UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Felipe Bezerra Martins

Exploring Multi-Agent Deep Reinforcement Learning In IEEE Very Small Size Soccer

Recife

2023

Felipe Bezerra Martins

Exploring Multi-Agent Deep Reinforcement Learning In IEEE Very Small Size Soccer

A M.Sc. Thesis presented to the Centro de Informática of Universidade Federal de Pernambuco in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Concentration Area: Computer Intelligence

Advisor: Hansenclever de França Bassani

Recife

2023

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

M386e Martins, Felipe Bezerra
Exploring multi-agent deep reinforcement learning in IEEE very small size soccer / Felipe Bezerra Martins – 2023.
85 f.: il., fig., tab.

Orientador: Hansenclever de França Bassani.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2023.
Inclui referências.

1. Inteligência computacional. 2. Aprendizado por reforço. 3. Robótica. 4. Sistemas multiagentes. I. Bassani, Hansenclever de França (orientador). II. Título

006.31 CDD (23. ed.) UFPE - CCEN 2024 – 009

Felipe Bezerra Martins

“Exploring Multi-Agent Deep Reinforcement Learning In IEEE Very Small Size Soccer”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 27 de setembro de 2023.

ACKNOWLEDGEMENTS

I want to thank Edna for giving me the final push to motivate me to enroll in the pursuit of my master's degree. Hans, you are an outstanding advisor from whom I learned so much; thank you.

I am very grateful to RobôCIn. There, I met some fantastic people who inspired me to achieve great things; they taught me a lot and made me a better person. I want to apologize to everyone I interrupted to babble about my problems, especially Gonça; circumstances made us work together a lot. I am grateful to have been able to work with a friend.

There was no moment in which I doubted if my family would support me in whatever I set out to do; I am very grateful for that.

ABSTRACT

Robot soccer is regarded as a prime example of a dynamic and cooperative multi-agent environment, as it can demonstrate a variety of complexities. Reinforcement learning is a promising technique for optimizing decision-making in these complex systems, as it has recently achieved great success due to advances in deep neural networks, as shown in problems such as autonomous driving, games, and robotics. In multi-agent systems reinforcement learning research is tackling challenges such as cooperation, partial observability, decentralized execution, communication, and complex dynamics. On difficult tasks, modeling the complete problem in the learning environment can be too difficult for the algorithms to solve. We can simplify the environment to enable learning, however, policies learned in simplified environments are usually not optimal in the full environment. This study explores whether deep multi-agent reinforcement learning outperforms single-agent counterparts in an IEEE Very Small Size Soccer setting, a task that presents a challenging problem of cooperation and competition with two teams facing each other, each having three robots. We investigate diverse learning paradigms efficacies in achieving the core objective of goal scoring, assessing cooperation by comparing the results of multi-agent and single-agent paradigms. Results indicate that simplifications made to the learning environment to facilitate learning may diminish cooperation's importance and also introduce biases, driving the learning process towards conflicting policies misaligned with the original challenge.

Keywords: reinforcement learning; robotics; multi-agent systems.

RESUMO

O futebol de robôs é considerado um excelente exemplo de ambiente multiagente dinâmico e cooperativo, podendo demonstrar uma variedade de complexidades. A aprendizagem por reforço é uma técnica promissora para otimizar a tomada de decisões nestes sistemas complexos, obtendo recentemente grande sucesso devido aos avanços nas redes neurais profundas, como mostrado em problemas de direção autônoma, jogos e robótica. Em sistemas multiagentes, a pesquisa de aprendizagem por reforço está enfrentando desafios de cooperação, observabilidade parcial, execução descentralizada, comunicação e dinâmicas complexas. Em tarefas difíceis, modelar o problema completo no ambiente de aprendizagem pode ser muito desafiador para os algoritmos resolverem, podemos simplificar o ambiente para permitir a aprendizagem, contudo, as políticas aprendidas em ambientes simplificados geralmente não são ideais no ambiente completo. Este estudo explora se a aprendizagem profunda por reforço multiagente supera as contrapartes de agente único em um ambiente de futebol de robôs da categoria IEEE Very Small Size Soccer, uma tarefa que apresenta um problema desafiador de cooperação e competição com duas equipes frente a frente, cada uma com três robôs. Investigamos a eficácia de diversos paradigmas de aprendizagem em alcançar o objetivo central de realizar gols, avaliando a cooperação, comparando os resultados de paradigmas multiagentes e de agente único. Os resultados indicam que as simplificações introduzidas no ambiente para facilitar a aprendizagem podem diminuir a importância da cooperação e introduzir vieses, conduzindo o processo ao aprendizado de políticas conflitantes e desalinhadas com o desafio original.

Palavras-chave: aprendizado por reforço; robótica; sistemas multiagentes.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

CPU	Central Processing Unit
DDPG	Deep Deterministic Policy Gradient
DQN	Deep Q-Network
GPU	Graphics Processing Unit
IL	Independent Learners
IQM	Interquartile Mean
JAL	Joint-Action Learners
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MARL	Multi-Agent Reinforcement Learning
MDP	Markov Decision Process
OU	Ornstein–Uhlenbeck Process
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
RSA	Replicated Single-Agent
SA	Single-Agent
VSSS	IEEE Very Small Size Soccer

CONTENTS

1 INTRODUCTION

The process of trial-and-error exploration is a common form of learning in nature. To illustrate this, let us consider a driver who wants to complete a lap on a racing circuit in the shortest possible time. While directions and instructions can be given to reduce lap time, the driver can also learn through training. When performing laps on the circuit, the driver can refine their actions by experimenting and honing in on behaviors that maximize their speed.

A driver attempting to reduce their lap time is an example of goal-directed learning. The driver learns by interacting with its environment, exploring possible actions and their outcomes, such as controlling the accelerator, brake, clutch, steering angle, and gear selection in different ways. Racing is a complex task and requires knowledge of the vehicle dynamics, the circuit, and the processing of the driver senses. Using a classic artificial intelligence approach of defining a set of heuristics to tackle this decision-making task is challenging. For such problems, Reinforcement Learning (RL) has proved to be an exciting path. RL is a computational approach to goal-directed learning through the interaction of an agent with its environment (SUTTON; BARTO, 2018).

RL has succeeded in applications in several areas, such as in robotics, learning quadruped locomotion (SMITH; KOSTRIKOV; LEVINE, 2022), autonomous driving (KIRAN et al., 2021), in natural language processing, fine-tuning large-scale conversational models (TOUVRON et al., 2023; OPENAI, 2023), and in electronic (MNIH et al., 2015) and board (SCHRITTWIESER et al., 2020) games.

RL has interesting challenges when applied to multi-agent systems, known as Multi-Agent Reinforcement Learning (MARL). MARL is when multiple agents interact with the environment in a learning process (WEISS, 1999). To illustrate a multi-agent system, let us consider a driver in a race. This time, the driver is not alone, other drivers are on the track, making the problem more complex, as the driver must now consider the behavior of the other drivers and how their actions affect each other, such as overtaking, blocking, and drafting, in addition to continuing the need of taking actions to reduce his lap times.

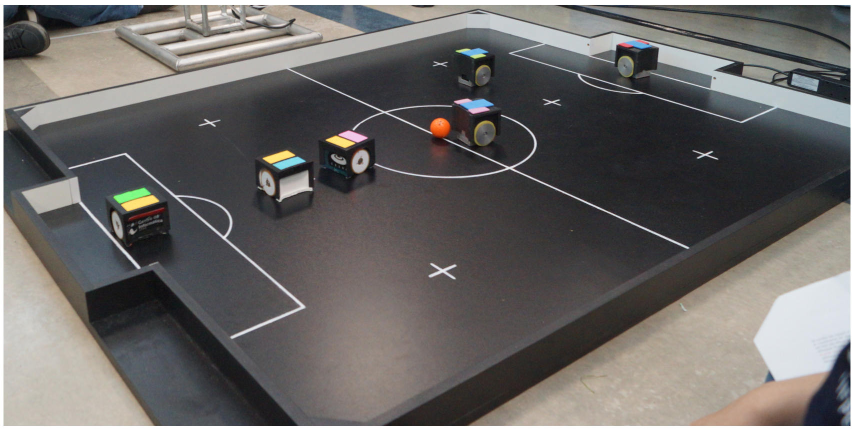
The success of RL also extends to MARL in applications such as traffic signal management (WU et al., 2020), hide-and-seek game (BAKER et al., 2019), and soccer (LIU et al., 2019; BRANDÃO et al., 2022).

We can attribute part of the recent advances that we observed in RL and MARL to the extraordinary advances that have been obtained in the use of deep learning methods (LECUN; BENGIO; HINTON, 2015) and the field of machine learning as a whole, which is a very active field today. The use of MARL in dynamic and complex problems is an open problem, which requires studies of MARL in these complex environments (NGUYEN; NGUYEN; NAHAVANDI, 2020), especially examining the most recent deep MARL approaches.

A soccer match in which two teams compete to see who can score the most goals is a globally renowned team sport (CLEMENTE et al., 2014). Soccer is a classic example of a multi-agent system, typically seen as challenging and intricate. Consequently, soccer is an intriguing subject for the exploration of MARL. It involves robots coordinating with each other in a dynamic setting while also controlling their movements.

The IEEE Very Small Size Soccer (VSSS) competition (Figure 1) presents an interesting platform for researchers to experiment with RL techniques. Previous research has demonstrated the effectiveness of using RL in VSSS to teach robots how to score goals (DELGADO, 2019; BASSANI et al., 2020), learning a defensive behavior (MEDEIROS; MÁXIMO; YONEYAMA, 2020), transfer policies from simulations to real robots (BASSANI et al., 2020), coordinate a team through the selection of behaviors (PENA et al., 2020), and even coordinate a team through direct control over the robots (BRANDÃO et al., 2022).

Figure 1 – IEEE Very Small Size Soccer match during the 2022 Latin American Robotics Competition.



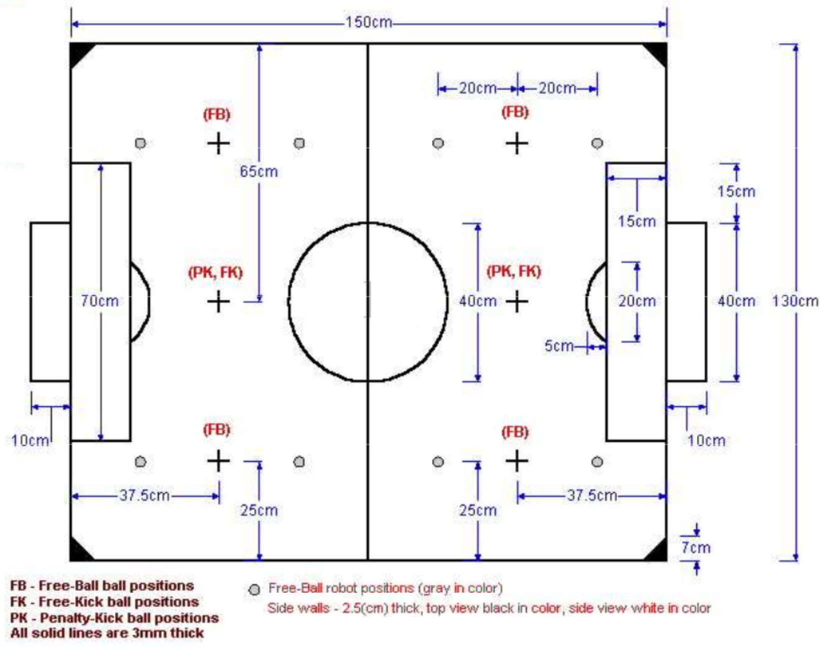
Source: Author (2022).

This work focuses on the VSSS competition to explore MARL. This competition offers the challenges of robot soccer while being more cost-effective and smaller in scale than other competitions. Additionally, the considerable research conducted on the VSSS competition highlights its relevance. Section 1.1 introduces the VSSS competition and its rules. In Section 1.2, we present the objectives of this work.

1.1 IEEE VERY SMALL SIZE SOCCER

The VSSS robot soccer category encourages the advancement of robotics and artificial intelligence through a soccer game challenge. Using pre-programmed autonomous robots, two teams compete to see who can score by moving a golf ball to the inside of the opposing team goal (Figure 1). The robots play in a field surrounded by walls, measuring 150 *centimeters* from one goal to the other and 130 *centimeters* between the sides (Figure 2). The robots are limited to a maximum dimension of a cube size with sides of 7.5 *centimeters*.

Figure 2 – IEEE Very Small Size Soccer field dimensions.

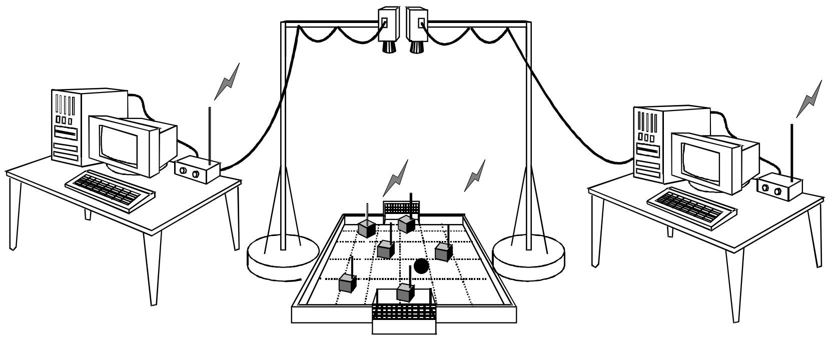

150cm
20cm
20cm
(FB)
(FB)
65cm
15cm
15cm
(PK, FK)
(PK, FK)
70cm
40cm
20cm
40cm
130cm
5cm
10cm
10cm
(FB)
(FB)
37.5cm
37.5cm
25cm
25cm
7cm
FB - Free-Ball ball positions
FK - Free-Kick ball positions
PK - Penalty-Kick ball positions
All solid lines are 3mm thick
Free-Ball robot positions (gray in color)
Side walls - 2.5(cm) thick, top view black in color, side view white in color

Source: (PINTO, 2023).

A VSSS match has two halves, each lasting five minutes with an eight-minute break between the halves. Robots are not allowed to hold or cover the ball and due to size restrictions, teams cannot fit a kicker device. Therefore, the dynamics of the match involve the robots dragging the ball. To score goals, teams program the robots to take the ball into the opponent goal. A camera is mounted above the field to capture the entire area, and teams use computer vision techniques to identify the position of the robots and the ball (Figure 3). The robots have a colored pattern on top to identify them, their position, angle, and team. With the information extracted from the camera, an external computer processes the data and sends a radio control signal to the robot.

During the match, if the ball does not move for 10 seconds, the referee pauses the game

Figure 3 – Overview of a IEEE Very Small Size Soccer match system with camera, external computer and radio.



Source: (KIM et al., 1997).

and restarts it after repositioning the robots and the ball. The rules also define three types of fouls: attacking fouls, which occur when a team in an attack move enters with more than one robot in the opposing team goal area, and the team that commits is penalized with a goal kick; defensive fouls, when a team in defensive play has more than one robot within its goal area when the ball is also inside, and the team that commits is penalized with a penalty kick; and personal fouls, which are fouls that affect the progress of the game. The rules mention cases of collisions with the potential to damage the robot of the opposing team, collisions that overturn the opponent robot, and a team taking more than 10 seconds to remove the ball from its goal area, as long as the opposing team does not impede it, which is also a personal foul and also the situation in which a robot holds the ball. The punishment for personal fouls depends on the situation.

1.2 OBJECTIVES

Two researchers apply MARL in the context of VSSS; PENA et al. (2020) learn a policy to manage a team by selecting a predefined behavior for each robot, not addressing the challenge of end-to-end control. In more recent work, BRANDÃO et al. (2022) successfully applies MARL to achieve team coordination and end-to-end control, but not comparing the result to single-agent approaches. We raise the question of how the MARL performance compares to single-agent approaches such as the one used by BASSANI et al. (2020) in a VSSS competition, which replicates a policy learned in a single-agent environment to control multiple robots.

Although MARL is a promising solution for the VSSS problem, its learning can present a difficult challenge. This thesis studies the application of MARL in a VSSS robot soccer environment with the following objectives:

a) to learn policies capable of scoring goals in the environment using single-agent and multi-agent paradigms;

b) to compare the performance of different multi-agent and single-agent RL paradigms in the environment;

c) to understand the role cooperation plays and its importance in the VSSS environment.

This thesis aims to deepen our understanding of the effectiveness and limitations of MARL methods in the VSSS environment. Our findings demonstrate that applying MARL is challenging and not always the best solution, and we hope our findings about cooperation importance in the VSSS environment motivate further research. Additionally, contributions include developing the rSoccer framework (MARTINS et al., 2021), which facilitates the creation of VSSS learning environments, developing a VSSS game simulation that accelerates RL training through Graphics Processing Unit (GPU) physics calculations, and publishing research in robotics.

We see as contributions of this thesis the following:

a) To the best of our knowledge, the first work to successfully find a policy capable of scoring goals in a VSSS environment, using a multi-agent paradigm with a centralized controller;

b) A comparison study of different multi-agent learning paradigm approaches and their performance concerning single-agent paradigms;

c) A study of cooperation in the VSSS robot soccer environment, highlighting its delicate balance between learning difficulty and motivation to cooperate;

d) VSSS RL environments, on both Central Processing Unit (CPU) and GPU physics calculations.

This work is structured as follows: Chapter 2 provides the theoretical foundation of reinforcement learning, which is the basis of this work. Chapter 3 reviews related work. Chapter 4 introduces the proposed methodology. Chapter 5 presents experiments and results in the VSSS environment. Chapter 6 examines the effects of changes in the environment on cooperation. and Chapter 7 summarizes the conclusions, limitations, contributions, and potential future work.

2 REINFORCEMENT LEARNING

RL is a type of machine learning that enables learning from interactions between an agent and its environment. This approach uses computational techniques to understand and automate goal-oriented behavior and decision-making processes from experiences. It is essential to differentiate between the problem and the solutions, as not doing so can lead to misunderstandings.

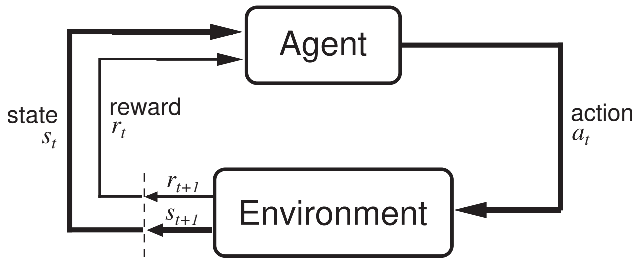
RL is a separate machine learning paradigm alongside supervised and unsupervised learning. It addresses the challenges of learning from interaction with the environment to achieve long-term objectives by maximizing a reward signal using feedback from the environment to evaluate the actions taken without explicit guidance or a complete understanding of the environment.

The learner must determine which actions will bring the greatest reward by attempting them. In the most stimulating and challenging cases, the actions may influence not only the immediate reward but also the following situation and, consequently, all future rewards. These two features, trial-and-error exploration, and delayed rewards, are the two most significant characteristics of RL (SUTTON; BARTO, 2018).

2.1 MARKOV DECISION PROCESSES

Markov Decision Processes (MDPs) are a formal framework that abstracts the problem of goal-directed learning from interaction (PUTERMAN, 2014). RL uses it to define the agent-environment interaction between the decision maker, called the *agent*, and the *environment*, the thing with which it interacts, comprising everything outside of the agent. MDPs possess the Markov property, which means that the current state determines future dynamics.

Figure 4 – The agent–environment interaction in Markov Decision Processes.



Source: (SUTTON; BARTO, 2018).

The agent-environment interaction (Figure 4), occurs in a sequence of discrete time steps in terms of actions chosen by the agent, states which are the basis for making a choice, and rewards for evaluating the options. Four components define MDPs: a state space S, an action space A, a State-transition function T, and a reward function R (SUTTON; BARTO, 2018).

The State Space S is a set of all the possible states s_t of an environment, and each state represents the environment entirely for a given time step, $s_t \in S$.

The Action Space A is the set of all actions a_t an agent can execute in an environment for a given state, $a_t \in A$.

The State-Transition Function T models the environment dynamics. It outputs the environment state for the next timestep s_{t+1} given its current state s_t and action taken a_t, $T(s_t, a_t) \mapsto s_{t+1}$.

The Reward Function R returns a scalar value representing the reward r_{t+1} for an action a_t taken in a given state s_t, $R(s_t, a_t) \mapsto r_{t+1}$.

