Multidiszciplináris tudományok, 13. kötet. 3. sz. (2023), pp. 175-187 **https://doi.org/10.35925/j.multi.2023.3.18**

CURRICULUM LEARNING FOR DEEP REINFORCEMENT LEARNING IN SWARM ROBOTIC NAVIGATION TASK

Alaa Iskandar
PhD student, University of Miskolc, Institute of Mathematics
3515 Miskolc, Miskolc-Egyetemváros, e-mail: iskandar.alaa@student.uni-miskolc.hu

Béla Kovács
associate professor, University of Miskolc, Institute of Mathematics
3515 Miskolc, Miskolc-Egyetemváros, e-mail: bela.kovacs@uni-miskolc.hu

Abstract

This study investigates the training of a swarm consisting of five E-puck robots using Deep reinforcement learning with curriculum learning in a 3D environment. The primary objective is to decompose the navigation task into a curriculum comprising progressively more challenging stages based on curriculum complexity metrics. These metrics encompass swarm size, collision avoidance complexity, and distances between targets and robots. The performance evaluation of the swarm includes key metrics such as success rate, collision rate, training efficiency, and generalization capabilities. To assess their effectiveness, a comparative analysis is conducted between curriculum learning and the proximal policy optimization algorithm. The results demonstrate that curriculum learning outperforms traditional one, yielding higher success rates, improved collision avoidance, and enhanced training efficiency. The trained swarm also exhibits robust generalization for novel scenarios.

Keywords: *swarm robots, navigation task, deep reinforcement learning, currcuilum learning, proximal policy optimization.*

1. Introduction

Swarm robotics (SR) is a field that involves the study of collective behavior exhibited by a group of autonomous, simple, and homogenous robots working together towards a common goal. It draws inspiration from the behavior of natural swarms, such as ant colonies or bird flocks, where individual agents interact locally with their environment and neighboring agents to achieve the required tasks. SR has emerged as a promising approach that leverages the power of teamwork to solve complex problems beyond the capabilities of individual robots. By collaborating in a decentralized manner, SR can tackle tasks such as cleaning large areas, exploring unknown territories, and foraging items from multiple sources. The key features of SR systems include scalability, flexibility, and robustness (Cheraghi et al., 2022).

Designing effective collective behavior in SR systems is a critical aspect of swarm engineering. Two main methods are commonly employed. The behavior-based design method explicitly defines rules for each robot, dictating their actions to achieve the desired collective behavior. On the other hand, the automatic design method utilizes algorithms rooted in artificial intelligence, enabling the robots to

become intelligent entities capable of learning and adapting from their environment (Iskandar et al., 2021)

Among the automatic design methods, two primary approaches have gained significant attention in SR: evolutionary algorithms and reinforcement learning (RL). Evolutionary algorithms, such as Particle Swarm Optimization (PSO) (Gbenga et al., 2016) and Bacterial Foraging Optimization Algorithm (BFOA) (Yang et al., 2014), rely on computational models inspired by natural evolution to optimize the collective behavior of the swarm. However, these approaches often require extensive computational resources and cannot learn in real time.

RL, a field of machine learning, has shown remarkable potential in SR due to its adaptive learning capabilities in complex and dynamic environments. By integrating it with deep learning techniques, SR can continually improve their behavior, enhancing their performance in challenging situations. This combination reduces the computational requirements and enables SR systems to acquire navigation skills and effectively solve navigation tasks autonomously (Shen et al., 2022).

In this paper, we propose the application of curriculum learning, a training strategy inspired by human learning processes, to enhance the reinforcement learning-based navigation capabilities of SR systems. Curriculum learning organizes the learning process by gradually increasing task complexity, allowing the SR to learn fundamental navigation skills before tackling more challenging scenarios. By leveraging curriculum learning in RL, SR systems can achieve improved convergence, generalization, and sample efficiency in navigation tasks.

2. Related work

RL has gained significant attention as a powerful technique for training autonomous agents to learn from their interactions with the environment. Its algorithms allow agents to make sequential decisions in dynamic and uncertain environments, aiming to maximize a cumulative reward signal. RL has been successfully applied to various tasks, including game playing, robotics, and autonomous vehicle control.

2.1. RL approaches and Algorithms

RL is a decision-making approach within the realm of machine learning. It revolves around two key components: the agent and the environment. Through interaction with the environment, the agent learns by sequentially making decisions, exploring the environment, and exploiting the acquired knowledge to solve a given problem.

An RL problem is typically formulated as follows:

- State space (S): The environment is divided into various states. At each time step, the agent perceives the current state S(t) as a frame of by sensors readings and takes action to transition to the next state S(t+1).
- Action space (A): The action space encompasses the set of actions that the agent can take, like turning right or left. Actions can be discrete or continuous, depending on the problem domain.
- Reward function (R): The agent's actions are evaluated using a reward function, which provides feedback based on the observed behavior. Actions that yield desirable outcomes receive positive rewards, while unfavorable actions receive negative rewards. RL aims to learn a policy that maps each state to the optimal action by maximizing the cumulative rewards obtained over time.

