Hierarchical Reinforcement Learning and Evolution Strategies for Cooperative Robotic Soccer

Conference Paper · November 2024

DOI: 10.1109/ROBOT61475.2024.10797413

5 authors, including:

Gonçalo Leão
University of Porto
13 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Luís Paulo Reis
University of Porto
602 PUBLICATIONS **5,022** CITATIONS
SEE PROFILE

Armando Jorge Sousa
University of Porto
158 PUBLICATIONS **1,071** CITATIONS
SEE PROFILE

This is a pre-print of a contribution published in "Robot 2024: Seventh Iberian Robotics Conference" published by IEEE.

The final authenticated version is avalaible online at: https://doi.org/10.1109/ROBOT61475.2024.10797413

Hierarchical Reinforcement Learning and Evolution Strategies for Cooperative Robotic Soccer

Bárbara Santos[1,2,3], António Cardoso[1,2,3], Gonçalo Leão[1,3], Luís Paulo Reis[1,4], Armando Sousa[1,3]
[1] *FEUP - Faculty of Engineering, University of Porto, Portugal*
[2] *FCUP - Faculty of Sciences, University of Porto, Portugal*
[3] *INESC TEC - INESC Technology and Science, Portugal*
[4] *LIACC - Artificial Intelligence and Computer Science Laboratory, Portugal*
{up202108573}@edu.fc.up.pt,{up202107224,goncalo.leao,lpreis,asousa}@fe.up.pt

***Abstract*—Artificial Intelligence (AI) and Machine Learning are frequently used to develop player skills in robotic soccer scenarios. Despite the potential of deep reinforcement learning, its computational demands pose challenges when learning complex behaviors. This work explores less demanding methods, namely Evolution Strategies (ES) and Hierarchical Reinforcement Learning (HRL), for enhancing coordination and cooperation between two agents from the FC Portugal 3D Simulation Soccer Team, in RoboCup. The goal is for two robots to learn a high-level skill that enables a robot to pass the ball to its teammate as quickly as possible. Results show that the trained models under-performed in a traditional robotic soccer two-agent task and scored perfectly in a much simpler one. Therefore, this work highlights that while these alternative methods can learn trivial cooperative behavior, more complex tasks are difficult to learn.**

***Index Terms*—Evolution Strategies, Hierarchical Reinforcement Learning, Multi-Agent Reinforcement Learning (MARL), Multi-Agent Systems, RoboCup, Robotic soccer, Simulation**

I. Introduction

Robotic soccer promotes the development of Artificial Intelligence and Robotics by applying individual and collective player skills in match contexts and physically demonstrating these actions and tactical planning with real-world robots. To take advantage of the computing power available and recent state-of-the-art Reinforcement Learning algorithms, most of the robot training is done inside simulated environments with precise physics feedback. However, due to the nature of the activity, these environments are dynamic, continuous and with incomplete information at times, making it difficult to train agents effectively in these settings.

While deep reinforcement learning has demonstrated over-achieving success in various domains, its complexity and computational demands often pose challenges, particularly in resource-constrained environments such as robotic soccer simulated environments. Thus, this work aims to explore simpler, traditional reinforcement learning and other machine learning algorithms, within the available environments of the RoboCup simulator. In particular, the objectives defined for this work focus on the equipment of robotic soccer agents with coordination skill sets for effective two-player gameplay by employing Evolution Strategies (ES) and Hierarchical Reinforcement Learning (HRL).

To guide the exploration of non-deep and more traditional learning strategies in robotic soccer, the following research question was posed: *"How can hierarchical and evolution learning strategies be applied to enable robotic soccer agents to coordinate their actions effectively?"*

The primary aim of this study is to implement and evaluate the performance of a variant of an Anchor-based Hierarchical Learning algorithm, and OpenAI's implementation of Evolution Strategies for developing coordinated gameplay skills between two robotic soccer agents within the RoboCup simulator. FC Portugal's codebase[1] was used in order to implement the simulated environments and define all robotic agent's environments, including observations, actions, rewards, and other implementation details [1]. Additionally, these high-level skills were developed using the already existing low-level skills from the codebase, such as walking and kicking.

