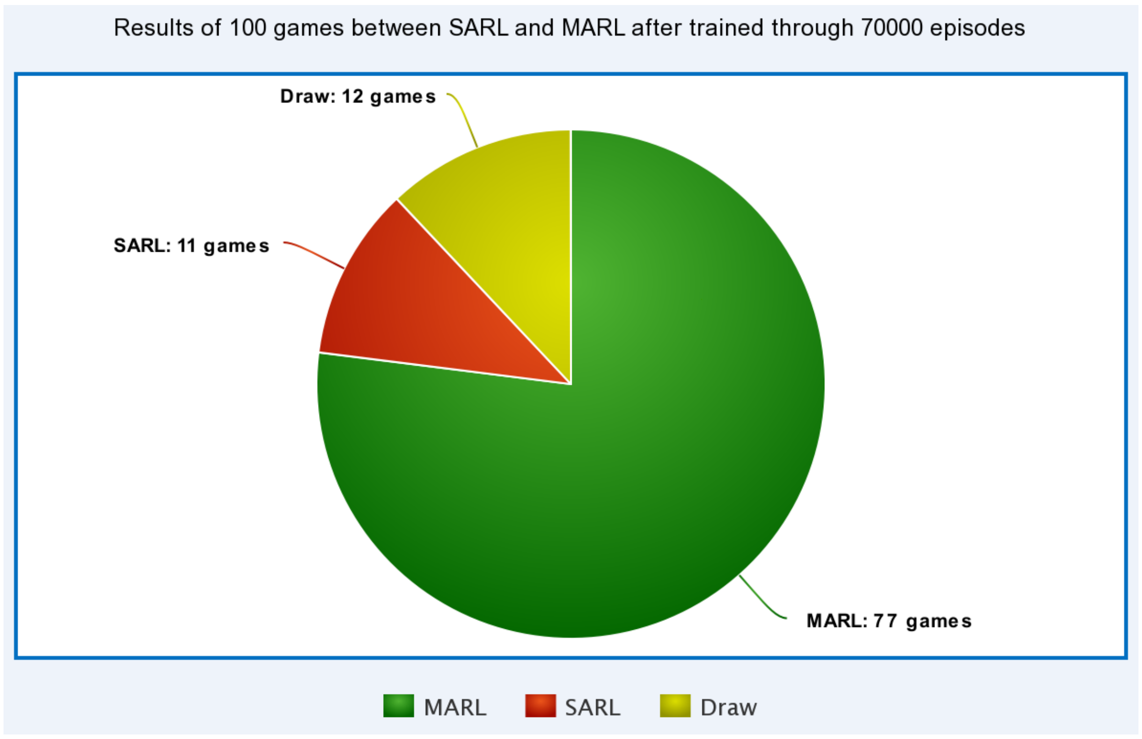


Figure 13: SARL vs MARL results after 70K episodes

5.1.2. Test Results of MARL vs MARL

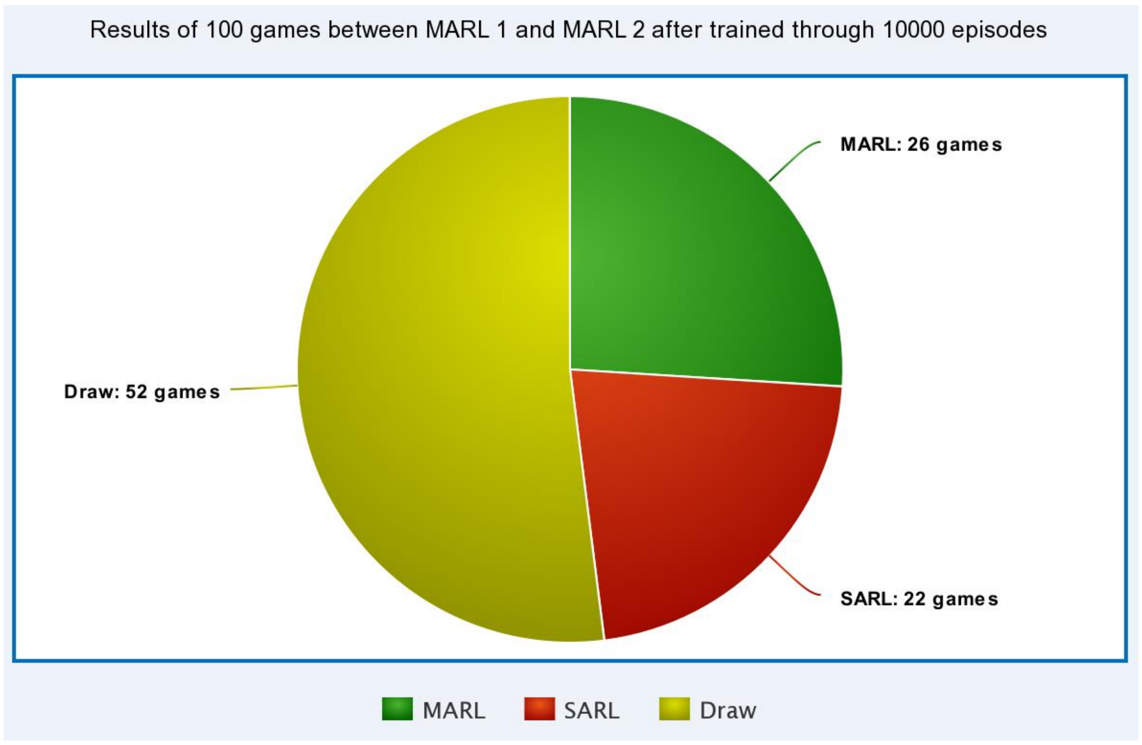


Figure 14: MARL vs MARL results after 10K episodes

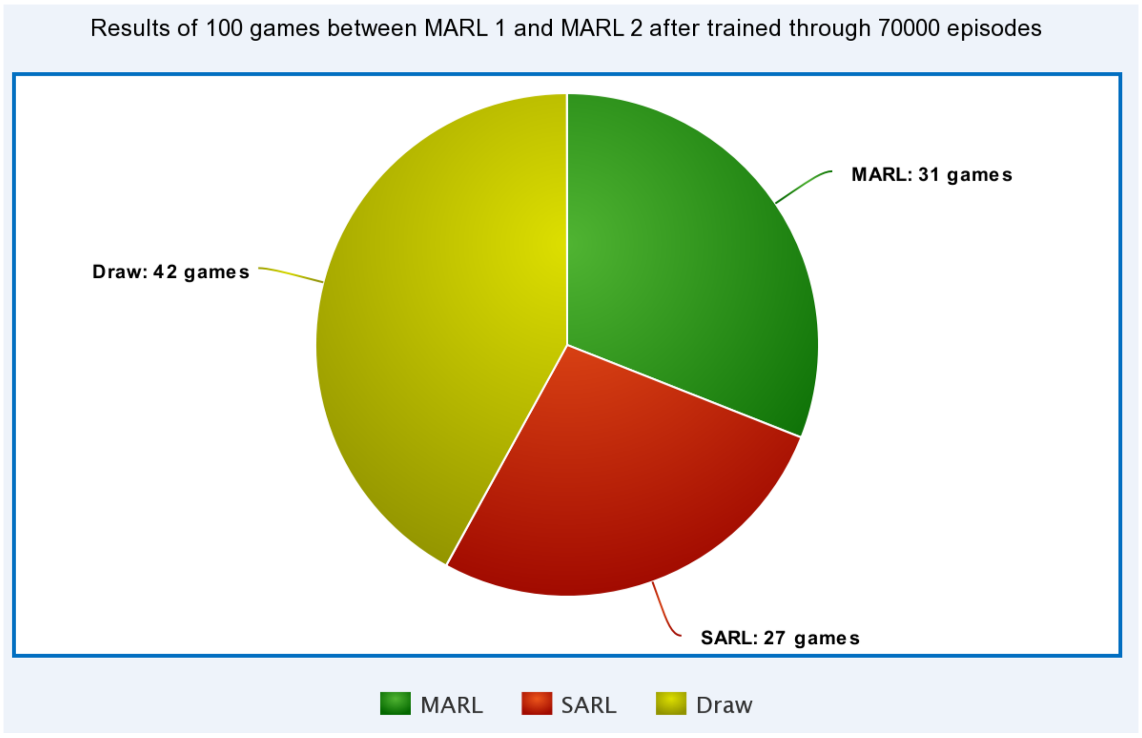


Figure 15: MARL vs MARL results after 70K episodes

5.2. Discussion

The reason for selecting these test cases is to see learning speed of the teams. When conducting the experiment we first compared the results at the end of 10K episodes. By looking at the results, we concluded that the team with more wins has learned faster. The reason of comparing results between 10K and 70K episodes is to control whether the learning continues or not.

In MARL vs SARL case, it is seen in the Figure 12 that after 10K episodes MARL team has 66 wins while SARL team has 9 wins and 25 of the matches have ended in a draw. As we can see, 88% of the total wins belong MARL team while 12% of the total wins belong to SARL team. This shows that MARL teams has learned faster at the end of 10K episodes. When we compare the result of 10K and 70K trainings, draw percentage was decreased by 48%. 84.61% of the decrease went up as MARL teams wins and remaining percentage went up as SARL teams wins. Therefore we concluded that both teams are continuing learning and MARL is still learning faster because portion of increase in MARL is higher than in SARL.

Before interpreting the MARL vs MARL results, MARL 1 team uses the Coaches Evaluation Function mentioned in Chapter 4 and MARL 2 team uses the Opponents Coaches Evaluation Function mentioned in Chapter 4. Both teams agents are identical.
In the Figure 14, we see that after training 10K episodes, 54.1% of the total wins belong to MARL 1 while 45.8% of the total wins belong to MARL 2 and 52% of the games resulted in a draw. The draw percentage is higher than total win percentage. This shows that both teams could not show superiority over each other. Between 10K and 70K trainings, the difference of improvements of both teams is negligible. Both teams continued improving as the win percentages have increased but draw percentage is still higher. That tells us that both teams have still a similar performance. As MARL 1 teams wins are more than MARL 2 teams wins, we can say that MARL 1 coaches evaluation function has a small advantage over MARL 2 coaches evaluation function.

