Stand by me: Learning to keep cohesion in the navigation of heterogeneous swarms

Thiago M. Grabe[1], Fabrício R. Inácio[1], Leandro S. Marcolino[2], Douglas G. Macharet[1], Luiz Chaimowicz[1]

[1] Department of Computer Science, Universidade Federal de Minas Gerais, Brazil.
E-mails: {thiago.grabe,fabricio.rod,doug,chaimo}@dcc.ufmg.br
[2] Department of Computer Science, Lancaster University, UK.
E-mail: l.marcolino@lancaster.ac.uk

Abstract. A robotic swarm is a particular type of Multiagent System that employs a large number of simpler agents in order to cooperatively perform different tasks. Oftentimes, the implementation of complex swarm behaviors is a challenging task, and researchers have started to rely on machine learning techniques, which normally require large and complex training setups. In this paper, we explore a segregated navigation task, in which different groups of robots should navigate in a shared environment without mixing with others. Specially, we investigate if a robot trained in simpler scenarios, using a smaller number of robots and groups, can use the learned behavior in more complex scenarios. We performed a series of simulations varying the number of robots and groups and discuss that learning in simpler scenarios can be effective in the segregated navigation task.

Keywords: Multiagent System, Robots, Reinforcement Learning, *ad-hoc Teamwork*

1 Introduction

Robotic swarms are a special type of multi-agent systems that employ a large number of robots to perform complex tasks. Usually, these robots have simple hardware, act based on observations made by sensors capable of perceiving a limited portion of the environment and are able to perform their tasks through interactions with other individuals in the swarm, which lead to the emergence of behaviors that were not explicitly defined.

As can be seen in several research areas, Machine Learning (ML) has become increasingly present in applications using multi-agent systems [10] [12]. Several researchers have been working on solutions for the most diverse types of tasks using ML algorithms. However, when proposing an approach that uses this type of technique, it is necessary to analyze a series of factors that can significantly influence the quality of the developed solution, such as the type of learning to be used, the most appropriate modeling to represent what will be learned by the robots, the tools used to evaluate the quality of what is being learned, etc.

Solving complex problems using robotic swarms controlled by ML algorithms can be an interesting approach, since it is not always simple to define the behavior of each robot during the execution of the task. However, training the swarm in complex scenarios, e.g. with a large number of robots, can make this type of solution unfeasible.

The main objective of this work is to develop a Reinforcement Learning (RL) approach that enables a robot to learn the behavior presented by a group of robots in which it is inserted. More specifically, we investigate if a robot trained in simpler scenarios (with fewer robots and groups) can use the learned policy in more complex scenarios. This is an example of *ad-hoc teamwork* where a group of robots had already been deployed with very well defined task. When adding new robots, without previous knowledge of the environment or the group policy, they are challenged to overcome difficulties and coordinate themselves to achieve the common goal.

To implement the proposed approach, we used as the swarm's desired behavior. This task consists of performing the navigation of the swarm without allowing robots belonging to different groups to mix during the execution of the task [11]. We use Double-DQN, a reinforcement learning algorithm proposed by Van Hassel *et al.* [9], to allow the robot to learn what action to take from the information captured by its sensors.

To evaluate the methodology, we perform a series of simulated experiments varying the number of robots and groups both in training and execution, and discuss the obtained results in terms of scalability and performance.

2 Related Work

An important area of Artificial Intelligence is the development of autonomous robots that can interact with the environment and other different robots to achieve common tasks and goals. In a *ad-hoc teamwork* context, these teammates are supposed to perform as a group without any prior knowledge of how to work together. In this context, Albrecht and Stone presented an important survey about modeling approaches where some methods like *group modeling* and *policy reconstruction* are covered [3]. However, these works are usually presented in a simple grid-world scenarios [1], [16], considering a global view of the environment [5], and, despite few works such as [4], their applicability to real robotic scenarios has not yet been demonstrated [2].