In many RL problems, the optimal action may vary depending on the specific situation, and the chosen actions can impact future rewards. To account for these aspects, RL problems are often formulated as Markov Decision Processes (MDPs). MDPs capture the sequential nature of decision-

making, considering the current state, chosen action, and resulting rewards as part of the learning process.

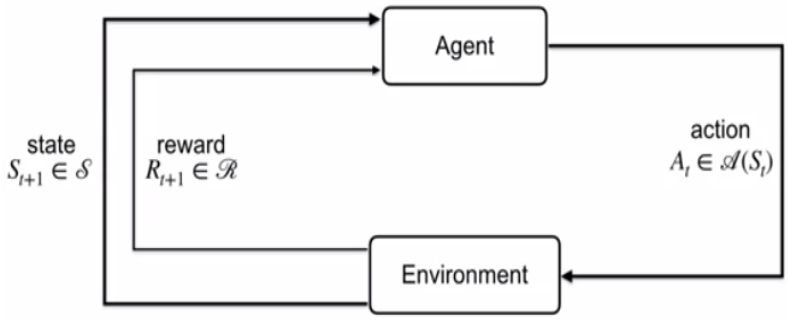


Figure 1. *Representing reinforcement learning approach as Markov decision process.*

The diagram in *Figure* 1 illustrates the RL problem formulated as an MDP, where the agent interacts with the environment, observes the current state, selects an action, receives a reward, and transitions to the next state. This cyclic process continues as the agent aims to learn an optimal policy that maximizes the cumulative rewards received over multiple interactions with the environment.

RL aims to maximize the future reward received each time. The cumulative received rewards are called Return G because of the dynamics of MDP, and it is considered a random variable. It represents an essential value to construct the equations used to find the optimal policy called function values, as in equation 1. There are many approaches to make the agent learn how to make decisions according to Return. It is achieved by computing the function values and choosing the corresponding actions as in equation 2 (Sutton et al., 2018).

$$Q(s,a) = E[G_t|S_t = s, A_t = a] \quad (1)$$

$$a^* = argmax\, Q(s,a); a \in A \quad (2)$$

$Q(s,a)$: State- Action value

$G(t)$: Return at time t, S_t: state at time t, a_t Action at time t

$a*$: Best action

Various approaches are employed in RL, including model-based methods such as Dyna and model-free methods like Q-learning, SARSA, and expected SARSA, among others (Plaat et al., 2023).

Deep learning has introduced novel methods for RL that depart from traditional value-based approaches. Policy-based methods utilize neural networks to directly learn the optimal policy instead of relying on value function estimation. Examples of policy-based methods include Soft Actor-Critic (SAC), Trust Region Policy Optimization (TRPO) (Schulman et al., 2015), and Deep Deterministic Policy Gradient (DDPG) (Casas, 2017). For instance, DDPG is an actor-critic algorithm specifically designed for continuous action spaces. It involves training an actor-network to select actions and a critic network to estimate the value of the selected actions. Proximal Policy Optimization (PPO) is another family of on-policy methods based on actor-critic and policy gradient techniques. PPO optimizes the objective function by employing policy updates with clipped values, ensuring computational efficiency and stability (Schulman et al., 2017).

Figure 2 visually represents the main diagram illustrating RL algorithms, showcasing the interaction between the agent, environment, state transitions, action selection, and learning processes.

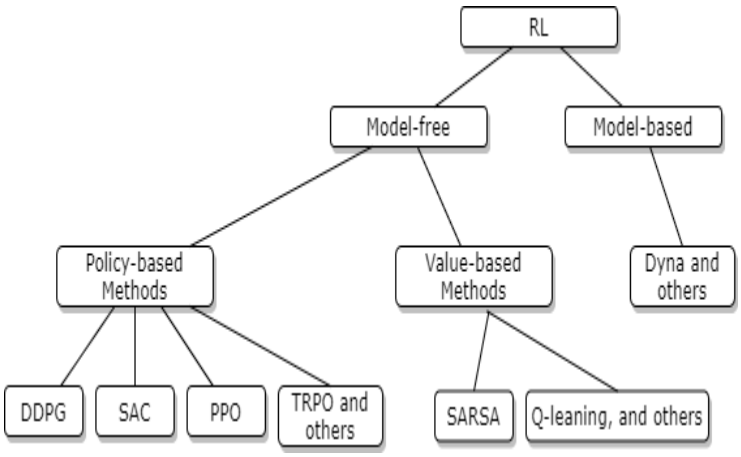


Figure 2. *RL algorithms classification diagram.*

2.2. Deep reinforcement learning in swarm robotic systems

Deep Reinforcement Learning (DRL) has emerged as a prominent approach in SR systems due to its ability to handle complex and dynamic environments and its suitability for multi-agent and swarm scenarios. DRL has been particularly effective in generating collective behaviors in SR systems, such as formation, aggregation, and foraging. For instance, DRL techniques have been employed in foraging tasks to enable robots to learn how to search for food resources and navigate from sources to sinks (Hüttenrauch et al., 2019).

