# Enhancing Deep Reinforcement Learning Approaches for Multi-Robot Navigation via Single-Robot Evolutionary Policy Search

Enrico Marchesini*, Alessandro Farinelli

***Abstract*— Recent Multi-Agent Deep Reinforcement Learning approaches factorize a global action-value to address non-stationarity and favor cooperation. These methods, however, hinder exploration by introducing constraints (e.g., additive value-decomposition) to guarantee the factorization. Our goal is to enhance exploration and improve sample efficiency of multi-robot mapless navigation by incorporating a periodical Evolutionary Policy Search (EPS). In detail, the multi-agent training "specializes" the robots' policies to learn the collision avoidance skills that are mandatory for the task. Concurrently, in this work we propose the use of Evolutionary Algorithms to explore different regions of the policy space in an environment with only a single robot. The idea is that core navigation skills, originated by the multi-robot policies using mutation operators, improve faster in the single-robot EPS. Hence, policy parameters can be injected into the multi-robot setting using crossovers, leading to improved performance and sample efficiency. Experiments in tasks with up to 12 robots confirm the beneficial transfer of navigation skills from the EPS to the multi-robot setting, improving the performance of prior methods.**

## I. Introduction

Robotic navigation is a key topic in Deep Reinforcement Learning (DRL) literature that ranges from indoor mobile robots [1] to outdoor aquatic platforms [2]. This particular task has also been extended to the Multi-Agent Deep Reinforcement Learning (MARL) domain, due to its wide range of applicability (e.g., search and rescue [3]).

MARL can be formalized in many different ways as, for example, the robots (or agents) can be competitive or cooperative and can optimize an individual or joint reward [4]. A MARL approach can then be expressed under various paradigms, such as centralized or independent learning, or the recently popular *Centralized Training with Decentralized Execution* (CTDE) [5]–[9]. The reason behind the successes of CTDE is because it combines the benefits of previous paradigms, while naturally addressing their limitation. In more detail, *centralized learning* uses a joint observation (i.e., knowledge of all the robots' states) to address the non-stationary nature of multi-agent environments, but hardly scales in the number of robots. In contrast, *independent learning* scales well as it only uses local observations, but does not foster cooperation nor successfully address non-stationarity, due to the environment's changes caused by the other robots. Hence, CTDE centralizes global information for the learning process, decentralizing execution as it only relies on local observations to compute the actions of each robot. Intuitively, this combines the benefits of centralized and independent learning, while being scalable, favoring cooperation, and addressing non-stationarity.

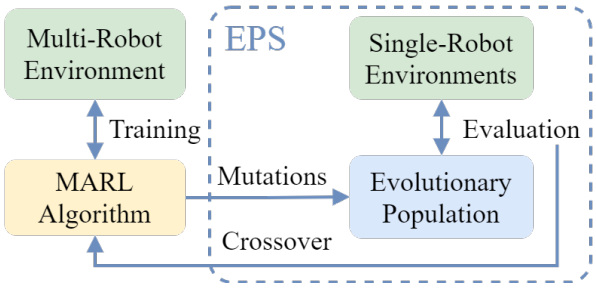


Fig. 1. Overview of our Evolutionary Policy Search.

Given the benefits of CTDE, several value-based MARL approaches have been designed in this direction. In detail, Value-Decomposition Networks (VDN), QMIX, WQMIX, and QTRAN [7]–[10] exploit the idea of factoring a global (or joint) action-value function for the learning process. The underlying issue is that they limit exploration (hence, they are limited in finding better navigation behaviors), due to the constraints (e.g., monotonicity, additivity) used to ensure the factorization. In contrast, a recent trend exploits the insights of dueling networks to maintain a separate estimation of state and advantage values, learning a joint state-value. This allows to avoid constraints [11], addressing the exploration problems of prior approaches (further details on these CTDE approaches are discussed in Section II).