The goal of RL is to find a policy π that maps actions the agent should execute a_t for a given environment state s_t, maximizing the total reward the agent receives in the immediate and following steps. This means that the agent should focus on something other than immediate rewards but rather cumulative rewards in the long run. We view any approach suitable for addressing such issues as the RL method. To achieve this, the expected return is maximized. The return is a function of the reward sequence:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2.1}$$

where γ is a parameter, $0 \leq \gamma \leq 1$, called the *discount rate*.

The discount rate determines the current worth of future rewards: a reward received in the future is only worth less than what it would be if received immediately. If $\gamma = 0$, then the agent ‘sees’ only immediate rewards. The agent becomes more concerned with future rewards as the discount rate increases.

A common issue in decision-making learning is *sparse rewards* (SUTTON; BARTO, 2018), when there is a large gap between rewards gained in the environment. This sparsity makes it hard for the learning algorithm to link an action with rewards that are far away. To address this problem, *reward shaping* (NG; HARADA; RUSSELL, 1999) is used. This approach involves defining intermediate rewards that are more frequent and provide better feedback on the quality of the actions taken in the environment.

2.2 ACTION VALUE METHODS

The concept of expected return leads to the concept of state value. We have a state-value function, $V_\pi(s_t)$, which gives us the expected return from a given state. It is important to note that this state value function is for a specific policy, as the expected return in a state depends on the actions taken from that point onward. The $V_\pi(s_t)$ function can be recursively written as:

$$V_\pi(s_t) = \mathbb{E}_{\pi,s_t}[G_t] = \mathbb{E}_{\pi,s_t}[r_{t+1} + \gamma V_\pi(s_{t+1})]. \tag{2.2}$$

We can use the V function to determine a policy by picking actions that will lead us to the state s_t with the highest value. To do this, we must be aware of the dynamics of the environment (T) to recognize which states are accessible from the current state and which action will take us to the desired state. This environment dynamic is only sometimes known, making it more advantageous to use the action-value function, $Q_\pi(s, a)$, which returns the expected return of rewards for an action taken in a given state, following a particular policy in subsequent steps. The Q function can also be recursively written as:

$$Q_\pi(s_t, a_t) = \mathbb{E}_{\pi,s_t,a_t}[G_t] = \mathbb{E}_{\pi,s_t,a_t}[r_{t+1} + \gamma Q_\pi(s_{t+1}, \pi(s_{t+1}))]. \tag{2.3}$$

Reinforcement learning problems can be solved using action value methods, which involve learning a Q function. We can estimate these values by taking the average values observed during multiple interactions between the agent and the environment. However, keeping track of all the averages may be impossible when the environment has many states and actions. In such cases, the agent can use a parameterized function that can be adjusted so that its return is close to the observed values (SUTTON; BARTO, 2018).

2.2.1 Deep Q-Network

The Deep Q-Network (DQN) (MNIH et al., 2015) combines neural network approximation with temporal difference learning, to effectively address complex reinforcement learning problems. It uses deep neural networks to approximate the Q function. This technique is particularly effective when dealing with complex and high-dimensional state spaces, including continuous ones, commonly encountered in various real-world scenarios.

The core of the DQN method is to learn an optimal Q_θ function, by minimizing a loss function $\mathcal{L}(\theta)$ given by the squared temporal difference error δ_t, using gradient descent to adjust the parameters θ from the neural network of the Q function.

$$\mathcal{L}(\theta) = \mathbb{E}_{s_t,a_t,r_{t+1},s_{t+1}}[{\delta_t}^2], \quad \text{where} \quad \delta_t = r_{t+1} + \gamma \max_{a_{t+1}} Q_{\theta'}(s_{t+1}, a_{t+1}) - Q_\theta(s_t, a_t). \tag{2.4}$$

Transition tuples $(s_t, a_t, r_{t+1}, s_{t+1})$ are collected during interactions with the environment. These tuples are subsequently stored in a memory buffer and sampled in batches for efficient learning. The parameters θ' are periodically updated with the most recent values of θ, $Q_{\theta'}$ is called the target Q function. This target network is a solution of the DQN algorithm to problems caused by changing targets during training, providing a consistent target to update the parameters θ (MNIH et al., 2015).

DQN stands out due to its off-policy nature, which allows it to update its action values using transitions from policies that are different from the one being optimized, this characteristic increases the stability of the algorithm and its ability to learn from a wide range of experiences.

It is important to note that the use of DQN is mainly restricted to problems with discrete action spaces. This restriction is due to the policy's approach, which involves selecting the action with the highest action value for a given state, usually through an argmax search.

2.3 POLICY GRADIENT METHODS

The policy gradient methods are distinct from the action value methods, as they learn the π_ϕ policy directly by adjusting their parameters ϕ to optimize the objective function $J(\phi)$:

$$J(\phi) = \mathbb{E}_{s\sim p(s|\pi_\phi), a\sim\pi_\phi}[G], \tag{2.5}$$

where $p(s|\pi_\phi)$ is the distribution of states given the policy π_ϕ. Parameters ϕ are adjusted in the direction of the gradient of the objective function $\nabla_\phi J(\phi)$. This gradient can be expressed using the function Q (SUTTON et al., 1999):

$$\nabla_\phi J(\phi) = \mathbb{E}_{s\sim p(s|\pi_\phi), a\sim\pi_\phi}[\nabla_\phi \log \pi_\phi(a|s) Q_{\pi_\phi}(s, a)]. \tag{2.6}$$

Policy gradient algorithms vary in the way they estimate the Q value. These algorithms are suitable for problems with continuous action space.

2.3.1 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) (LILLICRAP et al., 2015) is an off-policy reinforcement learning algorithm designed for environments with a continuous action space. It is a variation of the actor-critic approach, which involves learning a policy π_ϕ (the actor) and an action-value function Q_θ (the critic). DDPG utilizes deep neural networks for both the value function and the policy, with the policy performing a deterministic mapping from state to action.

It collects transition tuples $(s_t, a_t, r_{t+1}, s_{t+1})$ from the interactions with the environment in a memory buffer, these experiences are used in batches of samples for learning both the actor and the critic. The critic Q_θ, through gradient descent to minimize the loss function $\mathcal{L}(\theta)$, the squared temporal difference error δ_t of the batch, given by:

$$\delta_t = \mathbb{E}_{s_t,a_t,r_{t+1},s_{t+1}}[r_{t+1} + \gamma Q_{\theta'}(s_{t+1}, \pi_{\phi'}(s_{t+1}) - Q_\theta(s_t, a_t)], \tag{2.7}$$

where θ' and ϕ' are the target parameters for the critic and actor networks, they are adjusted by soft updates towards θ and ϕ (LILLICRAP et al., 2015). This is done to address the problem of changing targets in the parameter update, similar to the DQN algorithm (MNIH et al., 2015).

As neural networks are continuous and differentiable, the critic is used as an objective function $J(\phi)$ for the actor learning. The policy is learned by adjusting its parameters in a direction of the gradient of the objective function $\nabla_\phi J(\phi)$:

$$\nabla_\phi J(\phi) = \mathbb{E}_{s_t}[\nabla_\phi Q_\theta(s_t, \pi_\phi(s_t))]. \tag{2.8}$$

2.3.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a policy optimization technique that prevents drastic and unstable changes in the policy by limiting the update so that the new policy does not deviate too much from the policy used to acquire the experiences. It is an on-policy algorithm, as it collects new training data in each iteration rather than reusing old experiences (SCHULMAN et al., 2017).

The algorithm determines the probability ratio $r_t(\phi)$, which measures the difference between the policy before and after its update:

$$r_t(\phi) = \frac{\pi_\phi(a_t|s_t)}{\pi_{\phi_{old}}(a_t|s_t)}. \tag{2.9}$$

The objective of PPO is to discourage changes in the current parameters of the policy neural network (ϕ) that are far from the parameters used to collect experiences (ϕ_{old}). It does this by penalizing any updates that cause the ratio between the two to move away from 1.

Rather than optimizing the expected return directly, PPO uses the probability ratio $r_t(\phi)$ as a surrogate objective. A clip is applied to the ratio to discourage divergent modifications, limiting its value with a hyperparameter of the clip coefficient ϵ. This clipping encourages updates to remain within this limit. PPO optimizes the $\mathcal{L}^{CLIP}(\phi)$ clipped surrogate objective:

$$\begin{aligned} &\mathcal{L}^{CLIP}(\phi) = \hat{\mathbb{E}}_t \left[\min\left(r_t(\phi)\hat{A}_t, \text{clip}\left(r_t(\phi), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right], \\ &\text{where } \hat{A}_t = Q\left(s_t, a_t\right) - V\left(s_t\right). \end{aligned} \tag{2.10}$$

The Advantage, $\hat{A}_t$, is the difference between the estimated future rewards and the actual rewards received for a given trajectory. It provides a more informative value for policy quality than using the raw reward (SUTTON; BARTO, 2018).

PPO is an on-policy algorithm suitable for environments with continuous actions. However, it is less sample efficient regarding the number of interactions with the environment, as it requires considering the cost of obtaining these interactions. PPO is most advantageous in environments where it can interact without a considerable time cost to obtain experiences in large quantities.

2.4 MULTI-AGENT REINFORCEMENT LEARNING

Multi-Agent Reinforcement Learning (MARL) applies RL to systems with multiple agents interacting. These problems can be simplified to traditional RL problems when the other agents are considered part of the environment. However, when agents interact with other agents that are also learning, the environment becomes non-stationary, breaking the Markovian property, and making traditional RL methods inadequate (WEISS, 1999).

Several complexity factors characterize MARL, such as partial observations, ample state and action spaces, credit attribution, cooperative and competitive environments, and reward shaping. To categorize a problem as MARL, it does not need all of these factors to be present.

In Subsection 2.4.1, we discuss Markov Games, an extension of the MDP framework for multi-agent processes. Subsection 2.4.2 introduces various taxonomies of multi-agent learning problems, and Subsection 2.4.3 outlines the different learning paradigms available for multi-agent learning.

2.4.1 Markov Games

The presence of multiple agents can result in a variety of interpretations and extensions of the basic MDP structure. Whether the agents share the same state or each has its own, whether a single policy produces actions for all the agents (*Centralized Controller*) or each has its policy that produces its own actions (*Decentralized Controllers*), and whether there is a single reward value for all agents or each has its individual reward value (WEISS, 1999). These potential interpretations lead to multiple possible MDP formulations, as examples:

- (S, A, T, R): Shared system state $s \in S$, with shared system reward $r \in R$, where a single policy maps the state s to a joint system action $a \in A$.
- (S, A_n, T, R): Shared system state $s \in S$, with shared system reward $r \in R$, where each individual policy maps the state s to $Agent_n$ action $a_n \in A_n$.
- (S_n, A_n, T, R): Shared system reward $r \in R$, where each individual policy maps the $Agent_n$ state $s_n \in S_n$ to the $Agent_n$ action $a_n \in A_n$.
- (S_n, A_n, T, R): Individual $Agent_n$ reward $r_n \in R_n$, where each individual policy maps the $Agent_n$ state $s_n \in S_n$ to the $Agent_n$ action $a_n \in A_n$.

Markov Games (LITTMAN, 1994) integrates the general case of interactions of multiple agents into a generalized structure of an MDP. A *Markov game* is a tuple (N, S, A, T, R) where:

N is the number of agents in the game;

S is the state space, a set of all possible states of the environment;

A is the joint action space for the agents, A is made up of the action spaces of each agent $A = [A_0, ..., A_N]$, where A_n is the set of possible actions of $Agent_n$;

T is the state-transition function, it models the environment dynamics, outputting the probability of the possible next states given the state $s \in S$ and the joint action $a \in A$;

R is the joint reward function, R is composed of the reward functions of each agent $R = [R_0, ..., R_N]$, where R_n is the reward function of $Agent_n$, which maps the environment state s and the joint action a to $Agent_n$ reward value r_n.

2.4.2 Taxonomies of Multi-Agent Learning Problems

There are multiple approaches to characterizing the objectives and purposes of Multi-Agent Learning, leading to various taxonomies and classifications of techniques (TUYLS; WEISS, 2012). In the taxonomy of single-agent RL algorithms, since the problem is unambiguously defined, the taxonomy is mainly determined by the type of solution. On the other hand, in the multi-agent setting, the taxonomy is mainly determined by the type of problem rather than the solution (YANG; WANG, 2020). Defining the challenges of a multi-agent system is itself a research problem (SHOHAM; POWERS; GRENAGER, 2007).

A possible way to classify a MARL problem is concerning the type of activity, which can be *cooperative*, *competitive*, or a combination of both. *Cooperative* activities involve agents with the same goal, whereas *competitive* activities involve agents with antagonistic objectives. An example of a mixed activity is a soccer game, where each team has a shared goal that conflicts with the goal of the opponent team. Alternatively, we can base the classification on how the system defines the agents rewards. Agents who receive a single shared reward value are classified as *cooperative*, while agents who receive individual values are considered *competitive*. A single problem can define different rewards, and a mixed *cooperative-competitive* problem is possible when part of the reward value is shared and part is individual (HOEN et al., 2006).

The homogeneity of the agents can characterize a problem, agents can be homogeneous by their capabilities and decision procedures. Agents can be homogeneous by sharing the exact structure of observations of the environment, sharing the observation space, and they can also be homogeneous by having the same capacity of actions, sharing the same action space, or having the same physical capacities. Problems with heterogeneous agents can study the learning of distinct policies that best take benefit of the characteristic of each agent and its interactions or the learning of a robust policy for different types of agents (STONE; VELOSO, 2000).

We can classify MARL systems by communication mechanisms, communication enables agents to share information with other agents in the decision process. Problems that have these mechanisms may have them as an additional tool to enable better performance, or it may even be mandatory to use communication to solve the problem, this information that an agent communicates may contain state information not available to an agent, can indicate the actions that an agent will take, and can be used for agents to carry out planning together (STONE; VELOSO, 2000).

We can also classify problems by the agent awareness concerning the other agents, if it is aware of the other agents policies, knowing what actions they will take. Observability is also an environment characteristic. In a fully observable environment, the observations of an agent have complete information about the state, if the observations does not have complete information about the state, it is partially observable, a problem can have the complete information about the environment distributed among the agents, that is, the environment is partially observable for an agent, but gathering the observations from multiple agents it has complete information about the environment (BERNSTEIN et al., 2002).

2.4.3 Multi-Agent Learning Paradigms

There are multiple paradigms for multi-agent learning, as there are different interpretations of systems and also because of the multiple complexities that can arise when dealing with multi-agent problems, as shown in the various taxonomies (YANG; WANG, 2020). Some types of paradigm may be necessary due to the type of problem, and some seek to address specific problems. It is a matter of understanding the environment studied and selecting an adequate paradigm to avoid adding complexities to the training without being necessary. Some common paradigms are:

Joint-Action Learners (JAL): This paradigm uses a centralized controller, using a single policy that controls all agents, mapping the observations of the environment to the set of actions of all agents. This paradigm simplifies the challenges of multi-agent systems but suffers from high-dimensionality problems and does not apply to activities that do not allow central control;

Independent Learners (IL): In this paradigm, each agent learns its policy independently as decentralized controllers. This policy maps the observations of the agent to its actions;

IL with Shared Policies: We can use this paradigm when agents are homogeneous in their decision procedures, sharing the same space of states and space of actions, and objectives. It learns a single policy that maps the observations of each agent to its actions;

IL with Centralized Training: This paradigm uses additional information during training that is not accessible to the agent at the time of execution. The execution occurs in the same way as the IL paradigms. The additional information can address significant problems, such as the non-stationarity of the environment;

IL with Communication: It is also a form of an IL paradigm, but with communication mechanisms, this paradigm allows agents to exchange information with each other during execution to approach the activity, learning to use this communication system is part of the problem.

VSSS is a decision-making problem that RL methods can tackle. The goal is to optimize the number of goals scored by providing speed commands to the robot. VSSS falls under the category of continuous control problems, which makes it suitable for policy gradient methods. Additionally, there is freedom to utilize various learning paradigms since the overhead camera and the central external computer provide a complete view of the field without any limitations of decentralized control.

3 RELATED WORK

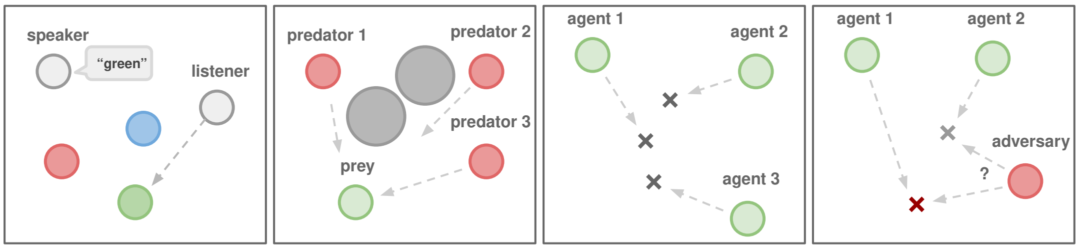
The type of problem that a MARL method addresses primarily determines its taxonomy. Consequently, it is essential to thoroughly understand the characteristics of the environment we are studying to know which types of methods we can use to address the proposed problem.

In Section 3.1, we will present the environments commonly used for MARL research. In Section 3.2, we will discuss the state-of-the-art MARL methods. In Section 3.3, we will examine the RL studies that address soccer challenges.

3.1 MULTI-AGENT DEEP REINFORCEMENT LEARNING ENVIRONMENTS

The **Multi-Agent Particle Environment** (Figure 5) was first introduced by MORDATCH; ABBEEL (2018) and released as part of LOWE et al. (2017) work. It is a popular environment in MARL due to its variety of tasks presenting multiple challenges, including cooperative, competitive, and mixed tasks, as well as partially or entirely observable environments, communication challenges, and homogeneous or heterogeneous agents. Each agent must interact with the environment and other agents to complete the objective of the task. Agents have continuous observations and discrete actions, with support for continuous actions. The Multi-Agent Particle Environment covers multiple challenges presented by MARL.

Figure 5 – Illustrations of some tasks from the *Multi-Agent Particle Environment*, shown in left to right order: *Cooperative Communication*, *Predator-Prey*, *Cooperative Navigation*, and *Physical Deception*.



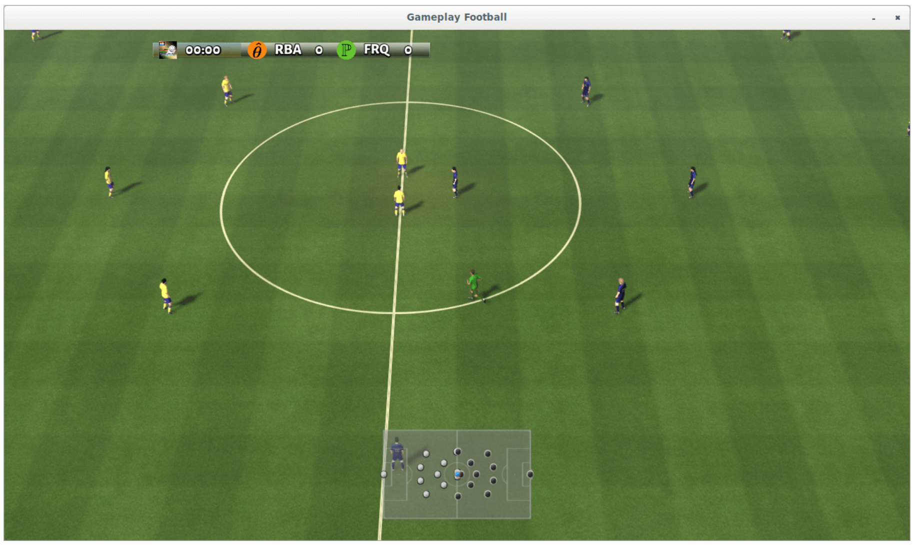
Source: (LOWE et al., 2017).

RL research has yielded impressive results in learning policies that can play the *StarCraft* video game with significant efficiency (VINYALS et al., 2019), *StarCraft* is a real-time strategy video game requiring players to manage their resources and control units through battles. SAMVELYAN et al. (2019) developed the **StarCraft Multi-Agent Challenge**, a Starcraft-based multi-agent challenge that does not model the standard game, but instead focuses on

MARL for decentralized control in multiple scenarios where each game unit is controlled by an individial agent. This environment has partial observations, as an agent observation is limited to its field of view, and scenarios with homogeneous and heterogeneous agents. The StarCraft Multi-Agent Challenge focus on solutions for decentralized control and is limited to discrete actions.

KURACH et al. (2020) proposed **Google Research Football** (Figure 6), a soccer simulation environment with continuous observations and discrete actions. The observations of an agent describe the complete state, homogeneous players, and cooperative task. It rewards the agent for scoring and approaching the enemy goal in a controlled manner. They initially proposed the single-agent environment, where an agent controls only one player at a time. However, it is extendable, and multiple studies use the multi-agent problem. It does not provide a path to transfer learned policies to the real world. It presents a video game challenge, and is limited to discrete control.