Numerous advancements have been made in adapting and enhancing DRL for SR systems. In one study (Jin et al., 2020), the author compared various methods, including Deep Q-Network (DQN), N-Step-Q Network (NSQ), and Double DQN, to generate collective behavior in a swarm. They also proposed a combined version, Double N-step Q-Network (DNQ), which demonstrated superior performance in learning speed and obtaining optimal policy rewards. Shared memory was utilized to store training data and provide accessibility to all robots. This facilitated faster training processes (Yasuda et al., 2018). Also (Deng et al., 2017) proposed a sequential Q-learning algorithm based on knowledge sharing to address scalability concerns in large swarms. Many researchers have employed value-based methods with sparse rewards, while others have used reward-shaping techniques. The choice of reward function plays a vital role in achieving optimal swarm behavior when modeled as DRL (Wei et al., 2019). The author employed sparse rewards with varying values to encourage obstacle avoidance and goal achievement and concluded that imposing high penalties redirected the robots' focus toward avoiding obstacles rather than reaching the goal.

Furthermore, a hierarchical RL approach with sparse rewards was utilized to solve the problem of food transportation in two steps. This method exhibited greater efficiency and flexibility than traditional RL techniques (Jin et al., 2022).

These prior studies have contributed significantly to the understanding and applying RL in SR navigation tasks. They have explored different algorithms, reward functions, and techniques to enhance collective behavior and optimize swarm performance. Challenges can arise in complex environments or tasks when using randomly initialized network parameters in DRL (Xu et al., 2006).

- Difficulty in Convergence: Random initialization of network parameters means that the initial policy or value function is likely far from optimal. In complex environments, it may take

significant time and experience for the DRL algorithm to converge to a desirable solution. The exploration process is essential to discover effective strategies but can be time-consuming and resource-intensive.

- Deterioration of Performance: Poorly initialized weights and biases can hinder the learning process and prevent the algorithm from effectively adapting to the environment. It may take longer for the agent to learn and converge to a good policy, or it may get stuck in suboptimal solutions. Proper initialization techniques, such as using pre-trained models or transfer learning, can help mitigate this issue.

Curriculum learning is a potential solution to address learning challenges in complex environments. It is an approach inspired by human learning, where concepts are typically introduced in a structured and gradually increasing difficulty manner. In machine learning, curriculum learning involves presenting the learning agent with a sequence of simplified tasks or environments that gradually increase in complexity. The agent can learn basic skills and strategies more easily by starting with simpler tasks or environments. It can gradually build its knowledge and abilities before being exposed to more complex scenarios. This approach helps to overcome the issue of poor initial performance and facilitates a smoother learning process. By exposing the agent to a range of tasks with increasing complexity, curriculum learning promotes robustness and generalization, where the agent learns to adapt its policies to various scenarios and develops a more comprehensive understanding of the environment. Recent advancements in robotics and reinforcement learning (RL), such as indoor path planning for mobile robots, have shown promising results (Gao, 2020). In this context, an incremental training mode for DRL is proposed, starting with a 2D evaluation and then transitioning to 3D environments. The technique is extended to navigate robots within known environments, leveraging the TD3 algorithm for enhanced efficiency and generalization. In the realm of swarm robotics, a novel curriculum-based RL strategy for autonomous shepherding is introduced in (Hussein, 2022). This approach effectively breaks down the shepherding task into sub-tasks, employing a progressive curriculum for improved learning. Achieving a notable 96% success rate, this underscores the value of well-designed curricula in facilitating skill transfer within swarm robotic systems.

In the present study, the Policy-Based PPO algorithm is employed by integrating with curriculum learning and offers advantages such as improved learning efficiency, enhanced exploration-exploitation trade-off, reduced sample complexity, better handling of complex environments, and the acquisition of transferable knowledge. These benefits make curriculum learning a valuable addition to DRL techniques in the context of swarm robotic navigation tasks, leading to more effective and adaptive swarm behavior.

3. Methodology

DRL is used for training a swarm of five E-puck robots to navigate a 3D environment in the Webots simulator and Deepbots framework (Kirtas et al., 2020). The objective is to guide the robots to reach their respective targets while avoiding collisions with each other and obstacles, as shown in *Figure 3*. The problem is formulated as a Markov Decision Process (MDP) represented by a tuple (S, A, T, R, γ).

The state space (S) comprises robot sensor readings, including data from 8 infrared (IR) sensors and the distance between each robot and its target. Each robot has two velocities to control its motion, constituting the action space (A). The transition function (T) governs the system's dynamics, while the reward function (R) provides feedback based on the achieved objectives. Shaping reward, as shown in equation 3, is used. These methods enable the robots to leverage the experience during the path, which increases learning speed by comparing to sparse rewards.