The remainder of this paper is structured as follows. Section II presents RoboCup, revises the concept of Reinforcement Learning, and overviews the concepts of Hierarchical Reinforcement Learning and Evolution Strategies used in this study. Section III presents related work on RL applied to robotic soccer. Section IV sets the experimental setup for this study, defining the multi-agent simulator and environment, the training algorithms and their parameters, and how the results will be evaluated. Section V discusses the evaluation and metrics scored from the results of the previous section. Lastly, section VI concludes with an overview of the work and lines for future work.

II. Background

A. RoboCup

RoboCup is an international robotics competition founded in 1997, with the goal of promoting robotics and Artificial Intelligence (AI) research. In the context of Multi-Agent Systems, this work serves as a benchmark of cooperative and competitive behaviours, as it is a system with multiple agents with different goals [2]. This event features several different leagues where teams of robots compete in various tasks, with

978-1-5386-5541-2/18/$31.00 ©2018 IEEE

[1] https://github.com/m-abr/FCPCodebase

the most famous being robot soccer. The overarching objective of RoboCup is to advance the state of the art in robotics and AI, with the ambitious long-term goal of developing a team of fully autonomous humanoid robots that can defeat, complying with the official rules of FIFA, the human world champion team by the middle of the 21st century[2].

The RoboCup Soccer competition focuses on robots playing soccer in different leagues, such as the Small Size League, Middle Size League, Standard Platform League, Humanoid League, and Simulation League[3].

B. Reinforcement Learning

Reinforcement Learning (RL) is a type of Machine Learning where an agent learns to make decisions by interacting with its surroundings to maximize a reward. Unlike Supervised Learning, which classifies or predicts data, and Unsupervised Learning, which finds hidden patterns, RL focuses on learning the consequences of actions through trial and error [3].

In RL, the main components are the agent (the decision-maker), the environment (the external system it interacts with), the state and action spaces (what the agent observes and can do), and the reward (feedback guiding it towards optimal behaviour) [4].

The learning process involves a policy (the agent's strategy) and a value function that estimates long-term returns. Various algorithms can be used, such as Q-Learning, a model-free method for learning the value of actions in state [5], Deep Q-Networks, that enhance this algorithm by exploiting deep neural networks to manage high-dimensional state spaces, Policy Gradient Methods, that directly optimize the policy using gradient descent, and Actor-Critic Methods, that combine value function approximation with policy optimization to improve learning efficiency [6], [7].

Balancing exploration (trying new actions) and exploitation (using known actions with high rewards) is crucial, with techniques like ϵ-greedy and Simulated Annealing aiding this trade-off [8].

RL has diverse applications, from mastering games like Chess and Go to autonomous control in robotics, and also optimizing trading strategies in finance, or personalizing treatment planning in healthcare [9].

Challenges in RL include handling vast state and action spaces, ensuring sample efficiency, and the previously mentioned exploration-exploitation trade-off. Deep learning has revolutionized RL by enabling the development of agents capable of efficient performance in complex environments but introduces challenges in Multi-Agent scenarios, such as handling dynamic changes of the environment, partial observations, and continuous state and action spaces [10].

C. Hierarchical Reinforcement Learning

Hierarchical Reinforcement Learning (HRL) branches out from RL by abstracting and splitting a problem into a hierarchy of sub-problems or sub-tasks. Each level of hierarchy in a HRL setting is considered as its own RL problem, with its own sub-goals, such that higher-level tasks call a lower-level child as if they were primitive actions. The advantage of hierarchical decomposition is a reduction in computational complexity if the problem can be represented more compactly and there are reusable sub-tasks. However, the solution of a HRL problem may not lead to an optimal solution to the original RL problem [11].

When a parent-task is formulated as a RL problem, it is commonly formalized as a Semi-Markov Decision Process (SMDP) because its actions are child-tasks that persist for an extended period of time [12] and, in general, these are formulated with real-time valued temporally extended actions [13].

1) Anchor-based HRL: An extension of HRL is Anchor-based Hierarchical Reinforcement Learning (AHRL) [14]. A subgoal-based HRL method uses subgoals to provide intrinsic motivation which helps the agent to reach the desired goal. However, it is hard to define subgoals in complex environments. In this approach, the subgoal is replaced by a concept named *anchor*, which is selected from the achieved goals of the agent. AHRL encourages the agent to quickly move away from the corresponding anchor in the right direction to reach the desired goal.