Figure 6 – A kickoff in Google Research Football environment.



Source: (KURACH et al., 2020).

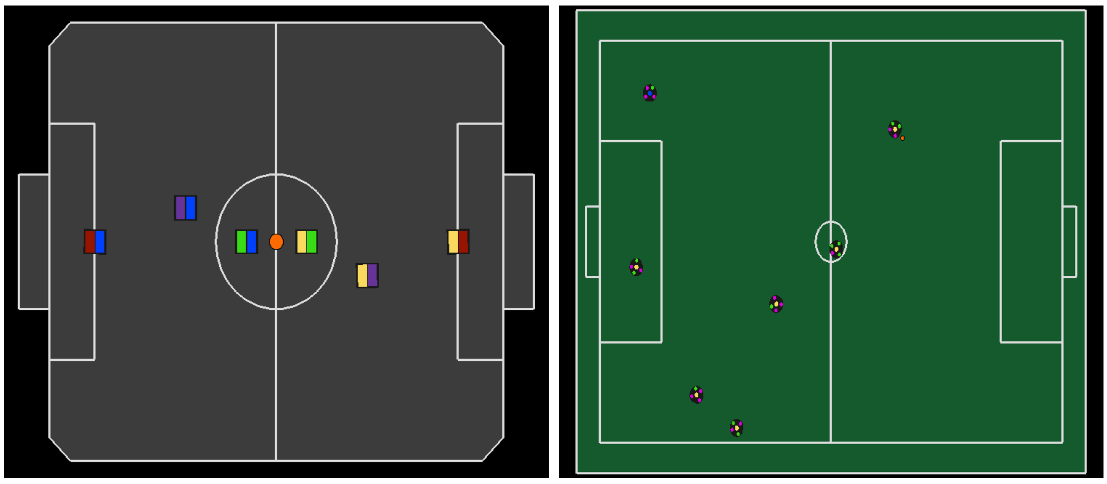
LIU et al. (2019) proposed the **MuJoCo Soccer Environment**, a simulated soccer environment with two agents on each team, continuous observations with a fully observable state and continuous low-level actions. They noted the difficulty of reward shaping in this environment and proposed using population-based training to optimize reward weights. In addition to the soccer match task, there are tasks in which the goal is to learn skills, such as agent movement and ball dribbling. This environment models a fictitious robot, which limits the ability to transfer learning to real robots. This environment uses the MuJoCo physics engine (TODOROV;

EREZ; TASSA, 2012).

RoboCup 2D Simulation (ITSUKI, 1995) is a two-dimensional soccer match simulation that deals with various multi-agent learning issues, such as partial observations, heterogeneous agents, decentralized execution, and communication. To be able to use the RL methods, it needs to be adapted. Although the complexity of the environment makes it difficult to develop end-to-end solutions, there are still open challenges that we can tackle in simplified settings. This is a challenging environment, being a interesting platform for studying decentralized control and communication in soccer tasks, it also models fictious players.

MARTINS et al. (2021) proposed the **rSoccer** (Figure 7), a robot soccer simulation environment framework, which has VSSS and RoboCup Small Size League (BURKHARD et al., 2002) environments, uses an adapted version of grSim (MONAJJEMI; KOOCHAKZADEH; GHIDARY, 2012) for physics simulations. rSoccer has two VSSS environments, a single-agent, *IEEE VSSS Single-Agent* and *IEEE VSSS Multi-Agent* environments, the second is a multi-agent environment with decentralized control. The agents are homogeneous, and their observations are continuous having the complete state of the environment, the actions are continuous and low-level values of the robot wheels desired speed, the environment has scoring and ball movement rewards shared between agents and individual rewards of robot movement and spent energy.

Figure 7 – rSoccer's IEEE Very Small Size Soccer (left) and RoboCup Small Size League (right) environments.



Source: (MARTINS et al., 2021).

rSoccer (MARTINS et al., 2021) is an adequate framework to use as a base, as it has VSSS environments. VSSS presents an exciting challenge due to the dynamicity of matches, state space, and continuous actions, since the competition uses a global view and centralized control. Additionally, as it simulates a real competition, it provides a viable path for RL research with real world applicability (BASSANI et al., 2020). We can add a centralized control environment to the ones proposed in rSoccer, allowing the study of methods of a different paradigm.

3.2 MULTI-AGENT DEEP REINFORCEMENT LEARNING METHODS

SUKHBAATAR; FERGUS et al. (2016) proposed ***CommNet***, a network architecture for problems where agents have partial observations and can communicate with each other. This method learns a policy and a communication model, allowing agents to cooperate. The authors demonstrated its effectiveness in problems with these characteristics, such as traffic management. They designed *CommNet* for tasks with discrete actions, but we can apply its ideas to those with continuous actions. In VSSS, we do not have the issues addressed by methods that deal with partial observations or the need for communication. However, an interesting factor of this architecture is its communication model, which has a dynamic size and is invariant to permutation. It is a practical method for future studies aiming to create robust solutions for environments with various robots in the field.

LOWE et al. (2017) **Multi-Agent Deep Deterministic Policy Gradient (MADDPG)** is an adaptation of DDPG (LILLICRAP et al., 2015) designed to address the issues of MARL in continuous control tasks. It uses the *Centralized Training with Decentralized Execution* paradigm, allowing additional information to be used during training that will not be available during execution. In training, the critic Q has access to the policies and observations of other agents. The centralized training allows it to address the problem of partial observations, and by knowing the actions taken by other agents, it can address the non-stationary environment problem, even if learning modifies the policies of other agents during training. Although MADDPG theoretically works with continuous actions, its performance was evaluated only in the *Multi-Agent Particle Environment* with discrete actions.

FOERSTER et al. (2018) proposed the counterfactual multi-agent policy gradients, **COMA**, a method that uses the paradigm of *Centralized Training with Decentralized Execution*. It seeks to address the problem of credit assignment. When multiple agents are cooperating to maximize a reward, the problem of credit assignment is about measuring how much the action of an agent contributed to the result obtained. It uses the concept of counterfactual baselines, using the centralized critic learned, it evaluates the effect of the different actions that an agent could take while fixing the action of the other agents, thus measuring how much an action taken contributed to the return obtained. They demonstrated *COMA* in *StarCraft unit* *micromanagement*, which predates *StarCraft Multi-Agent Challenge* but contains the problems of partial observations and decentralized execution. Credit assignment is a relevant problem in robot soccer, as agents need to know the contribution of their actions to the reward. However, it is not the topic of this study, and using *COMA* in a VSSS environment would require us to adapt the method to spaces of continuous actions.

RASHID et al. (2020) proposed **QMIX**, a value decomposition approach that combines individual value functions of agents into a global one using a mixing network. This approach deepens the studies in the *Centralized Training with Decentralized Execution* paradigm, which seeks to find decentralized policies in partially observable environments. They demonstrated the performance of this method in the *StarCraft Multi-Agent Challenge* and highlighted its good performance in environments with heterogeneous agents. However, QMIX, as an action-value function method, is limited to use in environments with discrete action spaces.

PPO (SCHULMAN et al., 2017) is an on-policy policy gradient method which is known to be inefficient in number of experiences and conceptually unstable. Moreover there is limited research on its application for MARL. However, it has yielded impressive results in single-agent RL. Two works have demonstrated promising results with similar training times, advocating using PPO in multi-agent environments. YU et al. (2022) work presents Multi-Agent Proximal Policy Optimization (**MAPPO**), which uses PPO in a *Centralized Training with Decentralized Execution* paradigm. In this method, the learned value function utilizes information from the entire environment state, which may not be available to the agent. They tested the method in the *Multi-Agent Particle Environments*, *Google Research Football*, and *StarCraft Multi-Agent Challenge*. They yielded results comparable to state-of-the-art off-policy methods. WITT et al. (2020) work studies Independent Proximal Policy Optimization (**IPPO**), arguing for the efficiency of PPO when used in a decentralized learning paradigm, even when the environment is partially observable. They hypothesize that policy clipping mitigates non-stationarity effects. Both methods make use of parameter sharing whenever possible. It should be noted that when the agent has complete information about the environment in its state, *IPPO* works similarly as *MAPPO*. As they are based on PPO, they support continuous action spaces.

3.3 REINFORCEMENT LEARNING IN SOCCER TASKS

HAUSKNECHT (2016) investigated cooperation from deep MARL in soccer tasks, he introduced the *Half Field Offense* task, based on RoboCup 2D Simulation (KITANO et al., 1997), the task involves learning to play an offensive situation in a reduced field size and reduced number of players, with partial observations and parameterized actions. The environment itself does not provide rewards, only the end-of-episode conditions, but reward shaping is necessary to learn it. In the single-agent case, the author rewards the agent for moving towards the ball, reaching it, kicking it towards the goal, and scoring it. He does not think that reward engineering is ideal, but it is necessary for the current methods. In the multi-agent case, he added credit assignment, modifying the reward depending on whether the agent has possession of the ball or the other has possession of the ball. He studied four conditions: independent learning, centralized controller, parameter sharing, and memory sharing. In an environment of two attackers, independent and centralized learning could not learn to use both players, while parameter sharing and memory sharing could. The difficulty increased in the task featuring a goalkeeper, where he observed some cooperation when using parameter sharing and memory sharing, but the obtained policy was not considered optimal.

HAUSKNECHT (2016) work found that a simple cooperation strategy can already achieve 80% of the goals. He hypothesizes that although joint work can improve the result, it is difficult to learn and remains an open question. His work shows learning difficulties in a football environment and has various findings. He points out that, despite being cooperative, the environment benefits the agent for “monopolizing” the reward, by being the one who scores the goal. Cooperation can improve the result, but learning to cooperate is much more challenging than learning a simpler policy that already gives partially good results.

OCANA et al. (2019) conducted a study on MARL in a RoboCup Standard Platform League (KITANO et al., 1997) robot soccer environment. They employed two paradigms, JAL and IL, and gave each agent an individual reward that took into account scoring, their own and their teammate’s movement towards the ball, and the ball’s movement towards the goal. They studied a case where they matched two robots against a goalkeeper and a defender with fixed policies. The observations were complete, and the environment had parametrized actions, being a mix of discrete and continuous. A high-level action selector existed between walking and kicking, paired with the walking speed value. They observed better results using JAL. An interesting result is that when removing the defender, the agent did not learn well and could only score without cooperation. This could be due to difficulty reduction, which removes the requirement for cooperation.

DELGADO (2019) investigated the use of reinforcement learning in VSSS, and achieved a learned policy that could score goals by rewarding scoring, robot movement towards ball,

ball movement towards goal and energy expenditure. He successfully learned policies for both discrete and continuous actions, with continuous actions yielding better results than discrete actions. This study only focused on the single-agent challenge, even though the environment contained six robots. The policy controlled one robot during the learning process, treating the other robots as part of the environment.

MARTINS et al. (2021) achieved success in learning in the *rSoccer* single-agent VSSS environment. However, when they attempted to learn in the multi-agent environment with parameter sharing, the outcome was not as good as the single-agent approach, even though they were controlling more robots. Additionally, they were unable to obtain successful results when using PPO in the *rSoccer* multi-agent environment.

BRANDÃO et al. (2022) investigated MARL in a VSSS environment, using independent learning and parameter sharing. Their VSSS environment had continuous observations and actions and simulated the game, including faults. They studied two forms of training, one of self-play and one against a predetermined heuristic. The agents could not learn against the predetermined heuristic but were successful with self-play. The authors divided the rewards into universal and specific. Universal rewards were for the ball motion toward the goal and the robot motion toward the ball, only considering the closest robot. Specific rewards were for defensive and offensive positioning, considering the angles between the robot, ball, and goals. Their study showed exciting advancements, such as using PPO (SCHULMAN et al., 2017), which other works could not, cooperative learning, self-play, and complete game modeling with opponents and rules. Their experiments used an evaluation process where they observed winning rates of the trained policy versus the heuristics and random opponents. We believe that training against the predetermined heuristic caused a complex task in learning initial behaviors, and they did not evaluate their results against learning with a single-agent approach.

Robot soccer is an exciting opportunity to explore MARL, and the VSSS category is a great setting, with previous work showing success in learning using reinforcement learning methods, including transferring the learned policies to real robots. It provides a stimulating environment, presenting various challenges in the field, being a dynamic environment with low-level continuous action and continuous states. This research will use the VSSS environment to assess cooperation learning through MARL.

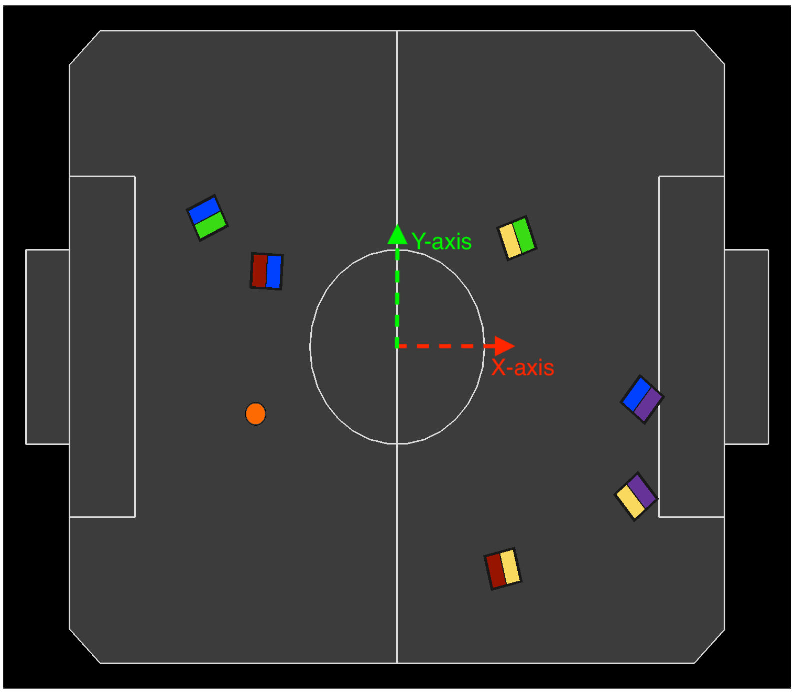
4 PROPOSED EXPERIMENTAL METHODOLOGY

In this chapter, we present our experimental methodology to continue the research of the VSSS environment introduced in MARTINS et al. (2021). We use an environment based on the one they proposed to compare the performance of multi-agent and single-agent paradigms in a VSSS environment. Furthermore, we extend the evaluation by measuring performance against intelligent teams. In Section 4.1, we explain the learning environment used, in Section 4.2, we describe the learning paradigms used, and in Section 4.3, we outline the evaluation procedure for the learned policies.

4.1 IEEE VERY SMALL SIZE SOCCER LEARNING ENVIRONMENT

The environment is a fundamental part of RL, it defines the task the agent is learning to solve. The learning environment (Figure 8) subject to this study is an VSSS environment based on MARTINS et al. (2021) work, noting that, since RL algorithms require an enormous amount of interaction steps to learn, this is a simulated environment. The *rSoccer* framework uses rSim for physics simulations through the *Open Dynamics Engine*.

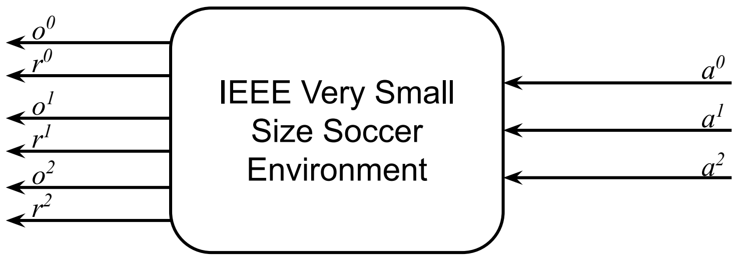
Figure 8 – Rendering of the IEEE Very Small Size Soccer environment on the rSoccer framework, with the X-axis (red) and Y-axis (green) shown.



Source: Author (2023).

The environment implements an interface derived from the *OpenAI Gym* (BROCKMAN et al., 2016) interface, a standard interface for RL environments, with adaptations to handle the multi-agent properties of the environment. Figure 9 shows a diagram of the adapted environment interface for the VSSS environment with three robots. a^n, o^n, and r^n are the action, observation, and reward of the robot with id n, respectively.

Figure 9 – Adaptation of the environment interface for three agents.



Source: Author (2023).

The main characteristics that define an RL environment are the actions it receives, the observations it returns, the rewards it provides, and the initial and terminal states. The following subsections describe these characteristics for the studied environment.

4.1.1 Actions

The actions that this environment receives are the desired wheel speed for the left and right wheels of the robot, the environment receives a normalized pair of floats for each robot ranging from -1.0 to 1.0. We will use a_0 and a_1 to refer to these actions. The environment converts this normalized value to the maximum linear speed of the wheels, ranging between -1.2 and 1.2 *m/s*.

Robots that the learning policy is not controlling receive random action values sampled from an Ornstein–Uhlenbeck Process (OU) (UHLENBECK; ORNSTEIN, 1930), this provides temporally correlated values, which increases exploration efficiency for physical control (LILLICRAP et al., 2015).

4.1.2 Observations

The agent perceives the state of the environment through the observations, as VSSS has a centralized controller which gathers the global state, we do not tackle partial observability, the observation has information of the whole state of the environment. The observations returned by the environment are 52 continuous values in an allocentric reference frame centered on the field center, presented in the following order: ball state, robot state, teammates state, and opponents state. The observations are homogeneous between robots, they share the same structure, however, each robot receives its perception of the observations, as the robot state refers to itself.

The ball state is its position (x, y) and linear velocity values (vx, vy). The agent state is comprised of its position (x, y), linear velocities (vx, vy), angle, represented by $(\cos(\theta), \sin(\theta))$, angular velocity (ω), and the previous actions (a_0, a_1). The state of its teammates and opponents consists of the same information as the agent state for each teammate concatenated, with the exception that the opponents information do not include the last actions values.

The position information is in *meters*, linear velocity is in *meters per second*, and angular velocity is in *radians per second*.

4.1.3 Rewards

Rewarding only goals, the main objective of a soccer match, would be an optimal reward function, as it would be unbiased, however, this reward is too sparse, especially in the initial learning episodes, making the environment very challenging for current RL methods. Reward shaping can reduce the sparsity by providing intermediate rewards (NG; HARADA; RUSSELL, 1999). Therefore, the environments of this work will use the same reward components defined in MARTINS et al. (2021). In addition to the reward for scoring a goal, they use three additional reward components: the ball motion toward the enemy goal and away from its own goal, the robot motion toward the ball, and the *robot energy* usage. The complete environment reward function is:

$$RW = GS * w_{gs} + BM * w_{bm} + RM * w_{rm} + RE * w_{re}, \tag{4.1}$$

where RW is the reward for a step in the environment, GS is the goal score component, BM is the ball motion component, RM is the robot motion component and RE is the robot energy component. w_{gs}, w_{bm}, w_{rm} and w_{re} are the scaling weights for the GS, BM, RM and RE components, respectively.

The goal score component, GS, and the ball motion component, BM, values are the same for every robot in the team, the robot motion, RM, and the robot energy component, RE, are individual for each robot.

The goal score component, GS, is the main component, its value is 1 if the agent team scores a goal, and -1 if a goal is scored against its team.

The ball motion component, BM, is positive when the ball moves closer to the opponent's goal post and away from the team's goal post, the value is negative when the contrary occurs. Its value is the difference in the potential value of the ball position on the field, this potential is the ball distance to the team's own goal post subtracted by the distance to the enemy's goal post:

$$\begin{aligned} BM &= (BP_t - BP_{t-1}), \\ BP &= \text{dist}_d - \text{dist}_a, \\ \text{dist}_d &= \sqrt{(x_{teamGoal} + x_{ball})^2 + 2 * {y_{ball}}^2}, \\ \text{dist}_a &= \sqrt{(x_{opponentGoal} - x_{ball})^2 + 2 * {y_{ball}}^2}, \end{aligned} \tag{4.2}$$

where BP is the ball potential, dist$_d$ is the distance from the ball to the back of the team goal, with the y-axis component doubled, dist$_a$ is the distance from the ball to the opponent team goal, with the y-axis component doubled. $x_{teamGoal}$ and $x_{opponentGoal}$ refer to the x-axis value of the position of the back of the goal, for its own and the opponent team goal, respectively, Figure 8 illustrate the axis, the back of the goal is used to reduce the bias of scoring in the center of the goal. The distance calculation doubles the weight of the y-axis component to discourage moving the ball to the field sides.