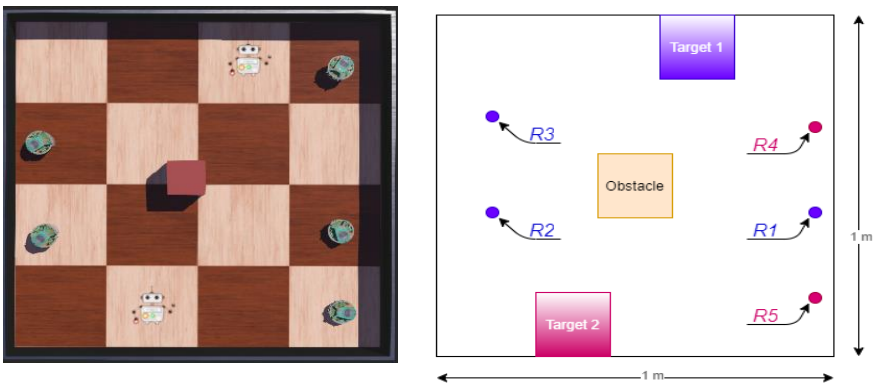


Figure 3. *Swarm's environment*

$$Penality = \begin{cases} -0.001, If\ there\ are\ obstacles\ in\ the\ range\ of\ IR\ sensors. \\ 0, If\ the\ are\ no\ obstacles\ in\ the\ range\ of\ IR\ sensors. \end{cases} \quad (3)$$

$$Reward\ =\ previous\ distance - current\ distance + penality.$$

The discount factor (γ) is a value between 0 and 1 that balances the importance of immediate rewards against future rewards. The primary objective is determining the optimal policy that maps states to the best actions, maximizing the cumulative reward for each state-action pair.

The effectiveness of choosing the value of penalty as in Eq.3 depends on the scaling and context of reward and punishment values. If the difference between the previous and current distance which represents the reward for preferred actions tends to be relatively small, a punishment of -0.001 is appropriate to maintain a balance between encouraging efficiency and avoiding excessive penalties. It prevents the punishment from overwhelming the rewards, but it still incentivizes the agent to improve its performance. Ultimately, the choice of reward and punishment values should be based on experimentation and observation by trying different values and assessing how they impact the agent's learning behavior and performance.

3.1. The agent and DRL algorithm

The PPO algorithm has several advantages that make it well-suited for swarm navigation tasks compared to other DRL algorithms like sample Efficiency: PPO tends to be more sample-efficient compared to other DRL algorithms, such as REINFORCE or DQN, stable Training: PPO incorporates a policy constraint that ensures policy updates remain within a safe region, preventing drastic policy changes, parallelization: PPO can be readily parallelized, allowing multiple robots in the swarm to learn simultaneously. Each robot can have its actor-critic network, collecting experiences and updating its policy independently, and policy gradient methods: PPO falls under the category of policy gradient methods, which directly optimize the policy space instead of value functions.

Each robot's controller has its own DRL algorithm, which learns to find the robot's optimal path by interacting with the other robots and obstacles. A deep neural network is used for nonlinear approximation for value and policy. The neural networks are designed as a series of fully connected layers FC. In PPO, two neural networks are used to train the robot, the policy network, which represents the actor-network, learns to map between states (IR readings and distance) and actions (velocities) with the objective of maximizing received rewards, and the critic network to calculate Q value to enhance

the training of critic network as shown in *Figure 4* and the hyperparameters for in PPO are shown in table 1.