To explore new states, the agent uses the achieved goal as an anchor and tries to move away from it. The intrinsic reward is computed by the distance between the achieved goal and the corresponding anchor, which is replaced periodically. To prevent random actions and ensure the agent reaches the desired goal, intrinsic rewards are weighted by extrinsic rewards collected during the process. This is followed by a normalization function that adjusts intrinsic rewards so that higher extrinsic rewards lead to greater intrinsic rewards, leading the agent to move faster to the desired goal.

AHRL is built on top of TD3 and is divided into two major parts: collecting transitions and training the critic and actor networks. TD3 is an upgraded version of the DDPG algorithm for continuous control tasks. It fixes the issue of overestimating action values by using two Q-value networks and only updating the policy occasionally, making the learning process more stable and reliable [15].

2) Sub-goal HRL strategy inspired on AHRL: Although AHRL has proven its effectiveness in complex environments, there are some settings that are still hard to work on due to dynamic agents and objects in the environment, for example. The RoboCup simulated environments are very complex for traditional RL methods to adapt the environment's agents to itself. In the same sense, the application of AHRL may have its drawbacks due to the definition of anchor in the context of the RoboCup environment. An anchor is automatically extracted from the agent's achieved goals, which are defined by a subset in the extremely dense observation space of the agent, even if the environment has incomplete information and partial observation.

This work employs a variant of HRL which uses the achieved goals of the agent to define intrinsic reward values,

[2]https://www.robocup.org/objective

[3]https://www.robocup.org/domains/1

without any anchor dependency, followed by the same normalization processing using extrinsic rewards of the agent just as in AHRL. While the anchor would be defined periodically in AHRL throughout an episode, this new approach removes all anchor factors, focusing solely on the optimization of the agent's reward optimization at each portion of the episode in which new anchors would be defined. In other words, this approach focuses on sub-goal optimization. This strategy will be referenced as Sub-goal Hierarchical Reinforcement Learning (SHRL) for the remainder of the article.

D. Evolution Strategies

Evolutionary Strategies (ES) offer a straightforward alternative to traditional optimization techniques like backpropagation used in neural networks. Unlike backpropagation, which requires gradient information and complex computations to update weights, ES uses simple, population-based approaches that are easier to implement and robust, especially when gradient information is hard to obtain.

Biologically, evolution refers to species undergoing changes over generations through natural selection, mutation, and genetic variation, allowing adaptation and survival of the fittest. ES mimics this natural process through an iterative algorithm. Initially, a population of candidate solutions is randomly generated. Each solution's performance is evaluated using a fitness function, and the top performers are selected to form the next generation. Variations are introduced through mutation (random changes) and recombination (combining parts of solutions). This process repeats until a satisfactory solution is found or a set number of iterations is reached.

There are several types of Evolution Strategies, each with their own unique mechanisms for selection and variation. The two primary types are $(\mu + \lambda)$-ES and (μ, λ)-ES. In $(\mu + \lambda)$-ES, the next generation is formed by selecting the best individuals from both the parent (μ) and offspring (λ) populations, ensuring the survival of the fittest. Conversely, in (μ, λ)-ES, only the offspring compete to form the next generation, and "forgetting" the μ previous parents that produced the λ new children. This last approach promotes exploration by constantly introducing new genetic material [16].

Given the conceptual difference and contrast between ES and RL, ES allows the learning process of an agent to be computationally cheaper and more efficient. OpenAI explored the capabilities of ES on continuous and discrete environments as a scalable alternative to RL methods, presenting significant results [17] in environments such as MuJoCo [18], where ES solved one of the hardest control tasks (making a 3D humanoid walk) using 1,400 CPUs across 80 machines in only 10 minutes, and the Atari environment collection where OpenAI trained ES on 720 cores in these environments, reporting that in 1 hour ES achieved comparable performance to A3C [19] trained on 32 cores in 1 day.

III. Related Work

RL has opened the doors to brand new ways of solving tasks with its wide deck of algorithms, each with its own learning strategy to handle environments with certain kinds of constraints. In the same sense, RL has been applied multiple times to Robotic Soccer settings.

Fuzzy Neural Networks (FNNs) have been employed along with RL for Robotic Soccer tasks. The residual algorithm is used to calculate the gradient of the FNN-RL method in order to guarantee the convergence and rapidity of learning. The complex decision-making task is divided into multiple learning sub-tasks that include dynamic role assignment, action selection, and action implementation. These sub-tasks constitute a hierarchical learning system [20].