The robot motion component, RM, has a positive value when the robot moves toward the ball and a negative value when it moves away from the ball. We use the magnitude of the robot movement and if it moves in the ball direction. We calculate using the dot product of a unit vector pointing from the robot position to the ball position with the robot velocity vector:

$$\begin{aligned}
RM &= (\hat{rb} \cdot \vec{v}_{robot}) * \Delta t, \\
\text{ball}_{pos} &= [x_{ball}, y_{ball}], \\
\text{robot}_{pos} &= [x_{robot}, y_{robot}], \\
\vec{v}_{robot} &= [vx_{robot}, vy_{robot}],
\end{aligned} \tag{4.3}$$

where Δt is the environment physics time-step. $\hat{rb}$ is the vector from the robot position robot$_{pos}$ to the ball position ball$_{pos}$, and $\vec{v}_{robot}$ is the vector of the robot velocity.

The robot energy component, RE, penalizes the robot for energy usage. The idea is to stimulate the policy to use actions of a smaller magnitude to achieve the same results. This component is restrictive to the robot movement, conflicting with the core objective, however, this reward has been shown to make the training more stable and is adequate if we want to transfer the policy to real robots. Its value is the sum of the absolute values of the desired wheel speeds set for the robot:

$$RE = -(|\omega_{desiredLeftWheel}| + |\omega_{desiredRightWheel}|) * \Delta t, \tag{4.4}$$

where $w_{desiredLeftWheel}$ and $w_{desiredRightWheel}$ is the desired wheel speed for the left and right wheels, respectively, set by the actions sent to the environment in *radians per second*.

4.1.4 Initial and Terminal States

For the initial state of the environment, we randomize the positions of the robots and the ball in the field, we also randomize the robots angle and the ball velocity, however, the initial speed of the robots is always zero as initializing the robots with speeds can cause abnormal behaviors in the simulator.

The environment terminal states occur when a team scores a goal or reaches the maximum number of steps for an episode.

4.2 REINFORCEMENT LEARNING PARADIGMS

The learning paradigms define the number of robots acting on a learning policy and whether it is a centralized or decentralized controller. We define single-agent learning paradigms if only one robot policy learns during training, we will use two single-agent paradigms: Single-Agent

(SA) and Replicated Single-Agent (RSA). The multi-agent paradigms have the three robots of the team acting with a learning policy, we will use two multi-agent paradigms: IL, and JAL. The following subsections describe their characteristics.

4.2.1 Single-Agent Learning Paradigms

In single-agent paradigms, the learning policy controls a single robot and random actions sampled from a OU (UHLENBECK; ORNSTEIN, 1930) control the remaining robots of its team during training.

4.2.1.1 Single-Agent Paradigm

The SA paradigm is equivalent to the one used by MARTINS et al. (2021), which used this approch in its *IEEE VSSS Single-Agent* environment. It can be classified as single-agent, considering that the other robots in the field are seen as part of the environment and not learning. The distinction between a centralized and decentralized controller cannot be made for this paradigm, as it controls only one robot.

It should be able to learn to interact with robots in the environment but not leverage on teammates who have competent policies.

4.2.1.2 Replicated Single-Agent Paradigm

The RSA paradigm is equivalent to SA during training, and we will use the same policy learned from training SA. During the evaluation, it will control the three robots of a team, inferring an action from the policy for each robot's observations, a decentralized controller. This paradigm is equal to the solution BASSANI et al. (2020) used for running their learned policy for evaluation during a VSSS tournament.

This paradigm is possible since the robots are homogeneous, their observations have the same structure, changing only the perspective. We added this paradigm because SA only controls a single robot, adding a different perspective that controls the entire team, but with a simplified single agent training. We still classify it as a single-agent paradigm since it is learning to control only one robot.

4.2.2 Multi-Agent Learning Paradigms

In multi-agent learning paradigms, learning policies control the entire team during training.

4.2.2.1 Independent Learners

The IL paradigm is a decentralized controller, inferring an action from the policy for each robot's observations, in the same way as RSA. However, it also controls the entire team during training. This paradigm presents the non-stationarity problem of multi-agent learning as the policy of the teammates is evolving throughout the training.

MARTINS et al. (2021) used this paradigm in its experiment with the *IEEE VSSS Multi-Agent* environment. As robots are homogeneous, we utilize parameter and memory sharing (HAUSKNECHT, 2016), this improves training speed as we effectively train a single policy, and each environment step amounts to three experience tuples.

4.2.2.2 Joint Action Learners

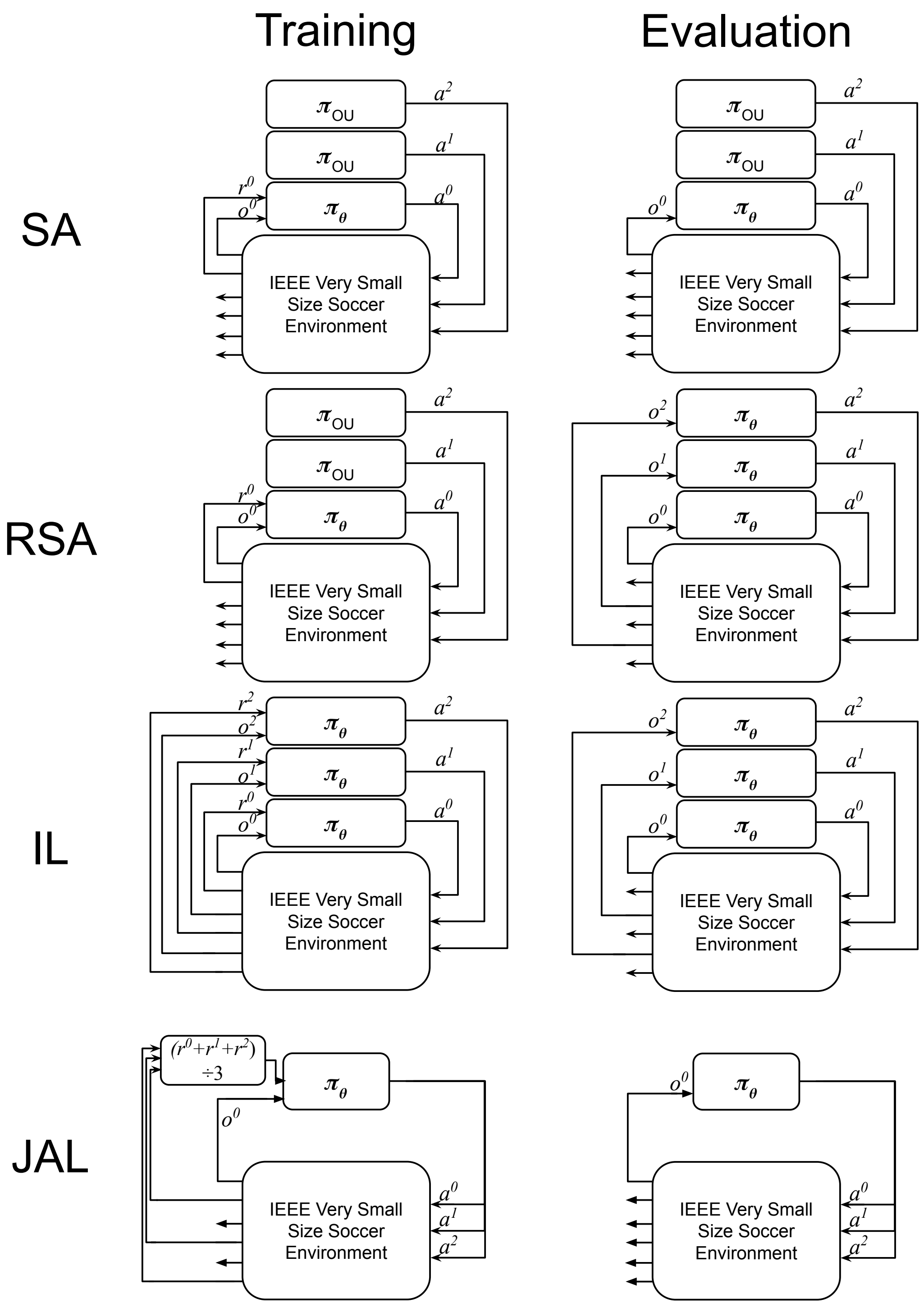
The JAL training paradigm uses a centralized controller, the learning policy simultaneously outputs the action for all team robots, tackling the non-stationarity problem. As the centralized controller controls all learning robots as one agent, this paradigm parallels a single-agent RL problem as the environment is stationary for the controller. As there is no extra information that it is not already using during training, the centralized control makes the training centralized by default.

The increase in the size of the policy inputs (observations) and outputs (actions) contrasts this paradigm's benefits as it increases the learning complexity. We minimize this by using just the observation of the id 0 robots as our robot's observations are allocentric and have complete observability. However, the number of robots the policy now controls multiplies its output size.

To handle rewards, we use the mean value from all robots on the team, as the GS and BM components are global, they are unchanged, but the RM and RE components are average values as those are individual components.

Figure 10 illustrates the interaction between the paradigms and the environment during the training and evaluation procedures. π_{OU} is a policy that takes random actions sampled from an OU process, while π_θ is a policy with trained parameters θ.

Figure 10 – Overview of each paradigm interface with the environment during training and evaluation.



Source: Author (2023).

4.3 EVALUATION PROCEDURE

This research will evaluate the effectiveness of multi-agent paradigms by comparing them to single-agent paradigms in a VSSS environment. We evaluate their performance on the main underlying challenge in a soccer match, scoring goals.

We train agents against enemies whose actions are random and with different reward components. However, what matters is the ability to score goals and the ability of the policy to extrapolate its learnings to different opponents with intelligent behaviors, behaviors not observed during training. This situation best represents a competition setting, where we want the policy to be superior to the various approaches brought by other teams.

The evaluation environment is initialized with random positions the same way as the learning environment. We only use the GS component of the reward during evaluation.

We engage in hundreds of matches to evaluate our policy, playing against a variety of opponents. These opponents include trained policies of all paradigms, random action opponents, a policy that samples actions from an OU, and stationary opponents that always send actions with zero value. We record the episode outcome, the policy either wins if it scores a goal, losses if it concedes a goal, or draws if it reaches the maximum number of steps. Additionally, we calculate a rating based on the number of each outcome:

$$\text{rating} = \frac{\text{Wins} - \text{Losses}}{\text{Matches}}. \tag{4.5}$$

where Wins is the number of matches won in the evaluation procedure, Losses is the number of matches lost, ad Matches is the total number of matches played.

Given the environment presented in this chapter, we study the ability of the different observed paradigms to learn policies capable of winning VSSS games. We will use the evaluation procedure to rank the different paradigms and compare their performances. We especially want to study the ability of multi-agent paradigms to use multiple robots in the team to get better results.

5 COMPARING LEARNING PARADIGMS ON IEEE VERY SMALL SIZE SOCCER ENVIRONMENT

This chapter presents experiments to compare the performance of different paradigms in a VSSS environment. Section 5.1 outlines the experimental setup, and Section 5.2 discusses the results.

5.1 EXPERIMENTAL SETUP

We conducted the experiments with the same environment settings and implementation of the DDPG algorithm as those used by MARTINS et al. (2021), this environment has simplifications relative to a complete VSSS match, the opponent team moves with random policies, and we do not implement the personal, attacking, and defensive foul game rules and the free ball condition (PINTO, 2023). We limited the number of steps in an episode to 1200 with a physics timestep Δt of 25 milliseconds, resulting in a maximum episode duration of 30 seconds, where we expect that trained policies to take around 5 seconds to score a goal against random policies. Table 1 shows the reward components weights, these values are equivalent to the one used in MARTINS et al. (2021) work.

Table 1 – Weights of the reward components of the IEEE Very Small Size Soccer learning environment.

Weight	**Value**
w_{GS}	10
w_{BM}	48
w_{RM}	20
w_{RE}	0.008

Source: Author (2023).

We made the following modifications to the DDPG base algorithm to improve training performance: to increase the diversity of collected data, we added an *experience collection to gradient ratio*, EGR, hyperparameter to control the ratio of the number of steps performed in the environment for each gradient step, and we collect interactions from multiple environments running in parallel, a *number of rollout processes*, NRP, hyperparameter defines the quantity of environments running in parallel; to provide better state value estimates we implement n-step bootstraping (SUTTON; BARTO, 2018), with the number of steps specified by the N-STEPS hyperparameter; To improve the initial exploration, we implement a decaying action noise, initializing the noise with an *initial noise sigma*, $\sigma_{initial}$, hyperparameter value, and at each *number of gradient steps per noise sigma decay*, $\sigma_{decaySteps}$, gradient steps, the noise value is reduced by the *noise sigma decay*, σ_{decay}, value until a minimum value of *noise sigma minimum*, σ_{min};

We used the same hyperparameters as MARTINS et al. (2021) work. Table 2 lists the hyperparameters used for DDPG training, Table 3 specifies the actor network architecture, and Table 4 sets the critic network architecture.

Table 2 – Deep Deterministic Policy Gradient hyperparameters values used for training.

Hyperparameter	**Value**
LEARNING RATE	0.0001
BATCH SIZE	256
GAMMA (γ)	0.95
TARGET NETWORKS TAU (τ)	0.999
INITIAL EXPERIENCE REPLAY SIZE	100000
EXPERIENCE REPLAY SIZE	5000000
TOTAL ENVIRONMENT STEPS	100000000
EGR	10
NRP	10
N-STEPS	3
$\sigma_{initial}$	0.8
σ_{min}	0.15
$\sigma_{decaySteps}$	3000
σ_{decay}	0.99

Source: Author (2023).

Table 3 – Architecture of the Deep Deterministic Policy Gradient actor neural network.

Layer	**Type**	**Input**	**Output**
0	Fully Connected (ReLU Activation)	Observations	400
1	Fully Connected (ReLU Activation)	400	300
2	Fully Connected (Tanh Activation)	300	Actions

Source: Author (2023).

We trained each paradigm using the DDPG algorithm, with three different random seeds against random action opponents. For the IL paradigm, we trained it with the algorithms

Table 4 – Architecture of the Deep Deterministic Policy Gradient actor neural network.

Layer	Type	Input	Output
0	Fully Connected (ReLU Activation)	Observations	400
1	Fully Connected (ReLU Activation)	400 + Actions	300
2	Fully Connected	300	Action Value

Source: Author (2023).

DDPG and MADDPG, respectively referred to as IL_{DDPG} and IL_{MADDPG};

The MADDPG algorithm uses extra information to ease training, as the robot observation in our environment has complete observability of the state, leveraging the teammates observations does not provide extra information, so we do not use it when learning a policy for this paradigm. However, we do use the information of the actions taken by the teammates and the access of its policy for training with a centralized critic which is a function of the robot‘s observation, its actions and its teammates‘ actions, tackling the problem of non-stationarity.

For evaluation, we reduced the maximum number of steps in an episode to 800, equivalent to game time of 20 *seconds*. This reduction accelerated the procedure and enabled us to run more matches. We tested 17 matchups: three random seeds for each of the five paradigms, random action opponents, and stationary opponents. We played 500 episodes for each matchup.

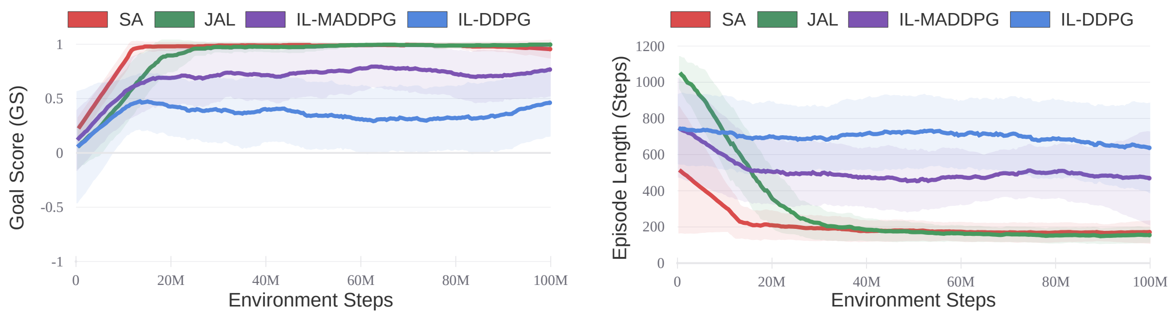
5.2 RESULTS

This section presents the results of the experiments for each paradigm, including performance during training and evaluation. We then compare the performance of the multi-agent paradigms, IL_{DDPG}, IL_{MADDPG} and JAL, to that of the single-agent ones, SA and RSA.

The experiments were executed using the Centro de Informatica (CIn) computing cluster, Apuana, using an Nvidia RTX 3090 GPU and 32 CPU cores. For the 100 million steps, each execution took around 17 hours to complete.

We begin by examining the graphs in Figure 11 that illustrate the value of the GS component of the reward and the duration of the episode throughout the training. The colored lines represent the mean value for the three training seeds and the shaded area around is the stan-

Figure 11 – Goal score (left) and episode length (right) during training in the IEEE Very Small Size Soccer learning environment.



Source: Author (2023).

dard deviation. These graphs demonstrate the outcome of what we are attempting to learn, namely, scoring goals and doing so efficiently. When we isolate the SA and IL_{DDPG} results, we can observe that we replicated the result obtained in the work by MARTINS et al. (2021). The single-agent approach quickly learns to maximize the score component and reduce the episode time. At the same time, the IL_{DDPG}, even controlling three robots, converges to a smaller score component value. This performance, not matching the single agent policy, and even being worse, leads us to believe that they are blocking each other. Watching the training recordings[1], we see situations in which this behavior occurs.

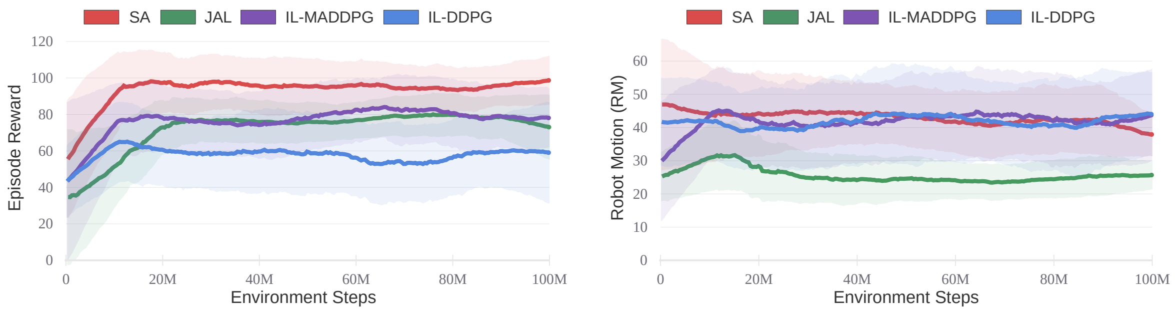
Observing the performance of IL_{MADDPG} in the same graph, we noticed an improvement compared to IL_{DDPG}, which confirms the hypothesis of MARTINS et al. (2021) that this approach would yield better results for multi-agent systems with decentralized execution. Nevertheless, its performance is still lower than that of SA, so we can draw the same conclusion as we did for IL_{DDPG}.

The JAL has an exciting outcome, as it converges in GS value and episode duration to values similar to those obtained by the SA, reaching the highest possible score. JAL requires more interactions to converge to optimal results than SA, which is expected since it is a more complex policy that returns three times more actions to control the three agents. This is actually a single-agent scenario, which provides greater assurance of convergence, and it observes the environment as stationary, contributing to better results than the decentralized execution paradigms.

The graphs in Figure 12 illustrate the total reward per episode and the value of the RM component during the training process. Although JAL converged to the same results as SA

[1] experiment recordings available at: <https://github.com/FelipeMartins96/exploring-deep-marl-in-ieee-vsss-recordings/tree/main/5-2>

Figure 12 – Total episode reward (left) and robot motion component (right) during training in the IEEE Very Small Size Soccer learning environment.



Source: Author (2023).