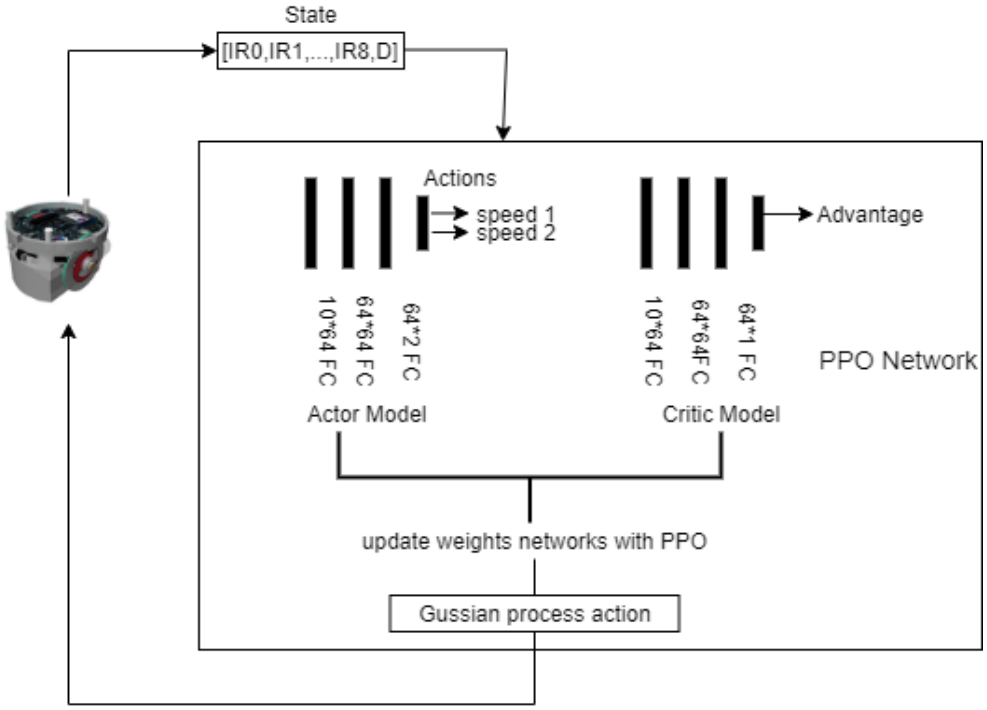


Figure 4. *PPO architecture.*

Table 1. *PPO hyperparameters.*

Parameter	Value
max training timesteps	1000000
Max timesteps per episode	1500
state space dimension	10
action space dimension	2
discount factor (gamma)	0.99
PPO epsilon clip	0.2
PPO K epochs	80
optimizer learning rate actor	0.0003
optimizer learning rate critic	0.001
Initializing a continuous action space policy	
starting std of action distribution	0.6
the decay rate of std of action distribution	0.05
minimum std of action distribution	0.1
decay frequency of std of action distribution	250000 timesteps

For other hyperparameters like PPO epsilon clip it helps to balance exploration and exploitation by limiting policy updates based on the ratio of new and old policies' probabilities, the PPO K epochs refers to the number of times the collected batch of data is used for updating the policy during each iteration, and the learning rate is a crucial hyperparameter that determines the step size by which the optimizer adjusts the weights and biases of the actor and critic networks based on the computed gradients. Choosing an appropriate learning rate is important because it affects how quickly the network's parameters converge to an optimal solution. A learning rate that is too high can cause the optimization process to oscillate or diverge, while a learning rate that is too low can lead to slow convergence or getting stuck in suboptimal solutions.

The hyperparameters related to initializing a continuous action space policy refer to the initial standard deviation of the probability distribution from which the agent's actions are sampled. This distribution is often used to add exploration noise to the actions, encouraging the agent to explore a variety of actions and learn more effectively. All mentiond hyperparameters is choosing based on empirical testing.

3.2. DRL with curriculum learning

To train a swarm of five E-puck robots using DRL with curriculum learning in a 3D environment, the given navigation task must be decomposed into a curriculum of progressively more challenging stages. The decomposition process is built based on curriculum complexity metrics. The metrics that quantify the complexity or challenge associated with different swarm sizes (2 robots, 3robots, and five robots), collision avoidance complexity (the existence of the obstacle or not), and the distances between the targets and robots (by changing the size of the environment from 0.5×0.5 m^2, 0.7×0.7 m^2,0.1×0.1 m^2, and 1.2×1.2 m^2). The following steps can be followed as shown in Figure 5:

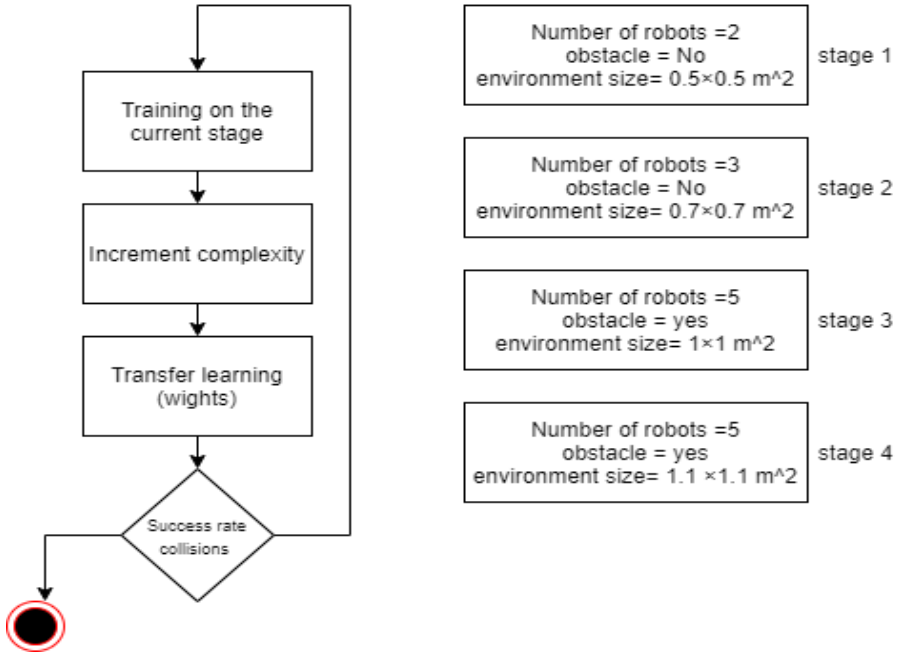


Figure 5. *DRL with curriculum learning*

According to figure 5, the training process is applied by iterating through the training process, gradually increasing the complexity of the stages, and applying transfer learning at each stage. Introduce more challenging scenarios, such as the size of the swarm, obstacles, and increased target distances, to continually improve the swarm's navigation abilities. Assessing the performance of the swarm on each stage by measuring metrics such as success rate (percentage of targets reached), and collision rate.

4. Results

All experiments are conducted on a standard laptop with an I5-8250U@1.6 GHz processor and 6 GB RAM. The parameters of E-puck robots are set as follows: linear velocity [0-0.25] (m/s), angular velocity [-3.14, 3.14] (rad/s), and IR sensor range [0-0.04] (m).