Against this background, we aim at improving exploration for MARL in the problem of multi-robot mapless navigation, by analyzing the problem from a different perspective. Specifically, in this context, a navigation policy can be decoupled into two (so-called) sub-policies: (i) navigation skills to reach the target; (ii) collision avoidance behaviors to avoid other robots. Following relevant literature, both policies could be learned as a single navigation policy [1], [12], however we argue that learning the two skills by training a single policy implicitly hinders sample efficiency and exploration and the motivation is straightforward. A collision between robots ends a training epoch because robots should be returned to a non-collision state (i.e., the environment is typically reset). However, the robots could have continued to explore the current path to explore and acquire novel skills that are useful to learn their end goal of reaching the target.

To this end, we propose an Evolutionary Policy Search (EPS) that works on top of existing MARL approaches. Our

*Contact author: enrico.marchesini@univr.it
Authors are with the Department of Computer Science, University of Verona, 37135 Verona, Italy.

goal is to bias the current multi-robot policies with basic navigation skills that can be easily acquired with EPS in a separate single robot environment. Intuitively, improving navigation skills in a stationary environment with a single robot is much simpler than learning them from scratch in a complex non-stationary environment such as the multi-robot task (and collisions occur rarely, improving the exploration).

In more detail, we propose the use of an Evolutionary population that is periodically generated from the robots' policies using mutation operators to explore different regions of the policy space. These mutated versions of the policies are evaluated independently over a set of trials in a single-robot environment. Hence, we select the individual with the highest return, which means better navigation behaviors, and inject its skills into the MARL policies by using a crossover operator. The beneficial effects of this information transfer have been highlighted by previous combinations of Evolutionary Algorithms and gradient-based DRL [13]–[17]. However, these frameworks were limited to single-agent scenarios and designed to improve the overall return. In contrast, EPS facilitates the learning of basic navigation skills online (i.e., during the MARL training) using our decoupled formalization, to improve both the performance and sample efficiency of existing approaches. Our ablation study in Section IV shows that this improves over the more intuitive solution of pre-training a single-robot navigation policy and use it to initialize the policies of the multi-robot task, similarly to transfer learning [18].

We evaluate EPS on a benchmarking task for multi-agent (i.e., Cooperative navigation) that is part of the *multi-agent particle envs*, a widely used benchmark for DRL methods [19]. Results confirm that our proposed approach improves over prior MARL approaches (i.e., independent learners and GDQ). Hence, we use a TurtleBot3[1]-based environment with up to 12 robots to show the beneficial effects of EPS in multi-robot navigation, highlighting its scalability and performance improvement in increasingly complex scenarios.

## II. Preliminaries and Related Work

We model multi-robot navigation as a Dec-POMDP [20], referring to prior CTDE works for a more formal definition [8]. In this section, we first discuss DRL for robotic navigation and prior MARL approaches and then provide the intuitions behind our CTDE baseline, GDQ [11].

### A. Deep Reinforcement Learning for Robotic Navigation

Navigation has served as a strong benchmark in DRL for robotics [1], [21] due to its practical implications. Recent work [12] showed the benefits of using value-based DRL in this domain, drastically reducing the computational complexity while maintaining comparable performance over policy-gradient (or actor-critic) solutions. Moreover, several recent work renewed the interest in using value-based DRL with discrete action spaces, showing that it can handle high-dimensional domains [22]–[24].