Several works were developed in order to allow an effective and safe navigation of a group of autonomous robots using Machine Learning. Godoy *et al.* [6] propose a solution to the navigation problem in environments with agent congestion that combines reinforcement learning with the ORCA collision control algorithm, where robots learn to adjust the preferred speed. In [7], the authors use Machine Learning and Game Theory to address the problem of congestion in environments with many robots. They propose a RL algorithm that uses a reward function that considers the agent's progress towards its goal and the impact of its behavior on the performance of robots close to him.

In [15], the authors used a deep neural network to allow robots to learn how to choose speeds that guarantee safe navigation by drawing inspiration from the Optimal Reciprocal Collision Avoidance (ORCA) algorithm [18]. A database was generated considering the possible outcomes of the ORCA in different situations, which is later used as input to a deep neural network responsible for calculating the speeds that the robots must perform during their navigation. In a different direction, the same group proposed an approach that uses a set of randomly generated scenarios and divided the training of robots into two parts [14]. They use a decentralized and local view collision avoidance policy for multi-robot systems based upon the Proximal Policy Optimization (PPO) [17].

In this paper, we explore some of the concepts of ad-hoc teamwork in a realistic swarm robotics setting. We investigate the use of Reinforcement Learning to allow a robot to learn and follow the behavior of a group, using mainly local information. Moreover, we show that a robot can be trained in simpler scenarios, and use the learned behavior in more complex ones.

3 Methodology

3.1 Problem Formulation

We consider a scenario in which a swarm, represented as a set $\boldsymbol{\mathcal{S}} = \{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_n\}$ of n holonomic robots, navigates in a 2D environment without obstacles. Each robot $\mathbf{r}_i$ is represented by its pose $q_i = [x_i, y_i]$, with kinematic model given by $\dot{q}_i = u_i$. The entire swarm is formed by m distinct types (groups) of robots, which we represent by the partition $\mathit{\Gamma} = \{\mathit{\Gamma}_1, ..., \mathit{\Gamma}_m\}$, where each $\mathit{\Gamma}_k$ contains all robots of type k and $|\mathit{\Gamma}_k| = n/m$. We assume that $\forall j, k : j \neq k \rightarrow \mathit{\Gamma}_j \cap \mathit{\Gamma}_k = \emptyset$, i.e., each robot is uniquely assigned to a single group. For notation purposes, the robot that is going through the learning process is defined as $\mathbf{r}_l$ where $\{l \in \mathbb{N} \mid 1 \leq l \leq n\}$ and the group to which $\mathbf{r}_l$ belongs is $\mathit{\Gamma}_d = \{d \in \mathbb{N} \mid 1 \leq d \leq m\}$.

We also consider that, at the beginning of the task execution, each group $\mathit{\Gamma}_k \in \mathit{\Gamma}$ is segregated. To check if two groups are segregated, we calculate the average distances between robots from the same group and robots from different groups [13]. Thus, two groups of robots, for example A and B, are considered segregated if the average distance between robots of the same group (group A or group B) is less than the average distance between robots of different groups (that is, between robots in group A and group B). Formally, we have $d_{XX} < d_{XY}$ and $d_{YY} < d_{XY}$ where d_{XY} is the average distance between the robots of groups X and Y:

$$d_{XY} = \frac{1}{|\mathit{\Gamma}_X|} \sum_{i \in \mathit{\Gamma}_X} \left(\frac{1}{|\mathit{\Gamma}_Y|} \sum_{j \in \mathit{\Gamma}_Y} (q_i - q_j) \right). \tag{1}$$