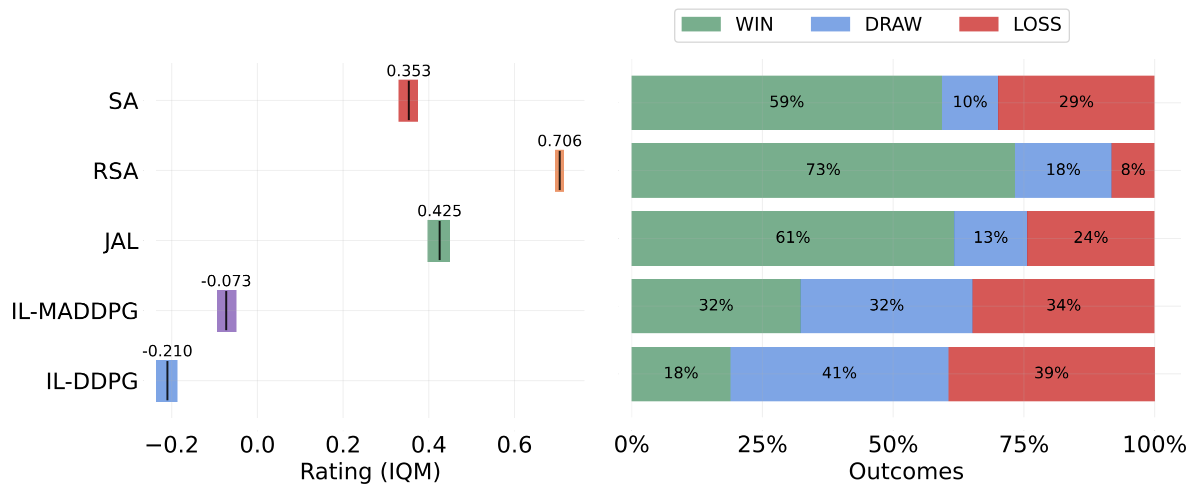
regarding GS and episode time, it achieved a lower total reward per episode due to a lower value of the RM component, as shown in the second graph. We also observed that IL_{DDPG} and IL_{MADDPG} achieved values of the RM component similar to SA despite their poorer performance in terms of GS. Possible explanations for the lowest values obtained by JAL could be how its reward is calculated as the average of the three robots, leading to lower values. Still, we also hypothesize that the policy sacrificed the RM component for cooperation to be possible. JAL could also have learned to control only one robot, an outcome we have seen in previous experiments. Watching the training recordings, we found that the policy is handling all three robots. However, we still found some situations where the approach does not control optimally one or more robots.

Since the paradigms with the best results, SA and JAL, converged to the maximum possible GS values, the evaluation will provide more information to compare their performance in a more challenging environment against intelligent opponents.

Figure 13 presents the evaluation results, including each paradigm's aggregated rating and match outcomes. The rating chart displays the Interquartile Mean (IQM) with a bootstrapped confidence interval, calculated as an aggregate metric over the paradigms of the opponent team played against during the evaluation procedure. We used the *rliable* python module to calculate these metrics (AGARWAL et al., 2021).

RSA is only present now and not in previous charts because it shares training with SA (it is the same agent replicated to control the other teammates). The multi-agent decentralized execution paradigms IL_{DDPG} and IL_{MADDPG} had the lowest ratings, even lower than SA, which controls only one robot. Still, we observe an improvement in the performance of IL_{MADDPG}, as expected from the training results. JAL received a higher rating than SA, which

Figure 13 – Aggregate rating (left) and percentage of outcomes (right) from the evaluation procedure for the learning paradigms.



Source: Author (2023).

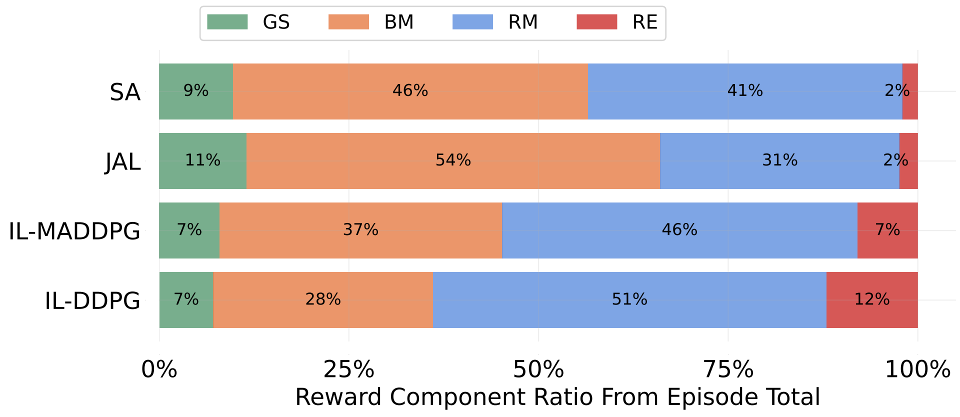
may indicate cooperation learning by JAL. Still, it is impossible to say, as the higher number of robots it controls could also explain the results. It is also likely that a better-positioned robot is under control at the start of the episode.

RSA had the highest rating, with a considerable margin, thus being a single-agent paradigm the best result among the experiments performed. We observe that it is improbable for RSA to learn to cooperate since we trained it with random-action teammates, yet it presents the best results. We investigated whether robots in RSA were blocking each other, similar to the IL paradigm. As both paradigms maximize their individual RM component, we analyzed the match recordings and observed instances of blocking, albeit less frequently than IL. The RSA policy also shows more dexterity. The difference is due to the SA training, where a stuck robot cannot rely on another robot to score.

These results are intriguing, making us question why the multi-agent paradigms obtained worse results since, in addition to controlling more robots, we expected them to learn to cooperate as they train with more information available with "intelligent" teammates. A hypothesis to explain the lower results is the more complex nature of multi-agent learning, preventing learning, but all paradigms converged to positive reward values during training.

We also hypothesized that the simplifications of the modeling of the environment when using random opponents and not including all game rules could have made cooperation less important. A shred of evidence is that both the JAL and the SA manage converged to the maximum value of the GS component. Even controlling only one robot in this environment, it is possible to obtain results close to the optimum during training, without observing difficult situations and not encouraging the agents to learn more advanced behaviors such as cooperation.

Figure 14 – Proportion of each reward component from the episode reward at the end of training in the IEEE Very Small Size Soccer learning environment.



Source: Author (2023).

Another hypothesis is that reward shaping, a simplification performed on the environment to enable training, may affect multi-agent paradigms ability, especially the RM component. Figure 14 shows that at the end of SA training, the RM component amounts to an average of around 42% over the trained paradigms and the hypothesis is that this reward can disturb cooperation since the robots will all tend to move as close as possible to the ball, blocking each other. This problem has additional implications in the IL_{DDPG} and IL_{MADDPG} paradigms, where each agent maximizes its own reward. As the RM and RE components are individual, we can consider their maximization as competitive behavior, even when the agents are on the same team. This issue is addressed in the next section.

5.3 REDUCING ROBOT MOTION REWARD COMPONENT BIAS

We investigate the hypothesis that reward shaping used in the VSSS environment with multiple agents leads to maximization of the reward components by RL that reduces the performance of the primary desired objective, scoring goals. Specifically, the hypothesis is that the RM component, which encourages the robot to move towards the ball, is detrimental when controlling multiple robots, as they all move closer to the ball and get in each other's way. As this component has a considerable weight in the total reward, it is not beneficial to the agent to renounce this component, even when it would result in gains in other components.

The scale of the RM component value compared to the other components and the discount factor, γ, used in the experiment affect its importance. The γ of 0.95 and a timestep, Δt, of 25 *milliseconds* make rewards obtained two seconds in the future less than 2% of their original value, thus increasing the influence of RM, which is received continuously, over the score received at a single moment. Although removing the RM component would be ideal, it is essential for learning in this VSSS environment (DELGADO, 2019).

BRANDÃO et al. (2022) also used a reward component to incentivize movement toward the ball. Their work only gave this reward to the robot closest to the ball. We tested this approach and others, such as transforming it into a reward component shared by the team and other forms of calculating it, such as displacement of the centroid of the robots positions towards the ball. Although we found the different approaches to this component attractive, we focused on reducing the influence of the RM as a whole rather than adapting it to the conditions of multiple agents.

We will then experiment with changing the values of the components weights to reduce the ratio of the RM component to the total and boost the γ. Both modifications make training more challenging. We will then compare single-agent and multi-agent paradigms again and investigate how the alterations influence the outcome.

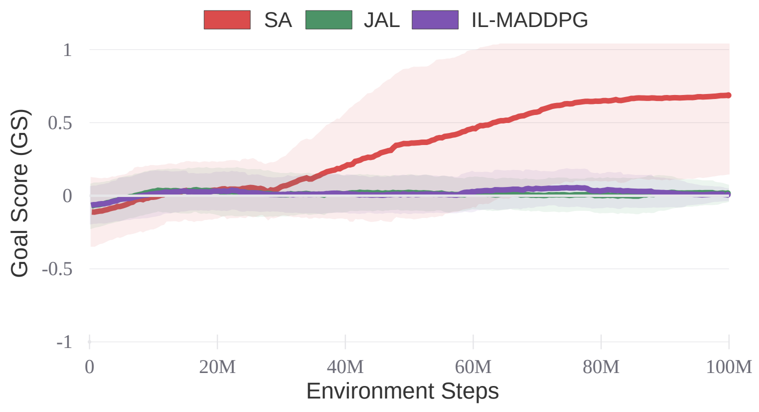
In Subsection 5.3.1, we outline the changes made relative to the previous experiments, and in Subsection 5.3.2, we present and examine the results.

5.3.1 Experimental Setup

We conducted extensive tests to reduce the RM component proportion. Unfortunately, the results were not successful, particularly for the multi-agent paradigms. Figure 15 displays the training performance of one of these tests, which only achieved limited success with the single-agent paradigm. Table 5 shows the changes in the reward components, with the RM and RE components halved and the GS component increased by a factor of 10. We increased the decay factor γ to 0.99 and did not test the IL_{DDPG} paradigm, as the MADDPG showed to be superior in previous results.

Since only SA could learn, it was impossible to compare the paradigms. Therefore, to continue our study of the reward proportions, we changed the training environment and algorithm to make them more robust, allowing learning in such a difficult scenario. We implemented the VSSS environment with the Isaac Gym platform (MAKOVIYCHUK et al., 2021) to cap-

Figure 15 – Goal score during training in the IEEE Very Small Size Soccer learning environment in the rSoccer framework with adjusted reward components weights.



Source: Author (2023).

Table 5 – Adjusted weights of the reward components of the IEEE Very Small Size Soccer learning environment to reduce robot motion component proportion.

Weight	Previous Value	New Value
w_{GS}	10	100
w_{BM}	48	24
w_{RM}	20	10
w_{RE}	0.008	0.004

Source: Author (2023)

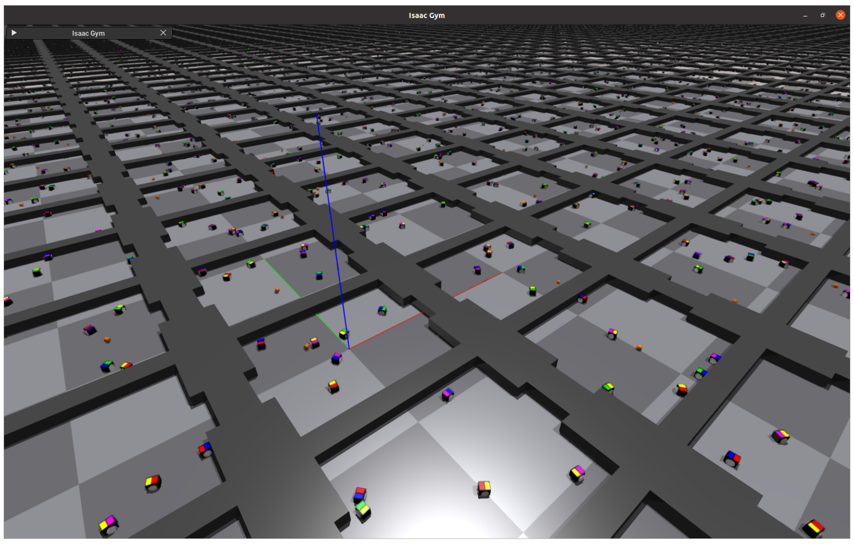
ture observations from multiple environments in parallel and we changed the RL algorithm to PPO (SCHULMAN et al., 2017). Subsubsection 5.3.1.1 outlines the modifications to the environment and Subsubsection 5.3.1.2 the changes to the training algorithm.

5.3.1.1 *Adapting The Very Small Size Soccer Environment to Isaac Gym Platform*

We adapted our environment to the Isaac Gym platform. Isaac Gym performs physical simulation computations on the GPU, this is beneficial for RL training, as it enables running a significant number of environments in parallel and also accelerates training by eliminating the bottleneck of memory transfer from the CPU to the GPU between environment interactions and neural network optimization procedures (MAKOVIYCHUK et al., 2021). Gathering data from multiple environments simultaneously increases data diversification, boosts exploration effectiveness, and enhances training stability (MNIH et al., 2016).

We used RobôCIn‘s real robot physical parameters to define the simulation robot. However, we did not validate the simulated robot‘s performance compared to the real robot. In this work, we did not intend to approach real-world problems, such as communication delays and observation noise.

Figure 16 – Rendering of hundreds of parallel IEEE Very Small Size Soccer environments on Isaac Gym platform, with the X-axis (red) and Y-axis (green) shown.



Source: Author (2023).

We maintained the same actions and observations of the rSoccer framework environment. However, we increased the timestep Δt to 50 *milliseconds* to accelerate training time, reduce the correlation of observations between steps, and have the same effect as increasing the gamma. We also decreased the maximum number of steps per episode to 600 to balance the timestep Δt increase. The calculation of the RM component was modified to be a function of the potential differences given by the distance between the robot and the ball, which is a more suitable format for reinforcement learning (NG; HARADA; RUSSELL, 1999).

$$RM = \text{dist}RB_{t-1} - \text{dist}RB_t. \tag{5.1}$$

Equation 5.1 shows the new calculation of the RM component, distRB is the distance of the robot to the ball.

Table 6 shows the adjusted weights of the reward components for the experiments on the Isaac Gym platform environment. We decreased the magnitude of the rewards as it yielded better results in the new setup and adapted the w_{RM} to the new formulation. Although we reduced the scale, the most significant adjustment was in the proportions to make the GS component the most significant by a large margin.

Table 6 – Adjusted weights of the reward components of the IEEE Very Small Size Soccer learning environment to reduce robot motion component proportion with new robot motion component calculation.

Weight	Value
w_{GS}	10
w_{BM}	2
w_{RM}	3
w_{RE}	0.001

Source: Author (2023)

5.3.1.2 Adaptation to the Proximal Policy Optimization Algorithm

We used the PPO algorithm to conduct our experiments. PPO has been effective in various continuous control tasks and has also demonstrated satisfactory performance in multi-agent learning, as shown recently in the works of WITT et al. (2020) and YU et al. (2022). We chose PPO because it is compatible with the highly parallelized Isaac Gym platform, which aligns well with the requirement for on-policy algorithms for more environmental interactions.

We based our PPO implementation on that made by HUANG et al. (2022b), adapting it to our environment. Various implementation details can affect the performance of PPO (HUANG et al., 2022a). We included the implementation improvements detailed in ENGSTROM et al. (2020) and ANDRYCHOWICZ et al. (2021), which studied the influence of various factors on the performance of the algorithm. We only list below the improvements applied here and refer the reader to the original papers for the details:

a) vectorized architecture;

b) orthogonal initialization of neural networks weights;

c) initialization of policy's neural network with a centered distribution with mean 0 and low standard deviation;

d) generalized advantage estimation (GAE);

e) mini-batch updates;

f) normalization of advantages;

g) clipped surrogate objective;

h) overall loss and entropy bonus;

i) global gradient clipping;

j) separate networks for policy and value functions;

k) continuous actions via normal distributions;

l) state-independent log standard deviation;

m) independent action component;

n) handling of action clipping to valid range and storage;

o) handling environment dones by timeout.

Additionally, we have included a parameter called the *speed factor* SF. This parameter controls the maximum speed of the robots that are not managed by the learning policy and the initial speed of the ball. During training, SF starts at 0 and increases to 1 linearly, the point in training in which SF reaches 1, given as a percentage of environment steps, is set by a *speed factor end*, SF_{end}, hyperparameter. SF helps the policy learn an initial behavior in a less dynamic environment, gradually increasing the difficulty during training, and acting as a curriculum learning (BENGIO et al., 2009). We performed a simple trial-and-error hyperparameter search to reduce training time and increase the validation score. We list the PPO hyperparameters used in Table 7.

Table 7 – Proximal Policy Optimization hyperparameters values used for training.

Hyperparameter	**Value**
TOTAL ENVIRONMENT STEPS	1000000000
NUMBER OF ENVIRONMENTS	3600
NUMBER OF STEPS	400
BATCH SIZE	1440000
NUMBER OF MINI-BATCHES	4
MINIBATCH SIZE	360000
UPDATE EPOCHS	7
LEARNING RATE	0.001
GAMMA (γ)	0.99
GAE LAMBDA (λ)	0.95
CLIP COEFFICIENT	0.2
ENTROPY COEFFICIENT	0.005
VALUE COEFFICIENT	4
MAX GRAD NORM	1.5
ADAM EPSILON	0.00001
SF_{end}	0.5

Source: Author (2023).

Table 8 displays the architecture of the actor's neural network, while Table 9 outlines the architecture of the critic's neural network. Previously, when using DDPG, we had two approaches for the IL paradigm, IL_{DDPG} and IL_{MADDPG}, as decentralized and centralized training. As our environment observations provide a full view of the environment and the PPO critic does not use actions, there is no distinction between centralized and decentralized training for the IL paradigms (YU et al., 2022).

Table 8 – Architecture of the Proximal Policy Optimization actor neural network.

Layer	**Type**	**Input**	**Output**
0	Fully Connected (Tanh Activation)	Observations	256
1	Fully Connected (Tanh Activation)	256	512
2	Fully Connected (Tanh Activation)	512	512
3	Fully Connected (Tanh Activation)	512	512
4	Fully Connected (Tanh Activation)	512	512
5	Fully Connected (Tanh Activation)	512	256
6	Fully Connected	256	Actions

Source: Author (2023).

5.3.2 Results

In this section, we present the results of our experiments with the different training paradigms in the new setup. We trained agents with five different seeds for each paradigm and then we plot the mean results (colored lines) and the standard deviation (shaded area).

The experiments were executed on Apuana with the same machine configuration of the previous experiments. For the 1 billion steps, each execution took around 4 hours to complete.

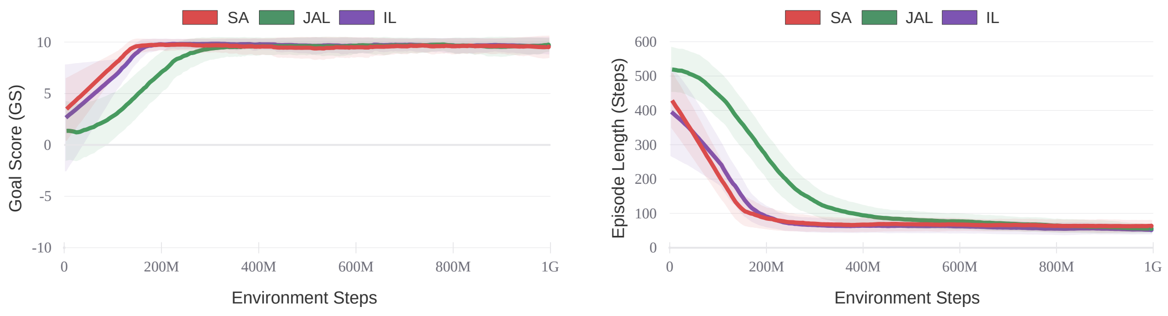
Figure 17 illustrates the performance of the three paradigms in training concerning the GS component and the episode's duration. All three paradigms converged to approximately the maximum possible value of the GS component, this shows an improvement of the IL paradigm, a relevant result to our hypothesis that the RM was conflicting. Furthermore, we

Table 9 – Architecture of the Proximal Policy Optimization critic neural network.

Layer	Type	Input	Output
0	Fully Connected (Tanh Activation)	Observations	256
1	Fully Connected (Tanh Activation)	256	512
2	Fully Connected (Tanh Activation)	512	512
3	Fully Connected (Tanh Activation)	512	512
4	Fully Connected (Tanh Activation)	512	512
5	Fully Connected (Tanh Activation)	512	256
6	Fully Connected	256	Value

Source: Author (2023)

Figure 17 – Goal score (left) and episode length (right) during training in the IEEE Very Small Size Soccer learning environment with adjusted reward components weights.



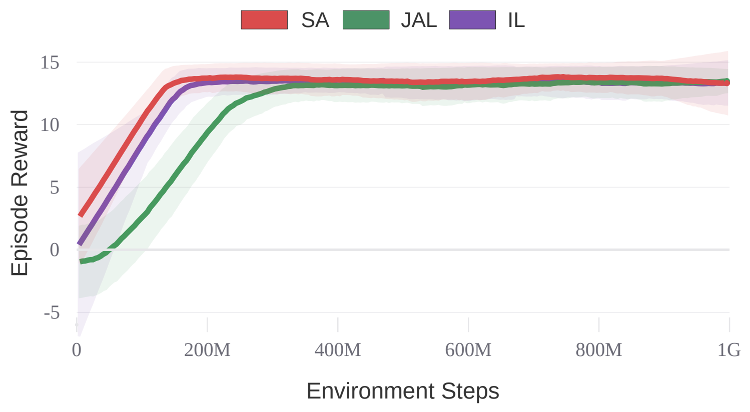
Source: Author (2023).

noticed that the episode time converged to similar values for all paradigms, and we continued to observe the training delay in JAL training to converge, as expected, as its policy has more actions.