6. CONCLUSION AND FUTURE WORK

In this work we showed that in multi agent systems, instead of using SARL approach the MARL approach where the interaction between agents are considered is more appropriate. We chose a soccer environment as a multi agent system because in the soccer environment agents can perform both cooperative and competitive behaviours. In this project we created our own algorithm as MARL algorithm and made comparisons between SARL algorithm and our algorithm. As a conclusion we observed that our MARL algorithm performed better in soccer environment.

The MARL algorithm that we developed has some advantages and disadvantages. Main advantage is the learning speed. It is because it returns rewards via an evaluation function. Normally in soccer, the only reward would be given when the goal occurs which results in slower learning. Also our algorithm takes opponent teams interactions into consideration in addition to the interactions of own team players which leads to selecting actions closer to the optimal ones.
On the other hand, the evaluation function limits the learning because agents' playing strengths are going to depend on the heuristic and heuristic function might not cover all the possibilities in soccer. In other words, an agent can learn as much as heuristic function allows. Also our algorithms at the moment supports only two-player teams.

We would like to mention about our future works. First of all, we will complete the parallelization of training neural networks. By doing that, we hope to decrease the training time proportionally to the number of CPU cores. We are now doing literature search. Secondly, we are planning to extend our algorithm by developing a mechanism to learn the evaluation functions. Thirdly, we will train teams that have agents with different properties. Finally, we will work on train teams with various number of agents.

References

[1] Sutton, R.S. & Barto, A.G., Reinforcement Learning: An Introduction, 2017.

[2] Watkins, C.J.C.H. & Dayan, P. Mach Learn (1992) 8: 279

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, "Human-level control through deep reinforcement learning", Nature, vol. 518, no. 7540, pp. 529-533, 2015.

[4] Tan, Ming. (1993). Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents.. 330-337.

[5] Lauer, M., and Riedmiller, M. (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In Proceedings of the Seventeenth International Conference in Machine Learning.

[6] Nagayuki, Yasuo & Ishii, Shin & Doya, Kenji. (2000). Multi-Agent Reinforcement Learning: An Approach Based on the Other Agent's Internal Model.

[7] "Capture the Flag: the emergence of complex cooperative agents | DeepMind", DeepMind, 2018. [Online]. Available: https://deepmind.com/blog/capture-the-flag/. [Accessed: 20- Oct- 2018].

[8] Chincoli, Michele & Liotta, Antonio. (2018). Self-Learning Power Control in Wireless Sensor Networks. Sensors. 18. 375. 10.3390/s18020375.

[9] "Applied Deep Learning - Part 1: Artificial Neural Networks", Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/applied-deeplearning-part-1-artificial-neural-networks-d7834f67a4f6. [Accessed: 25- Oct-2018].

[10] "Simple Reinforcement Learning with Tensorflow Part 4: Deep Q-Networks and Beyond", Medium, 2018. [Online]. Available: https://medium.com/@awjuliani/simple-reinforcement-learning-with-tensorflowpart-4-deep-q-networks-and-beyond-8438a3e2b8df. [Accessed: 25- Oct- 2018].

[11] "Modern Game Theory and Multi-Agent Reinforcement Learning Systems", Towards Data Science, 2018. [Online]. Available: https://towardsdatascience.com/modern-game-theory-and-multi-agentreinforcement-learning-systems-e8c936d6de42. [Accessed: 25- Oct- 2018].

[12] "Multi-agents Self-Driving Mario Kart with Tensorflow and CNNs", Medium, 2018. [Online]. Available: https://medium.com/@aymen.mouelhi/multi-agents-self-driving-mario-kart-with-tensorflow-and-cnns-c0f2812b4c50. [Accessed: 25- Oct2018].

[13]P. Stone, Layered Learning in Multiagent System, 1st ed. The MIT Press, 2000.

[14]Cs.toronto.edu, 2019. [Online]. Available: https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf. [Accessed: 01- Jan- 2019].

[15]Csc2.ncsu.edu, 2019. [Online]. Available: https://www.csc2.ncsu.edu/faculty/mpsingh/books/MAS/MAS-Springer/1.pdf. [Accessed: 02- Jan- 2019].

[16]"Guest Post (Part I): Demystifying Deep Reinforcement Learning - Intel AI", Intel AI, 2019. [Online]. Available: https://ai.intel.com/demystifying-deep-reinforcementlearning/. [Accessed: 02- Jan- 2019].

[17] Nair, A., Srinivasan, P., Suleyman, M. and Mnih, V. (2015). Massively Parallel Methods for Deep Reinforcement Learning.

[18] Kurach, K., Raichuk, A., Stanczyk, P. and Bachem, O. (2019). Google Research Football: A Novel Reinforcement Learning Environment.