Since Robotic Soccer usually implies team matches, this renders it a perfect challenging environment for RL in learning both control and strategies within the multi-agent system. Brandão *et al.* [21] applied a self-play RL strategy to produce robotic soccer agents to efficiently score goals in the IEEE Very Small Size challenge, from the Latin American Robotics Competition, achieving a 93% win rate against hand-coded heuristic strategies.

Lastly, within the same simulator presented in this article, work has been developed to produce high-level skill-set-primitives through RL, which have been used in global-scale competitions such as RoboCup 3D Soccer Simulation League. Abreu *et al.* [22] present FCPortugal's training framework, as well as a timeline of skills developed using the skill-set-primitives, which considerably improve the sample efficiency and stability of skills, and motivate seamless transitions. The most recent skill set includes a multi-purpose omnidirectional walk, a dribble with unprecedented ball control, a solid kick, and a few other skills.

IV. Methodological Approach

All the results of this work were obtained with an Intel Core i5-1350P processor, with 4.7GHz and a NVIDIA RTX A500 Laptop GPU. The results were produced using Ubuntu 24.04.4 LTS, and used the packages and versions shown in Table I.

TABLE I
PACKAGES USED IN THIS WORK

Package	Version
Python	3.9.5
gym	0.21.0
numpy	1.21.0
torch	1.8.1
pybullet	3.0.0
mujoco-py	2.1.2.14

A. The Simulator

This work was fully developed on the robotic soccer environment simulator RoboCup, mentioned in Section II-A. The simulator provides a virtual environment where robotic soccer agents can interact with each other and the ball. It leverages a physics engine to simulate the dynamics of the soccer field, ball and robot movements accurately. Moreover, it can be incorporated with a visualisation tool to allow the observation of the robotic agents in the simulated field. In this case, the visualisation tool is Offenburg University's RoboViz[4].

[4]https://github.com/magmaOffenburg/RoboViz

The robots in the simulated environment are equipped with a range of sensors and actuators, to perceive and act upon the world. These include joint actuators for controlling various body parts, such as legs, arms, knees, elbows and head. Additionally, sensors provide information on the robot's localization, orientation, joint angles, and the position of surrounding objects, for example, the ball, goalposts, and other players.

For this study, High-Level Skills will be the focus, such as the coordination between two robots, using the already implemented low-level skills from the FC Portugal Team, which include walking and kicking.

B. The Environment

For this work, two OpenAI Gym environments were implemented to assess the quality and results of the ES and SHRL algorithms when applied to a robotic soccer setting where two agents must perform coordinated skills.

The objective of the first environment was to coordinate a pass between two agents, where *Agent 1* would be kicking to *Agent 2*, and *Agent 2* would be walking towards *Agent 1*. The second environment was designed so that *Agent 1* would pass the ball to *Agent 2* as a corner kick, followed by *Agent 2* kicking the ball towards the goal. For the remainder of the article, the first environment will be referred to as the *PassAndWalk* environment, and the second one as the *CornerKick* environment.

The environments' properties are defined as follows:

- **Initial positions:** At the beginning of each episode of the environments, the agents are randomly positioned within a defined spawning area. In the *PassAndWalk* environment, *Agent 1* is positioned within the $[-3, -1] \times [1, 3]$ region, while *Agent 2* is placed inside the $[4, 6] \times [-4, -2]$ region. In the *CornerKick* environment, *Agent 1* is positioned in the bottom right corner, and *Agent 2* is placed inside the $[6, 10] \times [-2, -2]$ region. The ball always spawns next to *Agent 1*.
- **Observation Space:** The observation space was a concatenation between *Agent 1*'s and *Agent 2*'s observation spaces, being composed of a 1D array of 8 elements with:
  - the ball's 2D position relative to *Agent 1*;
  - *Agent 2*'s 2D position, absolute if *Agent 1* could see it and approximate if not;
  - the ball's 2D position relative to *Agent 2*;
  - *Agent 1*'s 2D position, absolute if *Agent 2* could see it and approximate if not, for the *PassAndWalk* environment and the Goal's 2D position for the *CornerKick* environment.
- **Action Space:** The action spaces of these environments are different, but both represent a concatenation of *Agent 1*'s and *Agent 2*'s action spaces. The *PassAndWalk*'s action space was composed of a 1D array of 5 elements with:
  - the destination's 2D coordinates for the kick;
  - the destination's 2D coordinates for the walk;
  - the absolute orientation of *Agent 2*'s torso, in degrees.