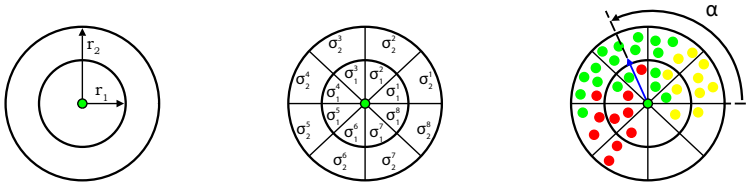
That said, we can define the problem addressed in this paper as follows: *make a robot learn to behave as part of a heterogeneous swarm formed by an arbitrary number of robots that navigate towards a goal in a shared environment while maintaining the condition of segregation between the swarm groups.*

3.2 MDP Modeling

The robot's environment is modelled as Markov Decision Process (*MDP*) [3] represented by a tuple (S, A, P, R, λ) where S is a set of states, A denotes a set of discrete actions, $P : S$ x A x $S \Rightarrow \{x \in \mathbb{R} \mid 0 \leq x \leq 1\}$ is the transition probability function, R is the reward function, and λ is the discount factor applied.

We consider that robots have local perception of the environment. We represent the robot local sensing by a circular area and an external ring, which is divided into q sectors of equal size, as shown in Figure 1(b). Hence, we define region (circular area) $\sigma_1 = \{\sigma_1^1, \sigma_1^2, ..., \sigma_1^q\}$ delimited by radius r_1 and a second region (external ring) $\sigma_2 = \{\sigma_2^1, \sigma_2^2, ..., \sigma_2^q\}$ delimited by radius r_1 and r_2 around the robot. We also consider that the robot knows the group to which it belongs and the goal position.

Any robot or obstacle perceived within a distance r_1 is considered in a "closer zone" by the learning robot. On the other hand, if the distance to another robot or obstacle is farther than r_2, it cannot be detected by the learning robot representing the "external zone". A "mid-term zone" is represented by the external ring where all sensed robots are farther than r_1 and closer than r_2.



(a) (b) (c)

Fig. 1: a) The two sense regions; b) Regions used for state representation; c) An example of the selected action (blue arrow) based on the state configuration.

State Space. We represent the state as a vector $S = \{s_1^1, ..., s_q^1, s_1^2, ..., s_q^2\}$ where s_j^i is the number of robots of the learning robot group (Γ_d) in sector j of region i, subtracted by the number of robots of different groups sensed by $\mathbf{r}_l$.

In Figure 1(c) we can see an example in which robots of distinct groups are represented by different colors, Γ_d is comprised by the green robots and $\mathbf{r}_l$ is the green robot at the center. Note that $\mathbf{r}_l$ is surrounded by robots of three distinct groups (yellow, red and green). The blue arrow represents the potential best action that can be selected by the robot, since the sector corresponding to this action has the largest net amount of individuals of Γ_d.

[3] Despite formally being an *POMDP*, due to the local observations, we abstract the environment representation as a Markov Decision Process (*MDP*) to be able to use the Double-DQN algorithm.

Additionally, we consider a terminal state when the learning robot reaches the goal position $goal_d = [x_d, y_d]$ with its group, which means that all robots from Γ_d, including $\mathbf{r}_l$, have the average distance $\delta_{X,goal}$ to the goal position less than r_2 (Equation 3.2). Other terminal states are reached when $\mathbf{r}_l$ no longer detects any other robot from its group in its sensing area or it collides with another robot in the environment.

$$\delta_{X,goal} = \frac{1}{|\Gamma_X|} \sum_{i \in \Gamma_X} (q_i - goal_X). \tag{2}$$

Action Space The action space is the set of allowed directions in a continuous space, where each action points towards the center of the sectors from regions σ_1 and σ_2. Given an action $\{a \in A \mid 1 \leq a \leq q+1\}$, we calculate an angle α_i towards the respective region. This angle represents the direction of the robot's movement in a global frame, and is given by Equation 3. Here, q represents the number of regions as previously defined and $\alpha_i \in [-\pi, \pi]$.

$$\alpha_i = \frac{\pi}{q} + \frac{a_i \cdot 2\pi}{q} - \pi. \tag{3}$$

We use a constant magnitude w for the performed velocity by $\mathbf{r}_l$. In summary, $u_l = (w \cdot \cos\alpha, w \cdot \sin\alpha)$.