The experiments were conducted in the Webots environment to train the robots in generating collective navigation behavior. The PPO algorithm with curriculum learning and the traditional PPO approach was employed, and both successfully generated the desired collective behavior.

A notable improvement in the learning process can be observed by contrasting Figuers 6-1, 6-2. When applying curriculum learning, the robot exhibits faster learning towards reaching the goal. In the case of the traditional PPO method figure 6-1, approximately 600 episodes are required to achieve successful goal-reaching attempts marked by blue vertical lines, contrasted with failed attempts indicated by white lines. However, with the incorporation of curriculum learning figure 6-2, the robot successfully attains the goal in approximately 300 episodes. The robots trained with curriculum learning exhibited significantly better performance. They were able to explore the environment at a faster pace and effectively utilize the acquired knowledge to reach the designated goals.

One key aspect contributing to the superior performance of curriculum learning is its ability to increase the complexity of the navigation task gradually. By decomposing the task into stages of increasing difficulty, the swarm is exposed to incremental challenges, allowing it to learn and adapt progressively. This enables the swarm to acquire a deeper understanding of the environment and develop more robust navigation strategies. As a result, the robots trained with curriculum learning exhibited a faster convergence towards optimal performance, which is obviously shown in *Figure 7*.

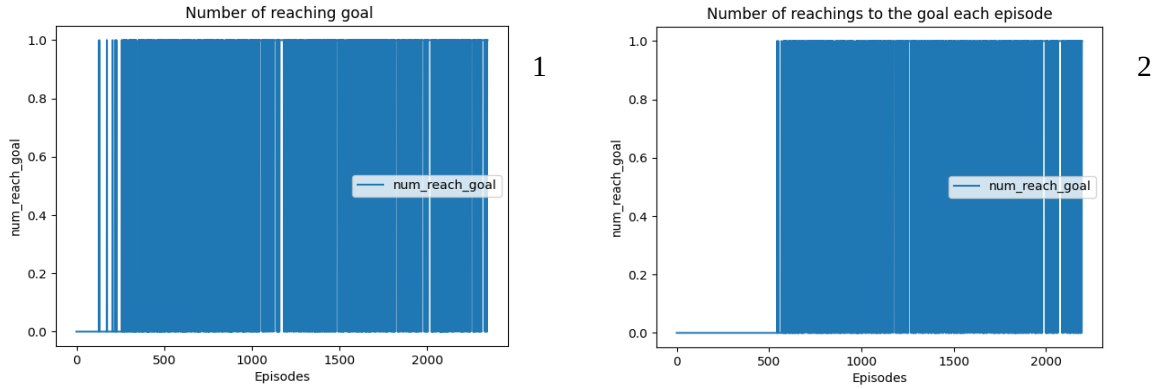


Figure 6. *Success rate 1. PPO with curriculum learning 2. PPO without curriculum learning*

The recorded reward further supports the effectiveness of the curriculum learning approach.

The curves presented in Figure 7 represent the cumulative rewards for one robot. The reward for PPO with curriculum learning reached the optimal value faster than the traditional PPO approach. The improved performance of the curriculum learning approach can be attributed to its ability to guide the swarm through a well-designed sequence of learning stages. The use of an averaging window of 20 values contributes to the smoothness and clarity of the dark plots.

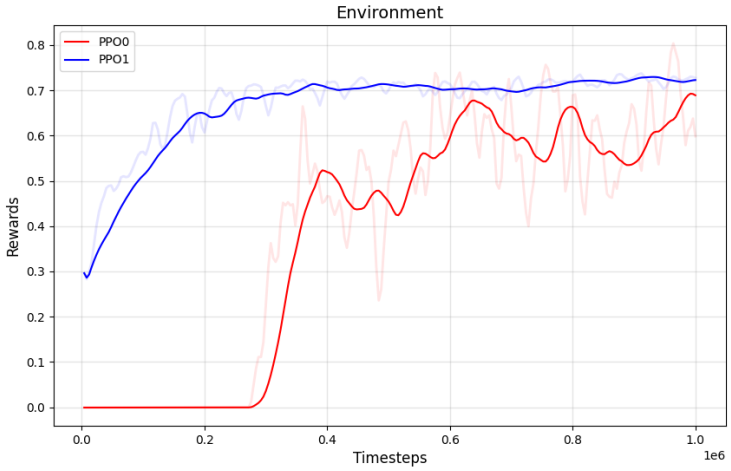
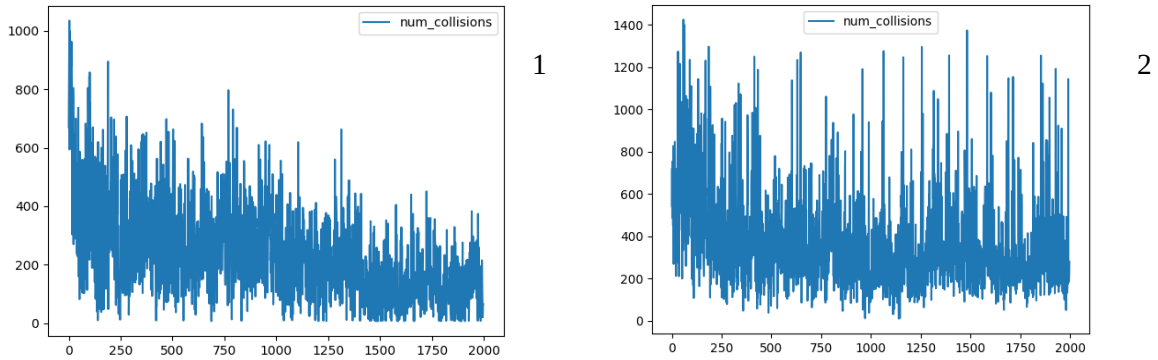