  The *CornerKick*'s action space was composed of a 1D array of 4 elements with:
  - the destination's 2D coordinates for *Agent 1*'s kick to pass the ball to *Agent 2*;
  - the destination's 2D coordinates for *Agent 2*'s kick to score a goal;
- **Reward:** In both setups, the reward is based on the change in the distance between *Agent 2* to the ball. The reward function ($R_1(t)$) was designed to encourage *Agent 2* to minimise the distance to the ball with each step. Specifically, the reward is given by the difference in the distance from the previous step (d_{t-1}) to the current step (d_t), encouraging the agents to reduce it, as shown in Equation 1.

$$R_1(t) = d_{t-1} - d_t \tag{1}$$

  Both environments take advantage of this reward, however, the *CornerKick* environment also implements an additional equivalent reward function ($R_2(t)$) by comparing the difference of the distances between ball and the goal at consecutive time steps (k_{t-1} and k_t), to motivate *Agent 2* to kick towards the goal, as shown in Equation 2. This alternative reward function substituted the prior once *Agent 2* got possession of the ball.

$$R_2(t) = k_{t-1} - k_t \tag{2}$$

  Furthermore, all reward values were multiplied by the maximum steps allowed over the number of steps already run, so that earlier rewards had greater magnitude than later rewards.
- **Neural Network:** In the development of this work, a custom neural network was developed. The primary rationale for this custom design was the necessity for separate learning for each robot agent. Each agent needed to develop specific skills while coordinating effectively with its counterpart, which required a specialised approach to neural network architecture, that separated their action and state spaces.

  The implemented neural network accepts as input the state space of each agent concatenated into one, and comprises two primary sections:
  - The first section processes the first half of the input features through two sequential layers, the first one with 32 units and the second with the same number of units as half of the output dimension (size of the action space for one agent). After each layer, a hyperbolic tangent activation function is applied.
  - The second section mirrors the first but processes the second half of the input features.

  After this, both outputs are concatenated. This dual-path structure allows each agent to independently process its own subset of features. An overview of the system can be visualised in Figure 1.

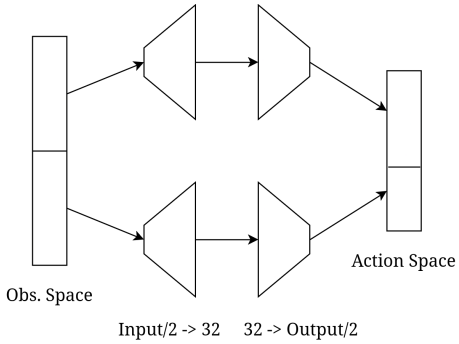


Fig. 1. Diagram of Network Structure

C. The Algorithms

In SHRL, the reward function of each environment was used as a subgoal, which was updated every 10 steps. Other parameters include the exploration noise (2.0), the batch size (128), the discount factor (0.999) and the policy noise (0.1). For both environments, the models were trained for 1000 episodes.

As for ES, the algorithm ran for 25 generations, with 5 offsprings each. Initially, the behaviour-defining parameters were changed in each generation by a Gaussian generated noise with a mean of 0 and a variance of 2. As generations went by, the noise distribution variance was multiplied by 0.999.

V. Experimental Evaluation

A. Evaluation Setup

To obtain test data, a run of 100 episodes was performed with models trained on both training algorithms for each environment. For each episode, experimental data was saved, including the indication of whether the episode was successful, the number of steps in the episode, and both the distance from the ball to *Agent 2* at the end of the episode as well as the distance from the ball to the goal.

B. Success Rate

Firstly, each trained model was assessed on how successful they were at completing each environment's objective.

It is possible to observe in Table II that the ES algorithm had good results on performing successful passes within the *PassAndWalk* environment, with 100% of successful test episodes, while the SHRL algorithm lacked the ability to do so as accurately, with 36% of the total test episodes finishing as desired. However, it is highlighted that both algorithms struggled to learn how to complete the objective of the *CornerKick* environment, given their null success rate. A video with the agents' performance in the *PassAndWalk* environment using ES was made publicly available[5].