Reward Design The reward has a positive sign if the actions lead the robot to the desired states, and negative of it does not [8]. More specifically, reaching the goal has a $+r$ reward, while losing the group or colliding with other robots have a $-r$ reward.

In addition to setting rewards for the terminal states, we also give rewards for approaching the goal. Basically, we use the number of steps ω to encourage faster arrivals to the goal position. Hence, when the robot is moving away from the goal, we give a reward equals to $-r/\omega$ and when moving closer to the goal, the reward is set to $= +r/\omega$. The value r is a parameter that can be set accordingly.

4 Experiments and Results

4.1 Experimental Setup

As mentioned, our objective is to investigate if we are able to train different policies in simpler scenarios with a smaller number of robots and groups and scale to more complex scenarios with a higher number of robots and groups. Furthermore, we explore the impact of local and global perception in our methodology to exemplify the benefits of using our learning representation in order to keep the robots segregated and reaching the group target.

In order to explore different setups, we use a different number of robots $(5, 10, 15, 20, 25)$ and groups $(1, 2, 3, 4, 8)$ when training. We start each training

with the $\mathbf{r}_l$ placed in a random group Γ_k and randomly selected among the robots of Γ_k.

For instance, in Figure 2 at time step 0, it is possible to observe the initial position of 8 groups with 10 robots each, totalling 80 robots in the simulation. Each group is assigned with a different color for visualization and the black robot is the learning robot $\mathbf{r}_l$, which is part of the green group and was randomly chosen among all robots of its group. The goal position is on the opposite side of the initial position of the groups, and it is marked as a black square in the case of the learning robot's group. At time step 560 in Figure 2, we show the end of the simulation, in which all robots and groups achieved their goals. When $\mathbf{r}_l$ and its group arrived at the goal position we consider it a "Success" and the success rate is the "Success" percentage over all tests runs.

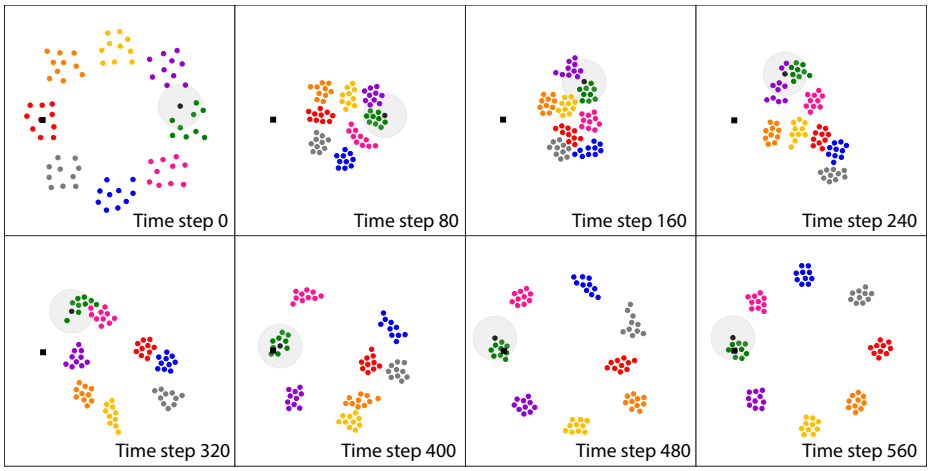


Fig. 2: Simulation step by step. The learning robot $\mathbf{r}_l$ is part of the green group and we represent the group's target as the black square. The shaded area is the area sensed by $\mathbf{r}_l$.

In order to investigate the generality of the learned policies across more complex situations, every combination of number of robots and groups were trained and the learned policy was tested on the same training situation and also increasing the number of robots and groups.