Figure 18 illustrates the total reward per episode, and it is evident that the three approaches have converged to similar values, unlike the prior experiment, in which the multi-agent approaches converged to lower values.

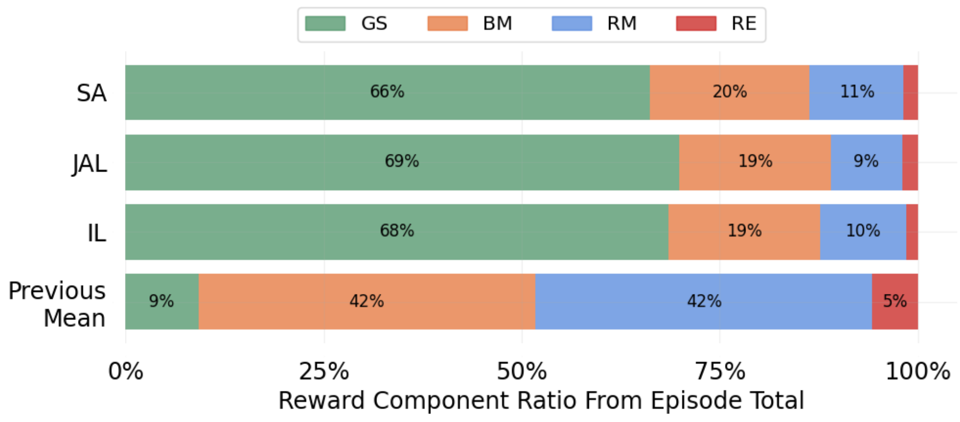
Figure 19 illustrates the percentage of each component of the total reward for the trained paradigms, and also includes the mean proportion of rewards for the previous experiments without the weights adjustments, tagged as *Previous Mean*. The new weights of the com-

Figure 18 – Episode reward during training in the IEEE Very Small Size Soccer learning environment with adjusted reward components weights.



Source: Author (2023).

Figure 19 – Proportion of each reward component from the episode reward at the end of training of each paradigm in the IEEE Very Small Size Soccer learning environment with adjusted reward components weights compared with the aggregate reward components proportions at end of training with previous weights.

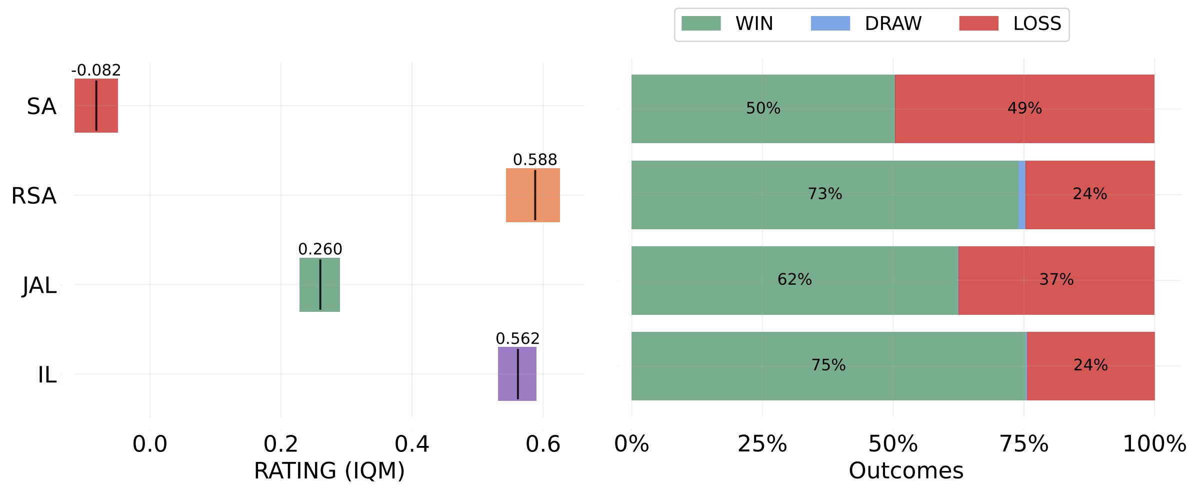


Source: Author (2023).

ponents have significantly decreased the RM component relevance, while also increasing the proportion of GS. Achieving working policies with these new proportions of reward components is remarkable, as GS is the underlying goal of the environment. Furthermore, with a discount factor γ of 0.99 and a mean episode time of around 3 seconds, the GS component retains more than 50% of its value. We also see that the component proportions are more uniform across paradigms when compared with the weights used in the previous experiments.

As GS reaches its highest value, the evaluation gives us more understanding of the policies learned, we question if the decreased RM component will improve the multi-agent paradigms performances. We changed the evaluation due to the faster speed that this version of the environment offers. Each matchup runs for 5,000 episodes, and we keep the same episode step limit of 600 steps as during training. Since we only have one IL paradigm type, we now have 12 different matchups, three seeds from each of the four paradigms, random action opponents, and stationary opponents. Since we train five seeds, we select the best three seeds for evaluation matchup opponents, ranked by the episode reward value achieved during training.

Figure 20 – Aggregate rating (left) and percentage of outcomes (right) from the evaluation procedure for the learning paradigms with adjusted reward components weights.



Source: Author (2023).

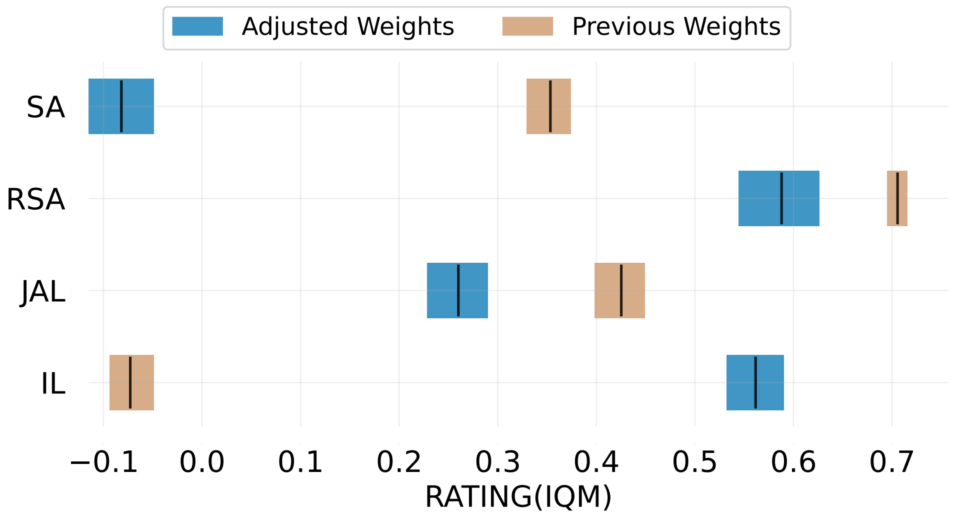
Figure 20 illustrates the calculated rating and the outcomes of the matches from the evaluation procedure.

The JAL, which treats the rewards jointly, has not improved its relative rating over the RSA paradigm. It is still unexpected that JAL has a result far from the best, even without cooperation, we anticipated that the result of multi-agent paradigms would be at least similar to RSA. SA is now, in fact, the paradigm with the worst rating, as expected, since it controls only one robot.

Regarding the evaluation matches outcomes, they reflect the results seen in the rating. RSA and IL have the highest win percentages and lowest loss percentages. An interesting point is a general decrease in the percentage of draws, we think differences in the simulator or the longer duration of the evaluation episode are possible causes for that.

Figure 21 compares the evaluation results of this experiment with adjusted reward weights relative to the previous experiment. It is impossible to compare the two experiments in absolute values, as we conducted them on different platforms with different opponent policies. However, we can observe the change in distribution within each experiment. From this graph, we make two observations, the first is that the paradigms that control three robots during the evaluation

Figure 21 – Aggregate rating from the evaluation procedure for the learning paradigms with adjusted reward components weights compared with previous weights



Source: Author (2023).

are now closer together in the lead, and the SA is now isolated the worst, as expected. Additionally, we observe significant improvement of IL.

Checking recordings[2], we verified that JAL learned to control all three robots, and we verified cases of cooperation, such as positioning. Notably, it does not have the behavior of always moving all robots to the ball, including cases of moving away of the ball to get into position. However, it has the worst result of those controlling three robots, even with more cooperation cases.

IL and RSA have similar behavior, where everyone constantly moves the robots closer to the ball, but they do not get in the way as much as in the previous experiment. It is more similar to the RSA of the previous one. However, the RSA has more dexterity in general. We even observed cases of cooperation in the IL, as robots positioned themselves, not moving directly to the ball and passing it. However, we must determine if it is by chance due to a lack of dexterity. Even in the case of having cooperation, RSA has better results due to its dexterity.

The results of this chapter support the hypothesis that the RM component conflicts with the underlying objective of this environment, observed mainly in the IL paradigm. However, the multi-agent paradigms still need to demonstrate an advantage over the single-agent approaches, we believe in the hypothesis that this environment model does not necessitate

[2] experiment recordings available at: <https://github.com/FelipeMartins96/exploring-deep-marl-in-ieee-vsss-recordings/tree/main/5-3>

learning cooperation since random opponents do not present much difficulty once the agent learns to move the ball.

With the improvements in training performance obtained after adapting the environment to the Isaac Gym platform (MAKOVIYCHUK et al., 2021) and using the PPO (SCHULMAN et al., 2017) RL algorithm, we expect now to be able to model the environment closer to the real problem and still have the policies learning. This will be investigated in the next chapter.

6 INVESTIGATING ENVIRONMENT COMPLEXITY AND COOPERATION

Although cooperative behaviors were observed in our previous experiment when the reward weights were adjusted, they needed to be more robust to outperform the single-agent paradigm. This chapter investigates the hypothesis that the training environment is too simple for cooperation to be essential. The simplified training environment against random action opponents does not incentivize the policy to utilize the three robots effectively. An intricate policy with cooperation does not amount to proportional rewards for its complexity compared to initial solutions.

We looked into two approaches that modify the complexity of the learning environment with changes that model it closer to the complete problem. We hope that it will encourage learning cooperative behaviors, improving the multi-agent paradigm performance. In Section 6.1, we experiment with changing the opponents policy which we train against, from random action opponents to intelligent opponents. In Section 6.2, we introduce the attacking foul condition into the environment.

6.1 LEARNING AGAINST INTELLIGENT OPPONENT POLICIES

In this section, we investigate the complexity of the environment and the cooperation in it when learning against intelligent adversary policies. Subsection 6.1.1 outlines the experimental setup, while Subsection 6.1.2 presents and examines the results of the experiments.

6.1.1 Experimental Setup

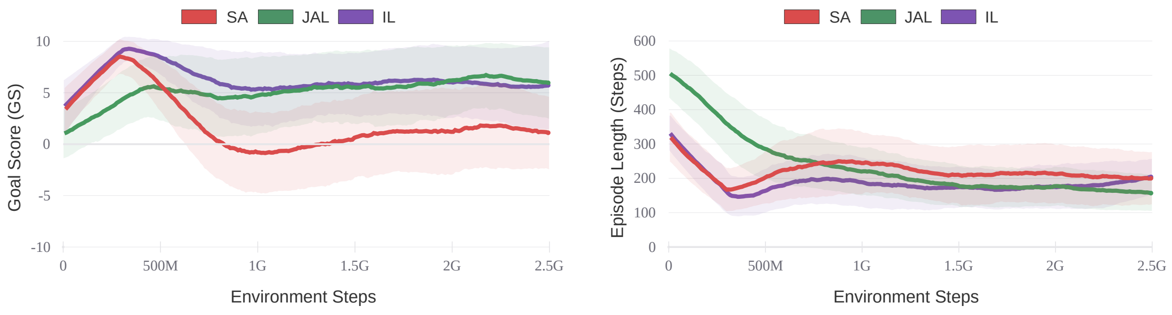
We kept the changes made for the Section 5.3 experiments, which altered the environment to use the Isaac Gym platform, as it led to more reliable learning, including the ratios of reward components that reduced the impact of RM. Additionally, in experiments we conducted outside of this work using the rSoccer framework environment, we were not able to learn policies when we trained against intelligent opponents. We expect that the modifications that enabled us to learn using the new reward component ratios will also make it possible to learn against intelligent policies.

We used the RSA paradigm as the opposing team policy, selecting one of the trained policies from the experiments described in Section 5.3. The choice to use RSA was because it had the highest rating in the evaluation. The only hyperparameter of the algorithm that we changed was the *total environment steps*, which we increased from 1 billion to 2.5 billion steps, as we wanted more training time for this more complex environment, thus, although the value of the hyperparameter SF_{end} remained the same at 0.5, since this value implies that the SF value will reach its peak at the midpoint of the training, the SF value will now reach its maximum value at 1.25 billion steps.

For the evaluation process, since we did not change the environment platform, we kept the same trained policies for the SA, RSA, IL, and JAL paradigms that we used in the Section 5.3 evaluation process, they are the policies with the best training results from the Section 5.3 experiments. Since matchups are against the same opponents, we can compare the evaluation process ratings of this section with the evaluation process results of Section 5.3.

6.1.2 Results

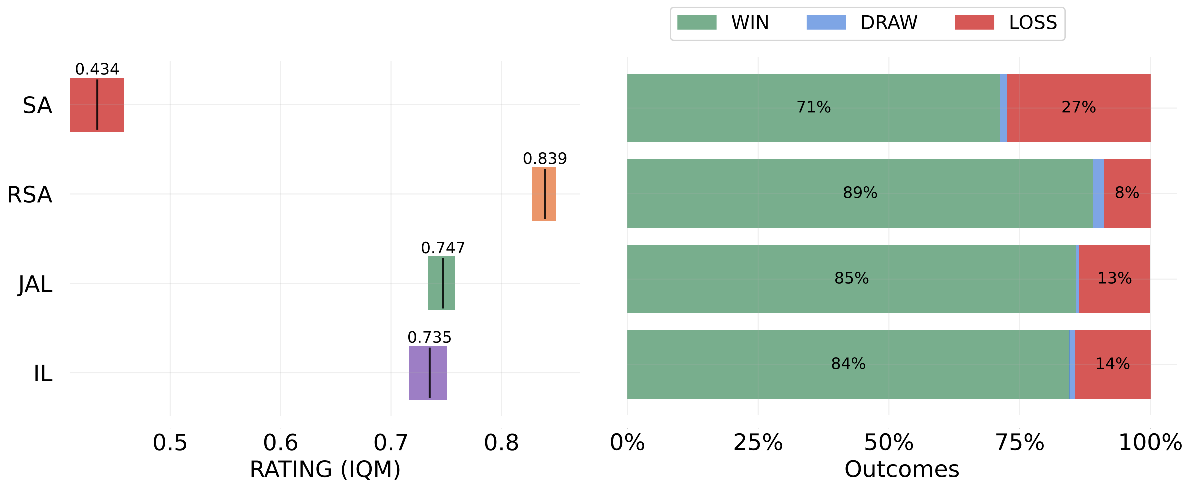
Figure 22 – Goal score (left) and episode length (right) during training in the IEEE Very Small Size Soccer learning environment against intelligent opponents.



Source: Author (2023).

Figure 22 illustrates the GS component and the duration of the episode during training. In these experiments, the paradigms did not reach the maximum GS value, indicating that the environment had become more complex and the agent could no longer win all matches. This complexity increase allowed us to compare the performance of the single and multi-agent paradigms. The multi-agent paradigms achieved a higher GS value than the SA, which we expect since the single-agent paradigm only trains with one robot. The decrease in the result until the middle of the training is due to the *speed factor*, which causes the environment to reach its peak complexity at the midpoint of training.

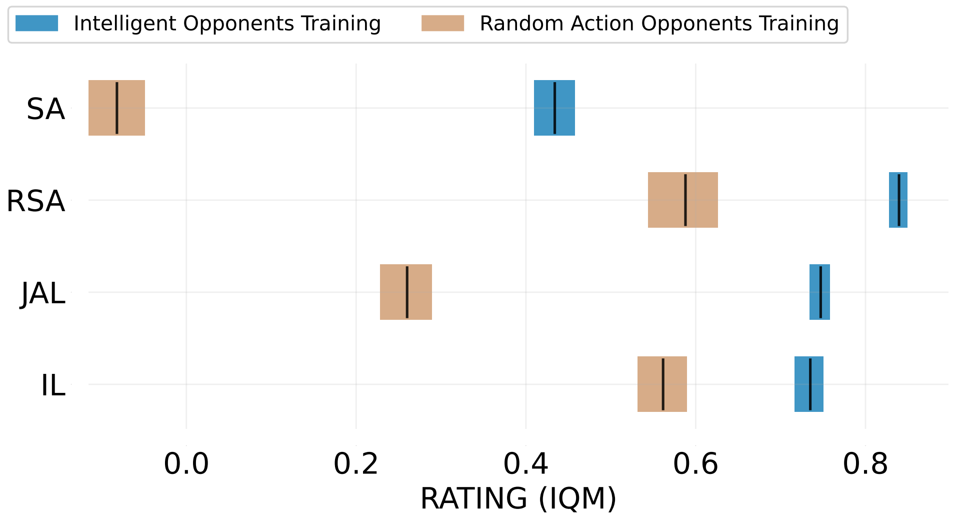
Figure 23 – Aggregate rating (left) and percentage of outcomes (right) from the evaluation procedure for the learning paradigms trained against intelligent opponents.



Source: Author (2023).

Figure 23 displays the calculated rating and matches outcomes from the evaluation process. The RSA paradigm achieved the best overall result, and the JAL and IL multi-agent paradigms had similar results, trailing the RSA rating and with a significant advantage to SA, which had the expected worst result as it controls only one robot.

Figure 24 – Aggregate rating from the evaluation procedure for the learning paradigms trained against intelligent opponents compared with random action opponents.



Source: Author (2023).

Figure 24 displays the evaluation rating of this experiment and the previous experiment (from Section 5.3), in which we trained against random action opponents. All paradigms achieved a higher rating when trained against intelligent opponents. However, contrary to our expectations, the RSA increased its edge against the IL. The fact that all paradigms, when trained against intelligent policies, improved their results relative to policies trained against random action opponents shows that the previous environment does not encourage learning more complex policies due to the limited challenge it proposes.

We highlight that the performance of JAL has improved, now matching that of IL, which illustrates the problem: in the environment against random opponents, it has no incentives to learn the optimal use of the robots it controls, as observed in the evaluation, while the IL paradigm does not suffer from this problem, as it shares the policy, learning of one robot or all is effectively the same thing.

In an analysis of the recordings[1], all paradigms move all robots close to the ball. JAL is the most cautious about getting too close. In it, we observe cooperative behaviors where one of the robots moves significantly away from the ball to position itself. In IL, we verified three cooperative behaviors: one of the robots blocks the opponent robot in a movement contrary to the direction of the ball, they seem to use a technique of pushing the ball side by side, and they seem to coordinate a movement where they send one robot at a time towards the ball. This last behavior seems to avoid hitting the wall and produce momentum, pushing the ball against opponents. We did not observe any cooperative behavior with RSA, and agents sometimes get stuck blocking each other. Although we observe some level of cooperation in the multi-agent paradigms, RSA still achieves the best result, possibly due to more dexterous control.

The JAL and IL approaches have yet to surpass the performance of RSA, even when we model the environment more closely to the actual problem with intelligent opponents. We hypothesize that we could not train optimal multi-agent policies due to limitations in the methods, the multi-agent paradigms should be capable of matching the RSA performance. Following these experiment results, although the training environment is now more complex, it still needs to present a clear advantage to using cooperating policies. We investigate other approaches to make the environment more conducive to cooperation.

[1] experiment recordings available at: <https://github.com/FelipeMartins96/exploring-deep-marl-in-ieee-vsss-recordings/tree/main/6-1>

6.2 LEARNING TO HANDLE ATTACKING FOULS

Despite being trained against intelligent opponents, the multi-agent paradigms failed to outperform RSA in previous experiments. Therefore, we hypothesize that the learning environment, with the increased complexity of intelligent opponents, is also not conducive to cooperation. In this section, we delve deeper into the complexity of the environment and cooperation when learning with VSSS attacking foul conditions (PINTO, 2023). We present the experimental setup in Subsection 6.2.1, and analyze the results of these experiments in Subsection 6.2.2.