The ES-trained agent for the *PassAndWalk* environment performs exactly as intended for this task, as seen in Figure 2. *Agent 1* kicks the ball in the direction of *Agent 2*, while it also runs towards the ball.

[5]https://github.com/GoncaloLeao/Scientific-Research

TABLE II
Percentage of successful episodes for each method

	PassAndWalk	***CornerKick***
SHRL	36%	0%
ES	100%	0%

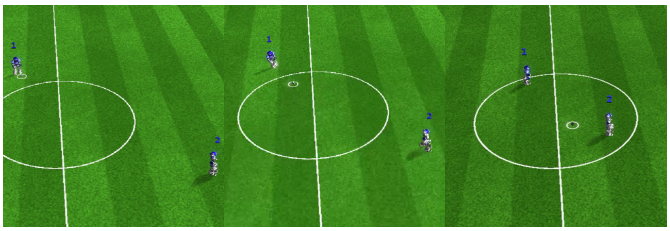
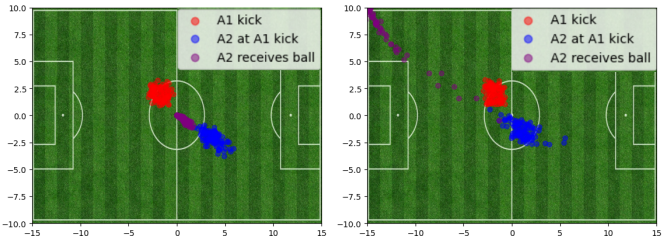


Fig. 2. Frames from an episode run with agents trained with ES in the *PassAndWalk* environment

C. Trajectory Analysis

To further understand the behavior of the agents in all tested settings, positional information and ball-goal distances were used.

Regarding the results for the *PassAndWalk* environment, Figure 3 solidifies the consistent execution from the ES-trained agents. On the other hand, it is possible to conclude that the poor execution from the SHRL models comes from the fact that *Agent 1* did not learn to orient its kick to *Agent 2*, but rather to a corner.



ES SHRL

Fig. 3. Agents' positions at the moment of kick and reception of the ball in the *PassAndWalk* environment

Next, the same position mapping information for the *CornerKick* environment is displayed in Figure 4. It is seen that *Agent 1* is not aiming perfectly at *Agent 2*, aiming mostly to the field's center. In the SHRL case, there are some *Agent 2* receptions that lay inside the penalty area, but the beforehand seen success rate implies that the second agent was not able to score any goal. The histograms present in Figure 4 also allow one to understand the magnitude of the failure of this environment's test episodes. ES was not able to have any goal kick that placed the ball next to the goal closer than 7 units of distance, probably due to *Agent 1*'s inaccurate passes. Given the greater sparsity in ball reception positions, SHRL's *Agent 2* was able to get the ball much closer to the goal in some

situations, however, as seen in the corresponding histogram, it also had situations where the ball must have been in the other side of the field by the end of the episode, indicating the poor ability to aim the kick towards the goal.

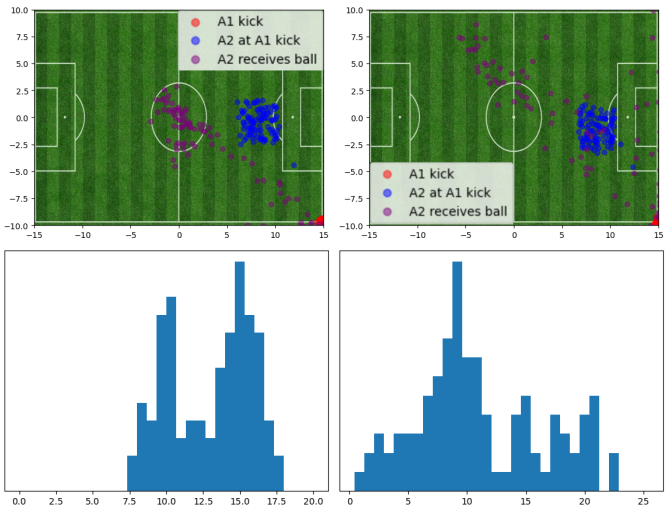


Fig. 4. Agents' positions at the moment of kick and reception of the ball, and ball-goal distances at the end of each episode in the *CornerKick* environment

VI. Conclusions and Future Work

This work aimed to enhance the coordination of two robotic soccer agents in the RoboCup simulator by exploring the Evolution Strategies and Hierarchical Reinforcement Learning algorithms.