In the graphs, we call *Fixed* the process of training and testing with the same setup (number of robots and groups). When training in a situation and testing in a more complicated setup, we call it an *Extrapolation*. Moreover, we use the success rate over simulations as a metric to compare the results. In order to estimate success rates, we ran 100 training and testing executions, and repeated the whole procedure 30 times, totalling 3000 runs for each combination. We averaged the results and calculated the confidence interval considering a

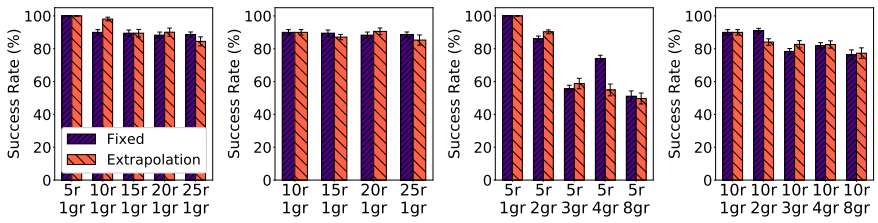
$\rho \leq 0.01$. When we say a result is significant, this means that it is statistically significant considering $\rho \leq 0.01$.

In all figures, "r" shows the number of robots and "gr" the number of groups. As an example, in Figure 3(a) the first two bars (blue and red) present the results for 5 robots and 1 group. Moreover, we can make an analysis over the bars or on each pair of Fixed/Extrapolation bars. When considering all bars in the same chart, we are analysing how increasing the number of robots or groups with a learned policy will affect the success rate comparing all *Extrapolation* bars, and how it affects the success rate when training and testing across these different situations in the case of the *Fixed* bars. On the other hand, comparing the pairs of Fixed/Extrapolation bars, it is possible to understand, in the same setup scenario, how the *Extrapolation* learned policy performs over the *Fixed* policy. This will be used for analysing the results throughout this section.

4.2 Single group training

We start by analysing the setup with one group only during the training and multiple groups during testing. This setup does not have the segregation problem at training time, and the learning robot only senses robots from its group when being trained. Basically, it has to learn to follow its group.

Analysing figures 3(a) and 3(b), "Fixed" and "Extrapolation" have similar results, except for 10 robots and 1 group where "Extrapolation" almost reaches 100% of success rate in Figure 3(a) and 20 robots and 1 group in Figure 3(b). On the other hand, increasing the number of groups (figures 3(c) and (d)) reduces the success rate significantly. Nevertheless, still in some cases the *Extrapolation* is equivalent or significantly better as shown for 5 robots, 2 groups and 5 robots and 3 groups, both in Figure 3(c). Similar results can be observed for 10 robots and 3, 4 and 8 groups in Figure 3(d).

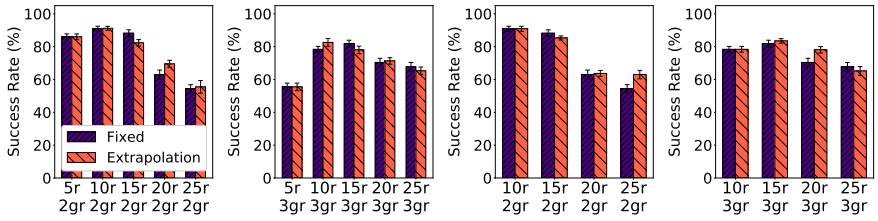


(a) Train: 5r 1gr (b) Train: 10r 1gr (c) Train: 5r 1gr (d) Train: 10r 1gr

Fig. 3: Results for setup with 5 and 10 robots with 1 group in training time for the Extrapolation policy. This scenario does not have the segregation problem and all unsuccessful results is due to lost robot during training. Figures (a) and (b): results upscaling the number of robots. Figures (c) and (d): results upscaling the number of groups.