6.2.1 Experimental Setup

The attacking foul rule requires coordination among the players so that no more than one robot enters the opposing team's goal area when the ball is also there. The team is penalized when this coordination does not occur, as the referee halts its attacking play. The rules indicate that following an attacking foul, the referee should resume the game with a goal kick for the opposing team (PINTO, 2023). In our training environment, we decided to end the episode when an attacking foul occurs, we added the Attacking Foul reward component, AF, with a value of -1 when an attacking foul occurs.

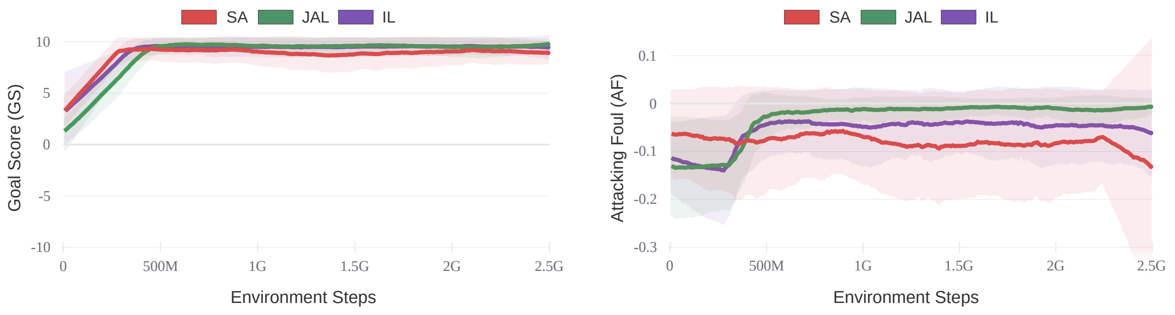
Apart from the environment now checking for attacking fouls, we maintain the same experimental configuration used on Section 6.1, with the increased number of training steps, but we revert to using random action opponents. We only check for attacking fouls caused by the team we are training.

We maintain the opponent policies used in evaluating the previous experiments on the Isaac Gym platform. However, for the evaluation procedure of this experiment, we will check for attacking fouls caused by the team we are evaluating, ending the match, affecting the rating and outcomes results.

6.2.2 Results

Figure 25 shows GS and AF components throughout training, all paradigms quickly converged to the maximum value of GS and kept AF close to zero. We observe a high AF penalization for the multi-agent paradigms at the beginning of the training that then de-

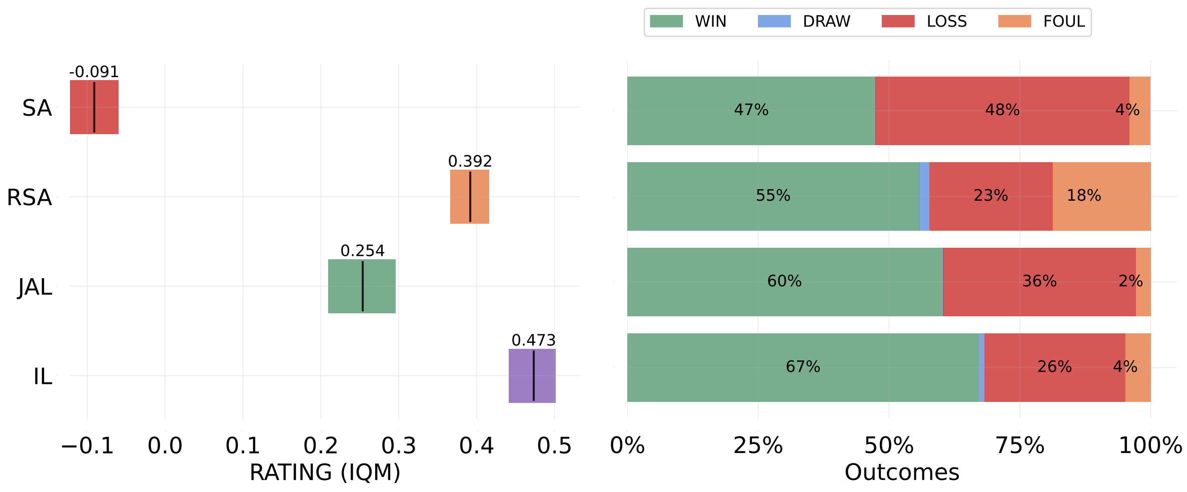
Figure 25 – Goal score (left) and episode length (right) during training in the IEEE Very Small Size Soccer learning environment with attacking fouls.



Source: Author (2023).

creases. This is because when learning the initial behavior of moving the ball towards the enemy goal, the multi-agent paradigms start to commit attacking fouls, so being penalized, it learns to avoid them. We do not see the same increase in AF in SA training, as it controls only one robot, it cannot cause attacking fouls, except when a teammate randomly enters the opponent's goal area. The SA teammates entering the opponent goal area do not occur frequently, hence the low AF penalization, but when it does, the SA has no control to remove them, explaining the higher AF value compared to the multi-agent paradigms, which control all robots.

Figure 26 – Aggregate rating (left) and percentage of outcomes (right) from the evaluation procedure for the learning paradigms trained with attacking fouls.



Source: Author (2023).

As the training of all paradigms converges to optimal values, the evaluation procedure gives us more information about their respective performances. Figure 26 shows the aggregated rating result and the outcome of the matches for the paradigms in the evaluation procedure, we see that IL was the paradigm with the best result, the first experiment where a multi-agent paradigm is the best. This result shows that cooperation is advantageous in this environment. The JAL paradigm, however, presented a lower result than RSA.

Observing the outcomes of the evaluation games shown in Figure 26, we noticed that RSA commits more fouls than the multi-agent paradigms by a significant amount, as it could not learn to coordinate to avoid fouls. Despite its lower rating result than RSA, JAL had a higher percentage of wins but is penalized by a higher proportion of losses. Despite not learning to cooperate, we assume that RSA has a more optimized control, if we consider matches in which the outcome was an attacking foul as situations of probable victory, RSA would have the best result, as seen in previous experiments.

Analyzing the recordings[2] of the evaluation procedure matches, we found that JAL does not learn to use all three robots to score goals, in one of the seeds, we see the policy always using the same robot to score. This non-optimal usage of the resources may explain its inferior performance to other paradigms. Although JAL does not use all robots to score goals, it controls all three to move close to the ball and coordinates so that other robots do not enter the opponent's goal area to avoid attacking fouls.

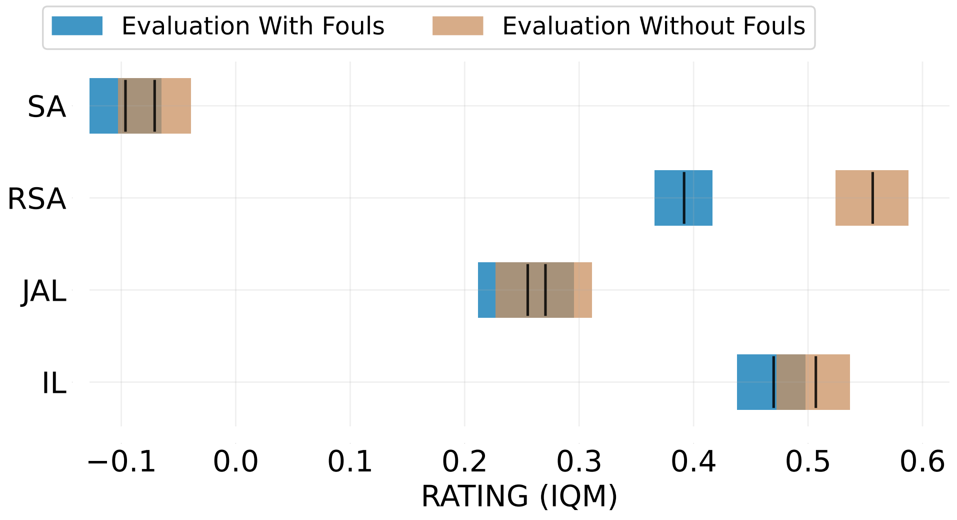
We also observed that RSA learned to "kick" the ball from outside the goal area. The agent pushes the ball and stops moving before reaching the goal area, letting it continue with its inertia, thus avoiding the foul. It learned this behavior as a way not to commit attacking fouls when a teammate is in the opponent's goal area, as it cannot control the teammates, however, we still see RSA causing attacking fouls. We do not consider this behavior as cooperation as this is an individual behavior learned due to the impossibility of cooperation. In the IL paradigm, we can observe that the robots avoid entering the opponent's goal area when a teammate is inside.

We performed additional experiments to understand better the effect of attacking fouls. In Subsubsection 6.2.2.1, we analyze how adding the attacking foul condition changes the environment and training complexity, and in Subsubsection 6.2.2.2, we evaluate the paradigms training them both with attacking fouls and against intelligent opponents.

[2] experiment recordings available at: <https://github.com/FelipeMartins96/exploring-deep-marl-in-ieee-vsss-recordings/tree/main/6-2>

6.2.2.1 *Evaluating How The Attacking Foul Condition Affects Environment and Learning Complexity*

Figure 27 – Aggregate rating from the evaluation procedure with attacking fouls compared with the evaluation procedure without attacking fouls for the learning paradigms trained with attacking fouls.

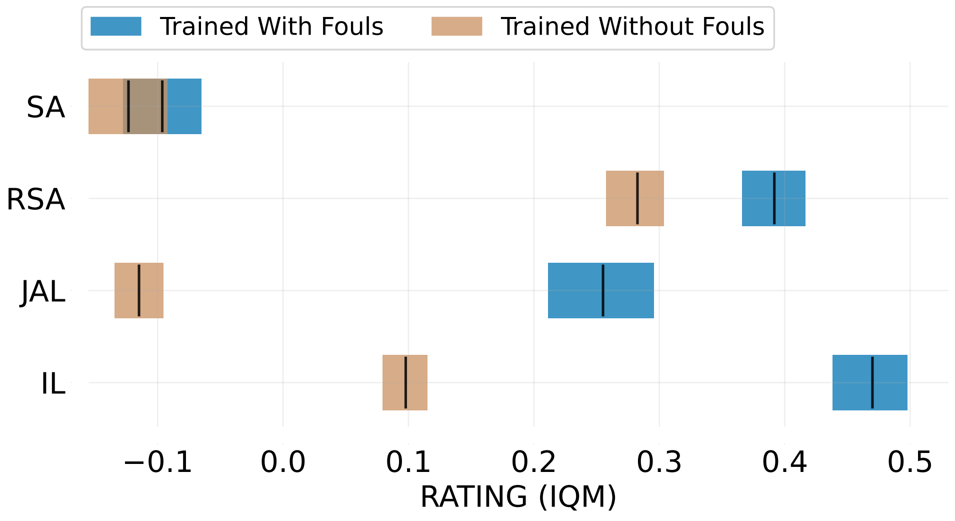


Source: Author (2023).

To compare how the attacking foul condition increases the environment complexity, we also evaluate the paradigms in an environment without the attacking foul condition. Figure 27 compares the evaluation rating with and without the attacking foul condition. We observe the reduction in rating of all the paradigms when evaluating the foul condition as indicative of its increase in complexity. Furthermore, compared to other paradigms, the more considerable rating reduction from the RSA indicates that cooperating is advantageous when we add the attacking foul condition to the environment.

To study the learning performance in this more complex environment, we ran the policies trained in the Section 5.3 experiment, in which we trained in an environment without attacking fouls, in the evaluation procedure checking for attacking fouls. Figure 28 compares its result with the paradigms trained in an environment with attacking fouls. We see that SA and RSA improved when training with fouls, an unexpected improvement since we initially expected cooperation to be necessary. This improvement of the single-agent paradigms indicates that they lean to avoid committing attacking fouls, as observed in the recordings that it learns the kicking behavior. However, we observe a more significant improvement in the multi-agent paradigms than in the single-agent, as avoiding attacking fouls through cooperation to obtain better results.

Figure 28 – Aggregate rating from the evaluation procedure with attacking fouls for the learning paradigms trained with attacking fouls compared to trained without attacking fouls.



Source: Author (2023).

6.2.2.2 *Training with Attacking Fouls Against Intelligent Opponents*

When we add the attacking foul condition, cooperation becomes more critical, as the JAL paradigm can learn to cooperate, we hypothesize that it can topple the result of the RSA paradigm. However, the rating achieved by the JAL paradigm was lower than that of RSA.

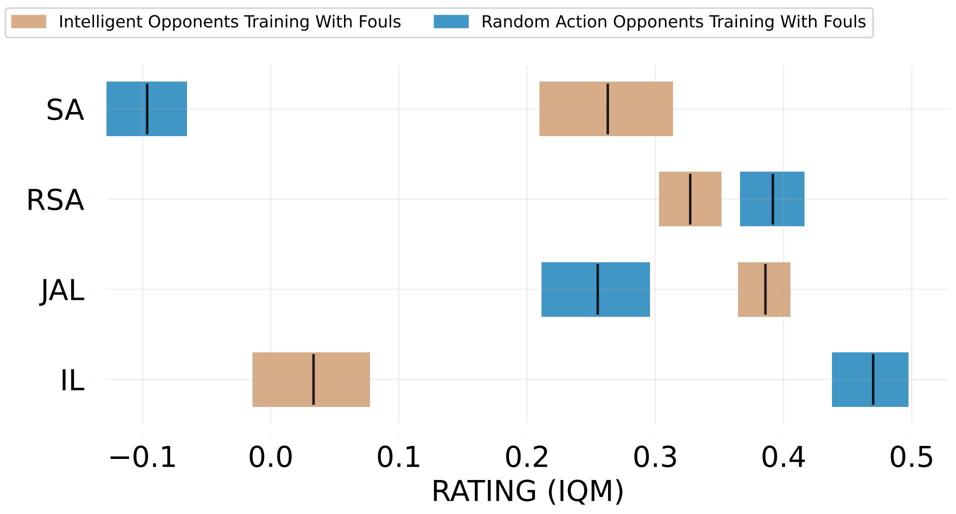
As observed in previous match recordings, the JAL paradigm, which we trained in an environment where attacking fouls were not allowed, did not learn to take advantage of controlling three robots. This non-optimal learned policy may be due to learning methods' limitations. Although optimizing the methods is a possible solution, we focus on investigating the environment.

In the following experiment, we test increasing the difficulty of scoring a goal during training in an environment that penalizes attacking fouls using an intelligent opponent policy. We check if we can replicate the performance gains we obtained in the JAL paradigm trained against an intelligent opponent policy compared to training against random opponent actions in the environment without attacking fouls, as shown in Figure 24.

We used the same opponent policy as in the previous experiment, training against intelligent opponent policies of Section 6.1, an RSA policy trained in the Section 5.3 experiment. Although the opponent policy does not avoid committing attacking fouls, the environment does not penalize the opponent for causing them.

Figure 29 shows the evaluation rating obtained when training to avoid attacking fouls

Figure 29 – Aggregate rating from the evaluation procedure with attacking fouls for the learning paradigms trained with attacking fouls against intelligent opponents compared to trained against random action opponents.



Source: Author (2023).

against random action opponents compared to training to avoid attacking fouls against intelligent opponents. As expected, we observed an improvement in the performance of JAL, which is the best result compared to other paradigms trained to avoid attacking fouls against intelligent policies. However, its performance is not superior to RSA and IL paradigms trained to avoid fouls against random action opponents.

We see an unexpected degradation of the IL performance. This degradation may have happened because the environment has become too complex for this paradigm, harming its learning, especially considering that, in IL, the environment is non-stationary. Returning to the learning challenge hypothesis, where the complete environment becomes too difficult for learning and simplifications limit the exploration for optimal policies.

SA was the paradigm that had the most significant performance gains in the modified environment, as it is less affected by the attacking foul rule, it can better take advantage of the more complex environment to find a more optimized policy. We expected that JAL and IL paradigms would be able to achieve the same gains as SA, but they did not.

Finally, it is interesting that despite the significant improvement of SA, RSA worsened, highlighting the cooperation requirement of this environment.

The experiments show the challenges of using MARL in the VSSS robot soccer environment. Learning policies using multi-agent paradigms does not necessarily result in a better approach than single-agent paradigms. When we model the environment closer to the real one, the multi-agent paradigms have the advantage of being able to cooperate. However, the increase in training complexity also amplifies the advantages of the single-agent paradigms for its more efficient learning.

7 FINAL CONSIDERATIONS

This thesis focused on applying deep MARL to cooperative learning in the VSSS robot soccer environment. On different environment configurations, we employed four paradigms: SA and RSA single-agent paradigms, IL and JAL multi-agent paradigms. We evaluated and compared the performance of the multi-agent and single-agent paradigms through an evaluation methodology proposed, and analyze the results to understand the importance of cooperation in the environment.

In Section 7.1, we discuss conclusions taken from the experiment results. In Section 7.2, we discuss our work contributions to science. In Section 7.3, we list the limitations of our work. In Section 7.4, we discuss future works.

7.1 RESULTS CONCLUSIONS

Here we organize our conclusions based on the goals set out in this research. In Subsection 7.1.1, we will discuss our findings on the use of deep MARL for learning policies in the VSSS environment. In Subsection 7.1.2, we will present our conclusion on the different paradigms employed. In Subsection 7.1.3 we will discuss our conclusions on the role of cooperation in the environment.

7.1.1 Learning Policies Using Single-Agent and Multi-Agent Deep Reinforcement Learning Paradigms in the IEEE Very Small Size Soccer Environment

Our research builds upon previous studies that trained agents to score goals in a VSSS environment using the SA and IL paradigms. However, we took a step further by applying the JAL paradigm in the same environment and achieved successful results.

A previous study by MARTINS et al. (2021) utilized the SA and IL paradigms to train agents in the VSSS environment. We replicated their results using the DDPG algorithm and also implemented the MADDPG algorithm, which improved the IL performance as they had proposed. Furthermore, we addressed issues that MARTINS et al. (2021) encountered, such as the reward function being biased towards keeping the robot close to the ball and the lack of success using PPO. We did this by adapting the environment to the Isaac Gym platform, which enabled massively parallel environment interactions. We show an increase in IL performance after learning with modified reward component weights.

BRANDÃO et al. (2022) successfully trained agents to score goals using PPO in the IL paradigm using self-play and noted cooperative behaviors, their environment included all the game rules. Although we did not experiment with self-play or the environment with all game rules, we experimented with the attacking foul rule and successfully trained agents in the IL paradigm.

Our research trained agents to score goals and observed cooperative behaviors on policies learned using the IL and JAL paradigms. By successfully training agents using single-agent and multi-agent paradigms in the same environment, we could draw comparative conclusions on their performance.

7.1.2 Paradigms Performance on IEEE Very Small Size Soccer Environment

The IL paradigm leverages the homogeneity of the agents in this environment, allowing the sharing of experiences and policy parameters between agents. This paradigm presented increased learning speed compared to JAL due to the sharing of experiences and because its policy also has the same number of actions as the single agent. Also notable in this comparison, is its joint learning of the control of the three robots due to the sharing of policy among agents. We observed as downsides of this paradigm its lower guarantee of convergence due to the non-stationarity caused by each robot being effectively an individual agent, and being more sensitive to increases in the complexity of the environment. This paradigm was also the most affected by the individual reward components that conflicted with scoring goals.

The experiments with the attacking foul condition exemplify the performance of the IL paradigm well. It presented the best results controlling the three robots and learning to coordinate so that only one enters the area. However, in the tests carried out which increased complexity by training it with attacking foul against intelligent opponents, this paradigm had a loss of performance. The environment proved to be too complex for learning, showing that advances with this paradigm require further improvements to MARL methods.

We observed a lower convergence speed during training for the JAL due to its increased action space dimensionality. However, JAL has a better guarantee of convergence and is less affected by the non-stationarity of the environment since it controls the three robots as if it were a single agent. However, the convergence towards non-optimal policies was observed, being especially more prevalent in simpler environments, such as the environment with random action opponents, in which learning to control a single robot is sufficient to score consistently, and, when training with attacking fouls, the paradigm can learn to use only one robot with moderately effective results due to the advantage of being able to attack with only one agent to prevent attacking fouls.

JAL learning to control only a single robot was not the prevalent outcome, in most cases we can see it moving the three robots and even cooperative behaviors such positioning, but in the same policies, we see situations where it does not control a robot in a favorable situation or we can identify it having a preferred robot used to score, since this behavior of controlling a single agent does not work well against intelligent opponents during evaluation, JAL produced poor results in this situation. In the experiments in the environment with the RM component with significant weight, JAL was encouraged to control the three robots towards the ball, thus reducing the tendency to control only one robot.