Figure 7. *The average cumulative rewards PP0: without curriculum. PPO1 with curriculum.*

As shown in *Figure 8*, PPO with curriculum learning demonstrates enhanced obstacle avoidance abilities due to its progressive curriculum design. By gradually increasing the complexity of the navigation task, the curriculum learning approach enables the swarm to develop robust obstacle avoidance strategies. The swarm learns to detect and respond to obstacles effectively, navigating them with improved agility and precision. The collisions do not go to zero because of the aggregation of the robots at the goal so each detects others as obstacles.



1 2

Figure 8. *Collison rate 1. PPO with curriculum learning 2. PPO without curriculum learning.*

When the area of the environment is expanded from 1.2×1.2 m^2 to 1.5×1.5 m^2, the performance of a swarm of five robots trained using the PPO with curriculum learning demonstrates more favorable outcomes. As the environment size increases, the curriculum framework guides the swarm in developing

and refining navigation strategies suitable for larger areas. In contrast, the traditional PPO struggles to converge and generate the desired collective behavior, as depicted in *Figure 9*. The robots trained using traditional PPO fail to reach the predefined goals due to the heightened complexity of the larger environment. Their navigation abilities are insufficient to overcome the challenges posed by the increased size. Conversely, the swarm trained with PPO and curriculum learning exhibits promising results. Despite the expanded environment, the curriculum learning framework facilitates the swarm's ability to converge and achieve the required collective behavior.

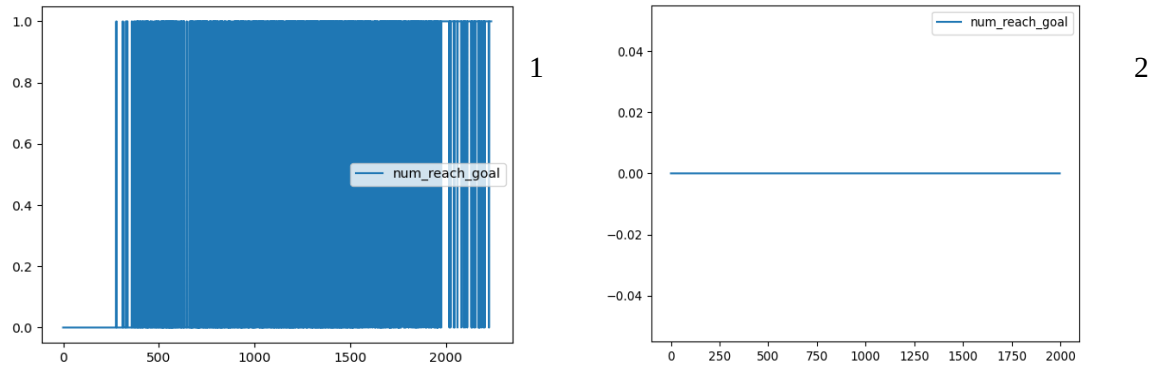


Figure 9. *The success rate for expanded environment 1. PPO with curriculum learning 2. PPO without curriculum learning*

5. Conclusion

In conclusion, this research study has explored the training of a swarm of five E-puck robots using Deep Reinforcement Learning with curriculum learning in a 3D environment. By decomposing the navigation task into a curriculum of progressively more challenging stages, the swarm could acquire robust navigation skills and adapt to complex scenarios.

The curriculum complexity metrics, which considered swarm size, collision avoidance complexity, and distances between targets and robots, guided the curriculum design and facilitated the gradual development of the swarm's abilities. The performance evaluation of the swarm was conducted based on key metrics, including success rate, collision rate, training efficiency, and generalization and adaptation capabilities.

The results demonstrated that PPO with curriculum learning yielded notable advantages over the traditional PPO algorithm. The swarm trained with curriculum learning exhibited a higher success rate in reaching the targets, demonstrating the effectiveness of the curriculum-based approach in guiding the swarm toward desired outcomes. Additionally, the collision rate was effectively mitigated through the acquired obstacle avoidance skills developed during the curriculum learning process. Furthermore, the training efficiency of the swarm was enhanced with curriculum learning, as evidenced by the faster convergence and reaching of optimal values in the cumulative received rewards. The curriculum-based approach enabled the swarm to efficiently explore and exploit knowledge gained in earlier stages, leading to accelerated learning and improved training efficiency.