4.3 Multi-group training

We now run experiments with multiple groups and investigate how increasing the number of robots would affect the success rate. We can observe in Figure 4(a) and Figure 4(b) that increasing the number of robots from 5 robots to 15 and 20 robots has a significantly positive impact since there are more robots of the same group to follow. By contrast, if the number of robots increases more than a certain threshold, the success rate starts to reduce since when there are too many robots in the simulation it is more likely that the learning robot would get captured by another group and then get lost from its own group, as will be discussed in Section 5. Moreover, with more robots in different groups during training time, the exploration becomes crucial to find states where robots from its group are closer. This analysis is also presented in Section 5, where we better discuss the number of explored states.



(a) Train: 5r 2gr (b) Train: 5r 3gr (c) Train: 10r 2gr (d) Train: 10r 3gr

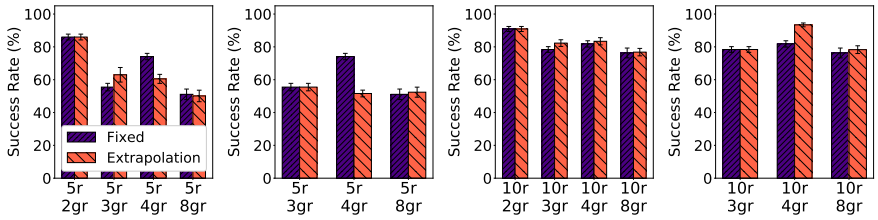
Fig. 4: Results for setup with 5 and 10 robots in training time and upscaling the experiments for more complicated situations increasing the number of robots with local sense.

On the other hand, Figure 4(c) and Figure 4(d) show a different situation. When training with 10 robots and 2 or 3 groups, the success rate decreases or stays similar when increasing the number of robots. Despite of that, an important result is that the majority of *Extrapolation* scenarios are significantly better than the *Fixed*, for example in Figure 4(a) 20 and 25 robots and 2 groups, Figure 4(c) 25 robots and 2 groups and Figure 4(d) 15 and 20 robots and 3 groups. All these examples are surprising results where the *Extrapolation* outperforms *Fixed* scenario. One possible explanation is that when the testing scenario has more robots than the training scenario, the situations in which a robot gets detached from its group is smaller.

4.4 Increasing number of groups

Figure 5 presents the results when upscaling the number of groups in a simulation. Differently from increasing the number of robots, when increasing the number of groups the results get worse or there is no significant impact. In a

scenario when training *Extrapolation* with 5 robots with 2 and 3 groups, Figure 5(a) and 5(b) show that the success rate deteriorates with the increase of groups; while training *Extrapolation* with 10 robots with 2 and 3 groups, Figure 5(c) and (d) show a slight decay. We can also observe that the *Extrapolation* is significantly better than the *Fixed* approach. Overall, the *Extrapolation* success rate is significantly better as shown for 5 robots and 3 groups in Figure 5(a), all cases in Figure 5(c), 10 robots with 4 and 8 groups in Figure 5(d). As mentioned in Section 4.3, this is a surprising result, since we would normally expect a decrease in the success rate. We further discuss this in Section 5.



(a) Train: 5r 2gr (b) Train: 5r 3gr (c) Train: 10r 2gr (d) Train: 10r 3gr

Fig. 5: Results for setup with 5 and 10 robots in training time and upscaling the experiments for more complicated situations increasing the number of groups with local sense.

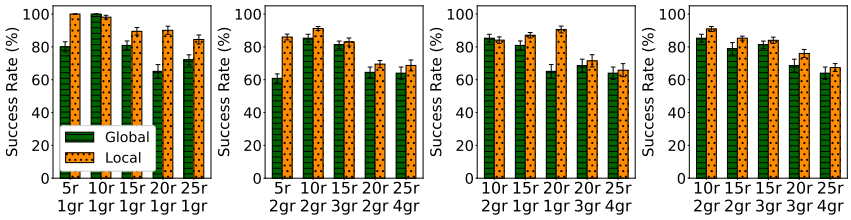
.