Both SA and RSA, presented surprisingly positive outcomes, showing good results in all environment configurations, being the best in almost all environments with the exception of the environment with the attacking foul condition. The fact that single-agent paradigms learn so much even though they are controlling only one robot during training is an unexpected result of this research. We conclude, in addition to the single-agent better guarantees of convergence, that happens due to training benefitting from not having to divide learning between control and coordination, while at the same time, the learning environment is intrinsically more challenging since it has fewer resources controlling only one robot, and as it cannot rely on teammates during training, so the focus is placed on learning a more robust policy.

7.1.3 Cooperation in the IEEE Very Small Size Soccer Environment

As simplifications must be made from the complete VSSS match to enable learning policies, our study initially focused on learning cooperation on those environment configurations with the simplifications.

We based our initial environment configuration on the environment proposed by the rSoccer framework (MARTINS et al., 2021). In this environment, the RSA paradigm outperformed the multi-agent paradigms and we discovered that the RM component was a heavy-weighted component of the total reward for the environment. When controlling multiple robots, the RM component conditions them to flock around the ball, hindering the performance as they get in each other's way. This behavior is more prevalent in the IL paradigm, as each robot aimed to maximize its individual RM component during training.

In our second environment configuration, we adjusted the proportions of the reward components to reduce the impact of the RM component, expecting to observe cooperative behaviors, resulting in better performance of the multi-agent paradigms. While we did observe cooperative behaviors in the multi-agent paradigms, their performance was inferior to that of the single-agent method, RSA, indicating that, in this environment configuration, cooperation may not yield a performance advantage.

To increase the complexity of the environment and align it more closely with real VSSS matches and the evaluation procedure, we studied a third environment configuration. We modified the opponent team to employ intelligent policies. We expected that the ability to cooperate among the multi-agent paradigms would lead to learning policies with superior performance in this more challenging environment. However, although we observed cooperative behaviors and performance of all the multi-agent paradigms improved when compared with the policies trained against random action opponents, the multi-agent paradigms' results remained similar to those of the RSA paradigm. Therefore, cooperation did not prove necessary for optimal performance in this environment. However, we did observe an improved performance of all the paradigms compared to the previous experiment, which trained against random action opponents.

In our final environment configuration, we added the condition of an attacking foul, which required coordination between agents to ensure that only one robot at a time enters the opponent goal area to score. In this configuration, we observed that the multi-agent paradigms learned to cooperate to avoid making attacking fouls, and the multi-agent evaluation result finally outperformed RSA. Interestingly, we observed that single-agent paradigms also adapted by learning to score goals without entering the area, as they cannot coordinate. Still, the worse results obtained with RSA showcase the advantage of cooperating on this environment configuration.

From our experiments, we deduced that simply training a cooperative policy does not guarantee the best performance in the environment. In most cases, a single-agent paradigm proved to be more capable. The possibility of training to control a single robot and using that policy to control all three robots, as shown in the RSA paradigm, makes the single-agent approach especially powerful in this environment. This characteristic also enables us to compare cooperating and non-cooperating policies on equal footing.

From our experiments, we outline the following dilemma of learning cooperating policies for the VSSS problem: simplifications to enable learning can make cooperation unnecessary for optimal control, and adding elements that can make cooperation more advantageous can cause the environment to be too challenging to learn an optimal policy. Additionally, the multi-agent paradigms that can learn cooperation in challenging environments have their learning even more impacted by the increased complexity than the single-agent paradigms.

7.2 CONTRIBUTIONS TO SCIENCE

We list as contributions of this work:

a) Showing that it is possible learning policies that cooperate and score goals in a VSSS environment using the JAL paradigm;

b) A comparison of different RL policy learning paradigms in the VSSS, showing that there is no clear better multi-agent paradigm, JAL and IL performance can vary with the environment characteristics, their capability of learning to cooperate can be advantegeous as we model the environment closer to the full problem, but they should be compared against the single-agent approach;

c) The study of cooperation in the VSSS environment, showing that environment simplifications made to enable learning to score goals can have a negative impact on learning cooperative behaviors.

During the development of this dissertation, we introduced the *rSoccer* framework for the development of RL environments in tasks of the VSSS and Small Size League categories of robot soccer:

- MARTINS, F. B.; MACHADO, M. G.; BASSANI, H. F.; BRAGA, P. H.; BARROS, E. S. rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer. In: Robot World Cup. [S.l.]: Springer, 2021. p. 165–176.

The *rSoccer* environment also served as a basis for this work and others. Below we list publications using the *rSoccer* framework until September of 2023:

a) SILVA, J. d. N. D. d. Evaluation of time limits in reinforcement learning applied to robot soccer simulation. B.S. thesis — Universidade Federal de Pernambuco, 2021.

b) MACHADO, M. G. DyLam: a dynamic reward weighting method for reinforcement learning policy gradient algorithms. Master's Thesi — Universidade Federal de Pernambuco, 2022

c) CRUZ, J. V. S. Exploring reinforcement learning in path planning for omnidirectional robot soccer. B.S. thesis — Universidade Federal de Pernambuco, 2023.

The *rSoccer* environment was also used by RobôCIn to learn an VSSS attacker policy, which won the Latin American Robotics Competition multiple times.

Other works in robotics published during the development of this dissertation:

a) ARAÚJO, V.; **MARTINS, F.**; FERNANDES, R.; BARROS, E. A telemetry-based pi tuning strategy for low-level control of an omnidirectional mobile robot. In: Robot World Cup. [S.l.]: Springer, 2021. p. 189–201.

b) MELO, J. G.; **MARTINS, F.**; CAVALCANTI, L.; FERNANDES, R.; ARAÚJO, V.; JOAQUIM, R.; MONTEIRO, J. G.; BARROS, E. Towards an autonomous robocup small size league robot. In: IEEE. 2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE). [S.l.], 2022. p. 1–6.

7.3 LIMITATIONS

The following limitations of this work are important to mention:

a) does not approach a complete modeling of the VSSS environment;

b) does not study how training through self-play can affect the learning paradigms;

c) does not outline the possible cooperative behaviors and their benefits;

d) does not compare the performance of this work paradigms to the results of the work of BRANDÃO et al. (2022).

7.4 FUTURE WORKS

We believe that an exciting path for future work is to study ways to add the strengths of the paradigms studied in this work:

a) permutations on the experiences obtained by the JAL paradigm so that they are agnostic to the id of the controlled robots, performing a mixture of sharing experiences with data augmentation (DYK; MENG, 2001), hoping to observe the advantages of IL in terms of efficiency of experiences and joint learning of the control of the three robots in JAL;

b) randomize the number of robots controlled during training, both aiming at the control advantages of the single-agent paradigms and preventing the JAL paradigm from controlling only part of the robots;

c) performance gains when using a massively parallel environment make us believe that we can benefit from the symmetries of our environment and permutations to perform a data augmentation to multiply the obtained experiences and thus achieve performance gains;

d) study different approaches to representing observations, such as images and egocentric frame of reference (KLATZKY, 1998);

e) study how the performance of cooperative policies learned in simulation transfer to real robots;

f) creating a set of pre-defined scenarios to isolate the study of specific cooperative behaviors;

g) self-play as a possible solution to the problem of training instability due to the difficulty of the environment, having an environment with opponent complexity adapting to learning.

REFERENCES

AGARWAL, R.; SCHWARZER, M.; CASTRO, P. S.; COURVILLE, A. C.; BELLEMARE, M. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, v. 34, 2021.

ANDRYCHOWICZ, M.; RAICHUK, A.; STAŃCZYK, P.; ORSINI, M.; GIRGIN, S.; MARINIER, R.; HUSSENOT, L.; GEIST, M.; PIETQUIN, O.; MICHALSKI, M.; GELLY, S.; BACHEM, O. What matters for on-policy deep actor-critic methods? a large-scale study. In: *International Conference on Learning Representations*. Virtual Event, Austria: OpenReview.net, 2021. Available at: <https://openreview.net/forum?id=nIAxjsniDzg>.

BAKER, B.; KANITSCHEIDER, I.; MARKOV, T.; WU, Y.; POWELL, G.; MCGREW, B.; MORDATCH, I. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

BASSANI, H. F.; DELGADO, R. A.; JUNIOR, J. N. d. O. L.; MEDEIROS, H. R.; BRAGA, P. H.; MACHADO, M. G.; SANTOS, L. H.; TAPP, A. A framework for studying reinforcement learning and sim-to-real in robot soccer. *arXiv preprint arXiv:2008.12624*, 2020.

BENGIO, Y.; LOURADOUR, J.; COLLOBERT, R.; WESTON, J. Curriculum learning. In: *Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2009. (ICML '09), p. 41–48. ISBN 9781605585161. Available at: <https://doi.org/10.1145/1553374.1553380>.

BERNSTEIN, D. S.; GIVAN, R.; IMMERMAN, N.; ZILBERSTEIN, S. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, INFORMS, v. 27, n. 4, p. 819–840, 2002.

BRANDÃO, B.; LIMA, T. W. D.; SOARES, A.; MELO, L.; MAXIMO, M. R. Multiagent reinforcement learning for strategic decision making and control in robotic soccer through self-play. *IEEE Access*, IEEE, v. 10, p. 72628–72642, 2022.

BROCKMAN, G.; CHEUNG, V.; PETTERSSON, L.; SCHNEIDER, J.; SCHULMAN, J.; TANG, J.; ZAREMBA, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

BURKHARD, H.-D.; DUHAUT, D.; FUJITA, M.; LIMA, P.; MURPHY, R.; ROJAS, R. The road to robocup 2050. *IEEE Robotics & Automation Magazine*, IEEE, v. 9, n. 2, p. 31–38, 2002.

CLEMENTE, F. M.; MARTINS, F. M. L.; MENDES, R. S.; FIGUEIREDO, A. J. A systemic overview of football game: the principles behind the game. *Journal of Human Sport and Exercise*, Universidad de Alicante, v. 9, n. 2, p. 656–667, 2014.

DELGADO, R. d. A. *Aprendizagem de comportamentos em robôs através de aprendizagem por reforço*. Master's Thesis (Master's Thesis) — Universidade Federal de Pernambuco, 2019.

DYK, D. A. V.; MENG, X.-L. The art of data augmentation. *Journal of Computational and Graphical Statistics*, Taylor & Francis, v. 10, n. 1, p. 1–50, 2001.

ENGSTROM, L.; ILYAS, A.; SANTURKAR, S.; TSIPRAS, D.; JANOOS, F.; RUDOLPH, L.; MADRY, A. Implementation matters in deep RL: A case study on PPO and

TRPO. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. Available at: <https://openreview.net/forum?id=r1etN1rtPB>.

FOERSTER, J. N.; FARQUHAR, G.; AFOURAS, T.; NARDELLI, N.; WHITESON, S. Counterfactual multi-agent policy gradients. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. New Orleans, Louisiana, USA: AAAI Press, 2018. (AAAI'18/IAAI'18/EAAI'18). ISBN 978-1-57735-800-8.

HAUSKNECHT, M. J. *Cooperation and communication in multiagent deep reinforcement learning*. Phd Thesis (PhD Thesis) — The University of Texas at Austin, 2016.

HOEN, P. J. t.; TUYLS, K.; PANAIT, L.; LUKE, S.; POUTRÉ, J. A. L. An overview of cooperative and competitive multiagent learning. In: TUYLS, K.; HOEN, P. J.; VERBEECK, K.; SEN, S. (Ed.). *Learning and Adaption in Multi-Agent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 1–46. ISBN 978-3-540-33059-2.

HUANG, S.; DOSSA, R. F. J.; RAFFIN, A.; KANERVISTO, A.; WANG, W. The 37 implementation details of proximal policy optimization. *The ICLR Blog Track 2023*, 2022.

HUANG, S.; DOSSA, R. F. J.; YE, C.; BRAGA, J.; CHAKRABORTY, D.; MEHTA, K.; ARAÚJO, J. G. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *The Journal of Machine Learning Research*, JMLRORG, v. 23, n. 1, p. 12585–12602, 2022.

ITSUKI, N. Soccer server: a simulator for robocup. In: CITESEER. *JSAI AI-Symposium 95: Special Session on RoboCup*. [S.l.], 1995.

KIM, J.-H.; SHIM, H.-S.; JUNG, M.-J.; KIM, H.-S.; VADAKKEPAT, P. Cooperative multi-agent robotic systems: from the robot-soccer perspective. In: *1997 Micro-Robot World Cup Soccer Tournament Proceedings*. [S.l.: s.n.], 1997. p. 3–14.

KIRAN, B. R.; SOBH, I.; TALPAERT, V.; MANNION, P.; SALLAB, A. A. A.; YOGAMANI, S.; PÉREZ, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 23, n. 6, p. 4909–4926, 2021.

KITANO, H.; ASADA, M.; KUNIYOSHI, Y.; NODA, I.; OSAWA, E.; MATSUBARA, H. Robocup: A challenge problem for ai. *AI magazine*, v. 18, n. 1, p. 73–73, 1997.

KLATZKY, R. L. Allocentric and egocentric spatial representations: Definitions, distinctions, and interconnections. In: *Spatial cognition: An interdisciplinary approach to representing and processing spatial knowledge*. [S.l.]: Springer, 1998. p. 1–17.

KURACH, K.; RAICHUK, A.; STAŃCZYK, P.; ZAJĄC, M.; BACHEM, O.; ESPEHOLT, L.; RIQUELME, C.; VINCENT, D.; MICHALSKI, M.; BOUSQUET, O. et al. Google research football: A novel reinforcement learning environment. In: *Proceedings of the AAAI conference on artificial intelligence*. [S.l.: s.n.], 2020. v. 34, n. 04, p. 4501–4510.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *nature*, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

LILLICRAP, T. P.; HUNT, J. J.; PRITZEL, A.; HEESS, N.; EREZ, T.; TASSA, Y.; SILVER, D.; WIERSTRA, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

LITTMAN, M. L. Markov games as a framework for multi-agent reinforcement learning. In: *Machine learning proceedings 1994*. [S.l.]: Elsevier, 1994. p. 157–163.

LIU, S.; LEVER, G.; MEREL, J.; TUNYASUVUNAKOOL, S.; HEESS, N.; GRAEPEL, T. Emergent coordination through competition. *arXiv preprint arXiv:1902.07151*, 2019.

LOWE, R.; WU, Y. I.; TAMAR, A.; HARB, J.; ABBEEL, O. P.; MORDATCH, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, v. 30, 2017.

MAKOVIYCHUK, V.; WAWRZYNIAK, L.; GUO, Y.; LU, M.; STOREY, K.; MACKLIN, M.; HOELLER, D.; RUDIN, N.; ALLSHIRE, A.; HANDA, A. et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

MARTINS, F. B.; MACHADO, M. G.; BASSANI, H. F.; BRAGA, P. H.; BARROS, E. S. rsoccer: A framework for studying reinforcement learning in small and very small size robot soccer. In: *Robot World Cup*. [S.l.]: Springer, 2021. p. 165–176.

MEDEIROS, T. F. de; MÁXIMO, M. R. O. de A.; YONEYAMA, T. Deep reinforcement learning applied to ieee very small size soccer strategy. In: *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. [S.l.: s.n.], 2020. p. 1–6.

MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T.; HARLEY, T.; SILVER, D.; KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. In: PMLR. *International conference on machine learning*. [S.l.], 2016. p. 1928–1937.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G. et al. Human-level control through deep reinforcement learning. *nature*, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015.

MONAJJEMI, V.; KOOCHAKZADEH, A.; GHIDARY, S. S. grsim–robocup small size robot soccer simulator. In: SPRINGER. *RoboCup 2011: Robot Soccer World Cup XV 15*. [S.l.], 2012. p. 450–460.

MORDATCH, I.; ABBEEL, P. Emergence of grounded compositional language in multi-agent populations. In: *Proceedings of the AAAI conference on artificial intelligence*. [S.l.: s.n.], 2018. v. 32, n. 1.

NG, A. Y.; HARADA, D.; RUSSELL, S. Policy invariance under reward transformations: Theory and application to reward shaping. In: CITESEER. *Icml*. [S.l.], 1999. v. 99, p. 278–287.

NGUYEN, T. T.; NGUYEN, N. D.; NAHAVANDI, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, IEEE, v. 50, n. 9, p. 3826–3839, 2020.

OCANA, J. M. C.; RICCIO, F.; CAPOBIANCO, R.; NARDI, D. Cooperative multi-agent deep reinforcement learning in a 2 versus 2 free-kick task. In: SPRINGER. *RoboCup 2019: Robot World Cup XXIII 23*. [S.l.], 2019. p. 44–57.

OPENAI. *GPT-4 Technical Report*. 2023.

PENA, C. H.; MACHADO, M. G.; BARROS, M. S.; SILVA, J. D.; MACIEL, L. D.; REN, T. I.; BARROS, E. N.; BRAGA, P. H.; BASSANI, H. F. An analysis of reinforcement learning applied to coach task in ieee very small size soccer. In: IEEE. *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*. [S.l.], 2020. p. 1–6.

PINTO, A. H. M. *Regras IEEE Very Small Size Soccer (VSSS) - Série A*. 2023. Available at: <https://www.cbrobotica.org/wp-content/uploads/2023/04/regrasVSS23.pdf>.

PUTERMAN, M. L. *Markov decision processes: discrete stochastic dynamic programming*. [S.l.]: John Wiley & Sons, 2014.

RASHID, T.; SAMVELYAN, M.; WITT, C. S. D.; FARQUHAR, G.; FOERSTER, J.; WHITESON, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, JMLRORG, v. 21, n. 1, p. 7234–7284, 2020.

SAMVELYAN, M.; RASHID, T.; WITT, C. S. D.; FARQUHAR, G.; NARDELLI, N.; RUDNER, T. G.; HUNG, C.-M.; TORR, P. H.; FOERSTER, J.; WHITESON, S. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

SCHRITTWIESER, J.; ANTONOGLOU, I.; HUBERT, T.; SIMONYAN, K.; SIFRE, L.; SCHMITT, S.; GUEZ, A.; LOCKHART, E.; HASSABIS, D.; GRAEPEL, T. et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, Nature Publishing Group UK London, v. 588, n. 7839, p. 604–609, 2020.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

SHOHAM, Y.; POWERS, R.; GRENAGER, T. If multi-agent learning is the answer, what is the question? *Artificial intelligence*, Elsevier, v. 171, n. 7, p. 365–377, 2007.

SMITH, L.; KOSTRIKOV, I.; LEVINE, S. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

STONE, P.; VELOSO, M. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, Springer, v. 8, p. 345–383, 2000.

SUKHBAATAR, S.; FERGUS, R. et al. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, v. 29, 2016.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press, 2018.

SUTTON, R. S.; MCALLESTER, D.; SINGH, S.; MANSOUR, Y. Policy gradient methods for reinforcement learning with function approximation. In: SOLLA, S.; LEEN, T.; MüLLER, K. (Ed.). *Advances in Neural Information Processing Systems*. MIT Press, 1999. v. 12. Available at: <https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf>.

TODOROV, E.; EREZ, T.; TASSA, Y. Mujoco: A physics engine for model-based control. In: IEEE. *2012 IEEE/RSJ international conference on intelligent robots and systems*. [S.l.], 2012. p. 5026–5033.

TOUVRON, H.; MARTIN, L.; STONE, K.; ALBERT, P.; ALMAHAIRI, A.; BABAEI, Y.; BASHLYKOV, N.; BATRA, S.; BHARGAVA, P.; BHOSALE, S. et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

TUYLS, K.; WEISS, G. Multiagent learning: Basics, challenges, and prospects. *Ai Magazine*, v. 33, n. 3, p. 41–41, 2012.

UHLENBECK, G. E.; ORNSTEIN, L. S. On the theory of the brownian motion. *Physical review*, APS, v. 36, n. 5, p. 823, 1930.

VINYALS, O.; BABUSCHKIN, I.; CZARNECKI, W. M.; MATHIEU, M.; DUDZIK, A.; CHUNG, J.; CHOI, D. H.; POWELL, R.; EWALDS, T.; GEORGIEV, P. et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, Nature Publishing Group UK London, v. 575, n. 7782, p. 350–354, 2019.

WEISS, G. *Multiagent systems: a modern approach to distributed artificial intelligence*. [S.l.]: MIT press, 1999.

WITT, C. S. de; GUPTA, T.; MAKOVIICHUK, D.; MAKOVIYCHUK, V.; TORR, P. H.; SUN, M.; WHITESON, S. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

WU, T.; ZHOU, P.; LIU, K.; YUAN, Y.; WANG, X.; HUANG, H.; WU, D. O. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Transactions on Vehicular Technology*, IEEE, v. 69, n. 8, p. 8243–8256, 2020.

YANG, Y.; WANG, J. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

YU, C.; VELU, A.; VINITSKY, E.; GAO, J.; WANG, Y.; BAYEN, A.; WU, Y. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, v. 35, p. 24611–24624, 2022.