Importantly, the trained swarm also demonstrated strong generalization and adaptation capabilities. The curriculum learning approach equipped the swarm with the necessary skills to navigate novel

scenarios successfully, showcasing the ability to transfer learned knowledge to previously unseen environments.

Overall, this research provides valuable insights into the application of curriculum learning in training robot swarms and emphasizes its potential for enabling efficient and effective collective behaviors in complex environments. The findings contribute to the advancement of swarm robotics and provide a foundation for future research and development in this field.

References

[1] Cheraghi, A. R., Shahzad, S., & Graffi, K. (2022). Past, present, and future of swarm robotics. In Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 3 (pp. 190-233). Springer International Publishing. **https://doi.org/10.1007/978-3-030-82199-9_13**

[2] Iskandar, A., & Kovács, B. (2021). A survey on automatic design methods for swarm robotics systems. *Carpathian Journal of Electronic & Computer Engineering*, 14(2). **https://doi.org/10.2478/cjece-2021-0006**

[3] Gbenga, D. E., & Ramlan, E. I. (2016). Understanding the limitations of particle swarm algorithm for dynamic optimization tasks: A survey towards the singularity of PSO for swarm robotic applications. *ACM Computing Surveys (CSUR)*, 49(1), 1–25. **https://doi.org/10.1145/2906150**

[4] Yang, B., Ding, Y., & Hao, K. (2014, June). Target searching and trapping for swarm robots with modified bacterial foraging optimization algorithm. In Proceeding of the 11th World Congress on Intelligent Control and Automation, (pp. 1348-1353). IEEE. **https://doi.org/10.1109/wcica.2014.7052915**

[5] Shen, J., Xiao, E., Liu, Y., & Feng, C.: *A deep reinforcement learning environment for particle robot navigation and object manipulation*, 2022 International Conference on Robotics and Automation (ICRA), pp. 6232-6239. IEEE. **https://doi.org/10.1109/icra46639.2022.9811965**

[6] Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

[7] Plaat, A., Kosters, W., & Preuss, M. (2023). High-accuracy model-based reinforcement learning, a survey. *Artificial Intelligence Review*, 1-33. **https://doi.org/10.1007/s10462-022-10335-w**

[8] Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P.: *Trust region policy optimization*, 2015 International Conference on Machine Learning, pp. 1889-1897. PMLR. **https://doi.org/10.48550/arXiv.1502.05477**

[9] Casas, N. (2017). *Deep deterministic policy gradient for urban traffic light control*. arXiv preprint arXiv:1703.09035. **https://doi.org/10.48550/arXiv.1703.09035**

[10] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms*. arXiv preprint arXiv:1707.06347. **https://doi.org/10.48550/arXiv.1707.06347**

[11] Hüttenrauch, M., Adrian, S., & Neumann, G. (2019). Deep reinforcement learning for swarm systems. *Journal of Machine Learning Research*, 20(54), 1–31.

[12] Jin, B., Liang, Y., Han, Z., & Ohkura, K. (2020). Generating collective foraging behavior for robotic swarm using deep reinforcement learning. *Artificial Life and Robotics*, 25, 588–595. **https://doi.org/10.1007/s10015-020-00642-2**

[13] Deng, Z., Guan, H., Huang, R., Liang, H., Zhang, L., & Zhang, J. (2017). Combining model-based q-learning with structural knowledge transfer for robot skill learning. *IEEE Transactions on Cognitive and Developmental Systems*, 11(1), 26–35. **https://doi.org/10.1109/TCDS.2017.2718938**

[14] Wei, Y., Nie, X., Hiraga, M., Ohkura, K., & Car, Z. (2019). Developing end-to-end control policies for robotic swarms using deep Q-learning. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 23(5), 920–927. **https://doi.org/10.20965/jaciii.2019.p0920**

[15] Jin, B., Liang, Y., Han, Z., Hiraga, M., & Ohkura, K. (2022). A hierarchical training method of generating collective foraging behavior for a robotic swarm. *Artificial Life and Robotics*, 1–5. **https://doi.org/10.1007/s10015-021-00714-x**

[16] Gao, J., Ye, W., Guo, J., & Li, Z. (2020). Deep reinforcement learning for indoor mobile robot path planning. *Sensors*, 20(19), 5493. **https://doi.org/10.3390/s20195493**

[17] Hussein, A., Petraki, E., Elsawah, S., & Abbass, H. A. (2022, May). Autonomous swarm shepherding using curriculum-based reinforcement learning. In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (pp. 633-641).

[18] Xu, X. N., & Lai, R. (2006). Present situation and future development of mobile robot path planning technology. *Comput. Simul.*, 10, 1–4.

[19] Kirtas, M., Tsampazis, K., Passalis, N., & Tefas, A.: *Deepbots: A webots-based deep reinforcement learning framework for robotics*, 2020 Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, Proceedings, Part II 16, pp. 64-75. Springer International Publishing. **https://doi.org/10.1007/978-3-030-49186-4_6**