4.5 Local Sense vs Global Sense

We also study the impact in using local versus global sensing. Figure 6 shows more complex scenarios (number of robots and groups) from the left bars to the right bars. It is also possible to note that when training in a simpler situation and extrapolating with local sense, the results are significantly better comparing to global sense at the same situation. As an example, looking at Figure 6(a) the bars "20r 1gr" presents the results when training with 5 robots and 1 group (local sense) and testing with 20 robots and 1 group with local (orange bar) and global sense (green bar). The local sensing outperforms the global sensing significantly. In a real world scenario this result is important due to hardware and sensors limitations.

5 Analysis

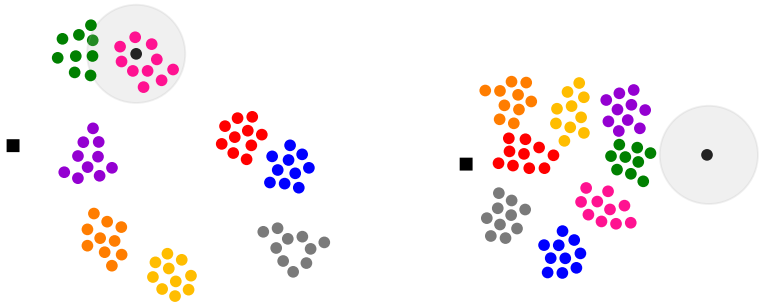
We start our analysis by defining two important concepts: **Captured Robot** is a robot that is surround by other robots that do not belong to its group. Figure 7(a) shows an example where the learning robot from green group is captured by robots from other group (pink). **Lost Robot** is the Robot that can no longer



(a) Train: 5r 1gr (b) Train: 10r 1gr (c) Train: 5r 2gr (d) Train: 10r 2gr

Fig. 6: Results for Global and Local sense. Experiments with 5 and 10 robots, 1 and 2 groups in training time. In test time is used the learned policy to investigate different setups with global and local view.

detect sense robots of its group inside its sensing region. This is exemplified in Figure 7(b).

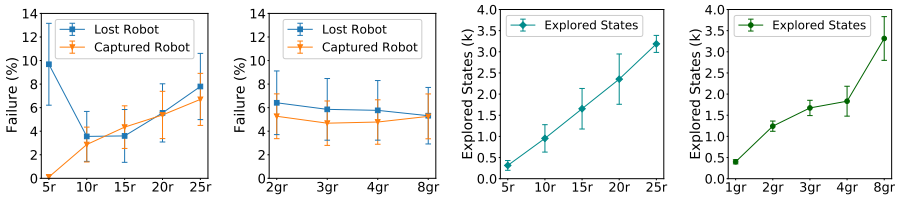


(a) (b)

Fig. 7: Example of (a) learning robot (black) from green group being captured by pink group; (b) learning robot (black) from green group is lost from its group.

Figure 8(a) shows how the failure rate for a "Lost" robot decreases when increasing the number of robots from 5 robots to 10 and 15 robots and then increases again for 25 robots. However, increasing the number of robots also increases the failure rate because of "Captured" robot. As mentioned in Section 4, increasing the number of robots until a certain value has a positive impact since there will be more examples to follow, although too many robots increases the chances of being captured. When looking exclusively to the number of groups presented in Figure 8(b), we can see a steady behavior. This result indicates that the increase in the number of groups does not have a significant impact on "Lost" or "Captured" robot and the failure rate keeps around 6% for all groups.

Concerning the number of explored states, Figure 8(c) and Figure 8(d) shows the explored states with the increase of number of robots and groups respectively. We observe a boost in explored states with the increase in the number of robots and groups. This is an expected result since more robots and groups in training time allow the learning robot to better explore the state space. The exploration is crucial to support the results presented in Section 4.3 for *Fixed* experiments where the increase in the number of robots is significantly positive until a certain point.