While these algorithms are computationally cheaper than DRL ones, their learning and generalization limitations are seen through the registered results in the test settings of this work. On one hand, ES proved to produce a model with a great performance for the agents in the *PassAndWalk* environment, but null in the *CornerKick* one. On the other hand, SHRL was not able to yield any good model for any of the tested environments. The *CornerKick* environment also allowed the authors to verify that the agents performed worse in an environment with more than one task (coordinated pass and directed goal kick).

Keeping in mind that only two experiments were conducted, there may be more tasks where these learning strategies may shine, although RoboCup environments' complexity has been proven to be solved more efficiently with DRL algorithms.

Building on these findings, there are several promising directions for future research and development. Further refinement and study of the ES and HRL algorithms could enhance performance, by experimenting with different parameters, and noise models to optimize learning and coordination, as well as testing the algorithms in other environment settings with different objectives. Other reward functions can also be devised in order to better capture the nuances of multi-agent interactions and further improve learning outcomes.

Acknowledgments

This work was financed within the project UIDB/00027/2020 (DOI 10.54499/UIDB/50014/2020, https://doi.org/10.54499/UIDB/50014/2020) of the Artificial Intelligence and Computer Science Laboratory (LIACC), funded by national funds through the FCT/MCTES (PIDDAC).

References

[1] M. Abreu, P. Mota, L. P. Reis, N. Lau, and M. Florido, "Fc portugal: Robocup 2023 3d simulation league champions," in *RoboCup 2023: Robot World Cup XXVI* (C. Buche, A. Rossi, M. Simões, and U. Visser, eds.), (Cham), pp. 416–427, Springer Nature Switzerland, 2024.

[2] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "Robocup: The robot world cup initiative," in *Proceedings of the first international conference on Autonomous agents*, pp. 340–347, 1997.

[3] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[4] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2016.

[5] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[6] R. S. Sutton, S. Singh, and D. McAllester, "Comparing policy-gradient algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, 2000.

[7] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[8] M. Yogeswaran and S. Ponnambalam, "Reinforcement learning: Exploration–exploitation dilemma in multi-agent foraging task," *Opsearch*, vol. 49, pp. 223–236, 2012.

[9] Y. Li, "Reinforcement learning applications," *arXiv preprint arXiv:1908.06973*, 2019.

[10] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE transactions on cybernetics*, vol. 50, no. 9, pp. 3826–3839, 2020.

[11] B. Hengst, *Hierarchical Reinforcement Learning*. Springer US, 2010.

[12] S. A. Lippman, "Semi-markov decision processes with unbounded rewards," *Management Science*, vol. 19, no. 7, pp. 717–731, 1973.

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

[14] R. Li, Z. Cai, T. Huang, and W. Zhu, "Anchor: The achieved goal to replace the subgoal for hierarchical reinforcement learning," *Knowledge-Based Systems*, vol. 225, p. 107128, 2021.

[15] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *CoRR*, vol. abs/1802.09477, 2018.

[16] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, pp. 3–52, 2002.

[17] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning. arxiv 2017," *arXiv preprint arXiv:1703.03864*, 2017.

[18] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

[19] M. Babaeizadeh, I. Frosio, S. Tyree, J. Clemons, and J. Kautz, "Reinforcement learning through asynchronous advantage actor-critic on a gpu," *arXiv preprint arXiv:1611.06256*, 2016.

[20] Y. Duan, Q. Liu, and X. Xu, "Application of reinforcement learning in robot soccer," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 7, pp. 936–950, 2007.

[21] B. Brandão, T. W. De Lima, A. Soares, L. Melo, and M. R. Maximo, "Multiagent reinforcement learning for strategic decision making and control in robotic soccer through self-play," *IEEE Access*, vol. 10, pp. 72628–72642, 2022.

[22] M. Abreu, L. P. Reis, and N. Lau, "Designing a skilled soccer team for robocup: Exploring skill-set-primitives through reinforcement learning," *arXiv preprint arXiv:2312.14360*, 2023.