(a) Increasing robots (b) Increasing groups (c) Increasing robots (d) Increasing groups

Fig. 8: Analysing the "Lost" and "Captured" failure rate in (a) and (b). Explored states increasing the number of robots and groups in (c) and (d).

6 Conclusion

In this paper, we tackled the use of machine learning techniques to learn behaviors in a complex swarm task. We explored a segregated navigation task, in which different groups of robots navigate in a shared environment without mixing with others. Specially, we investigated if a robot trained in simpler scenarios, using a smaller number of robots and groups, can use the learned behavior in more complex ones.

We performed a series of simulations with the FL-ORCA algorithm in different scenarios. The learned policy demonstrated the capacity to extrapolate its training setup and be applied with a large number of robots and groups. In some cases the extrapolated scenario performs even better than the fixed scenario showing a capacity of generalization without collisions.

As future work, we want to consider more complex environments, with the presence of static and dynamic obstacles. We also intend to explore other RL algorithms and different state representations. Finally, we plan to expand this methodology for Multi-Agent Reinforcement Learning (MARL) deploying more learning robots in the simulation.

References

1. Albrecht, S.V., Ramamoorthy, S.: A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. CoRR (2015). URL http://arxiv.org/abs/1506.01170
2. Albrecht, S.V., Ramamoorthy, S.: Are you doing what I think you are doing? criticising uncertain agent models. CoRR (2019). URL http://arxiv.org/abs/1907.01912
3. Albrecht, S.V., Stone, P.: Autonomous agents modelling other agents: A comprehensive survey and open problems. Artificial Intelligence **258**, 66–95 (2018)
4. Genter, K., Laue, T., Stone, P.: Three years of the robocup standard platform league drop-in player competition. Autonomous Agents and Multi-Agent Systems **31**, 790–820 (2016)
5. Genter, K., Stone, P.: Adding influencing agents to a flock. In: Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-16) (2016)
6. Godoy, J., Karamouzas, I., Guy, S.J., Gini, M.: Online learning for multi-agent local navigation. In: CAVE Workshop at AAMAS. sn (2013)
7. Godoy, J.E., Karamouzas, I., Guy, S.J., Gini, M.: Adaptive learning for multi-agent navigation. In: Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pp. 1577–1585 (2015)
8. Grunitzki, R., da Silva, B.C., Ana Bazzan, L.C.: Towards designing optimal reward functions in multi-agent reinforcement learning problems. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2018)
9. Hasselt, H.v., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16, p. 2094–2100. AAAI Press (2016)
10. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems **33**(6), 750–797 (2019)
11. Inácio, F.R., Macharet, D.G., Chaimowicz, L.: United we move: Decentralized segregated robotic swarm navigation. In: Distributed Autonomous Robotic Systems, Springer Tracts in Advanced Robotics. Springer (2016)
12. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. The International Journal of Robotics Research **32**(11), 1238–1274 (2013)
13. Kumar, M., Garg, D.P., Kumar, V.: Segregation of heterogeneous units in a swarm of robotic agents. Automatic Control, IEEE Transactions on **55**(3), 743–748 (2010)
14. Long, P., Fanl, T., Liao, X., Liu, W., Zhang, H., Pan, J.: Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 6252–6259. IEEE (2018)
15. Long, P., Liu, W., Pan, J.: Deep-learned collision avoidance policy for distributed multiagent navigation. IEEE Robotics and Automation Letters **2**(2), 656–663 (2017)
16. Panella, A., Gmytrasiewicz, P.: Interactive pomdps with finite-state models of other agents. Autonomous Agents and Multi-Agent Systems **31**(4), 861–904 (2017)
17. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR **abs/1707.06347** (2017). URL http://arxiv.org/abs/1707.06347
18. Van Den Berg, J., Guy, S.J., Lin, M., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics research, pp. 3–19. Springer (2011)