[1]www.turtlebot.com

### B. Centralized Training with Decentralized Execution

Multi-Agent DDPG (MADDPG) [5] has been one of the first MARL algorithms based on the CTDE, where each agent trains using centralized information, but it relies only on the local action-observation history for the decision-making process. Recently, several research efforts shifted to value-based MARL [7]–[10] (this is also due to the insights of the previous section). Such approaches focus on ensuring the so-called *Individual-Global-Max* (IGM), which states that the optimal joint action $\mathbf{a} \coloneqq [a_i]_{i=1}^{n}$ (where $n$ is the number of agents) selected using the global action-value $Q_G(\boldsymbol{\tau}, \mathbf{a})$, must return the same actions that each agent selects in its decentralized execution. Formally, we can factor $Q_G(\boldsymbol{\tau}, \mathbf{a})$ if and only if exists $[Q_i : \mathcal{T}_i \times \mathcal{A}_i \rightarrow \mathbb{R}]_{i=1}^{n}$ (where $\mathcal{T}_i, \mathcal{A}_i$ are the action-observation history space and action spaces of a single agent, respectively) such that $\forall \boldsymbol{\tau} \in \mathcal{T}$:

$$\underset{\mathbf{a} \in \mathcal{A}^n}{\arg\max}\, Q_G(\boldsymbol{\tau}, \mathbf{a}) = \begin{pmatrix} \underset{a_1 \in \mathcal{A}}{\arg\max} Q_1(\tau_1, a_1) \\ \vdots \\ \underset{a_n \in \mathcal{A}}{\arg\max} Q_n(\tau_n, a_n) \end{pmatrix} \quad (1)$$

where $\boldsymbol{\tau}$ is the joint action-observation history, and $Q_i$, $\tau_i$, $a_i$ are the same functions but for each individual $i$.

Prior work propose different structural constraints to ensure such factorization, e.g., *additivity* (VDN [7]), and *monotonicity* (QMIX [8], which have been improved in WQMIX [10]). However, this limits the representation expressiveness of the joint action-value [25]. Along this line, QTRAN [9] relaxes the IGM constraint using a linear soft regularization and linear constraint, but despite the theoretical guarantees it results in poor performance [10].

In contrast, our recent solution called Global Dueling Q-learning (GDQ) [11] avoids using constraints by estimating a joint state-value $V_G(\mathbf{v})$ and exploiting the insights of the Dueling Architecture [26] (where $\mathbf{v} \coloneqq [V_i(\tau_i)]_{i=1}^{n}$ are the individuals' state-values). We use GDQ as the baseline for EPS due to its superior performance over prior work. Moreover, in Section IV we present an additional experiment applying EPS to independent learners, to show its benefits even in non-collaborative architectures.

*1) Global Dueling Q-learning:* we briefly introduce the main ideas behind GDQ. From the Dueling Architecture [26], $Q$-networks maintain two separate streams to estimate the two components that form action-values, i.e., the state-value $V(s)$, and the advantage $A(s, a)$ functions.[2] These streams are then combined in the last network's layer to obtain the actual action-value function $Q(s, a)$. In its simplest form, such aggregation layer computes $Q(s, a) = V(s) + A(s, a)$, while in practice the following equation is used as it results in better performance:

[2]We use a generic notation of state $s$ and action $a$ for our discussion on the Dueling Architecture.

$$Q(s,a) = V(s) + \left( A(s,a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a') \right) \quad (2)$$

where $|\mathcal{A}|$ is the cardinality of the action space. It follows that GDQ maintains a separate network that takes as input the state-values of all the robots and estimates a joint state-value $V_G(\mathbf{v})$. This value replaces the individuals' state-values in their target computation. Hence, introducing global knowledge on the state of the system to all the agents. Crucially, GDQ ensures the IGM without constraints as $V_G(\mathbf{v})$ does not influence the decision-making process of an agent, which depends only on the advantage:

$$\arg\max_a Q(s,a) \equiv \arg\max_a A(s,a) \quad (3)$$

We refer to our GDQ paper [11] for further details about its implementation.

## III. Evolutionary Policy Search

We propose a novel framework that works on top of prior MARL algorithms to improve the learning of core navigation skills (i.e., navigate in a simplified stationary environment) during the training of the multi-robot task.

The general flow of our Evolutionary Policy Search is summarized in Algorithm 1. In more detail, the chosen MARL baseline runs on the multi-robot task following the algorithm's specifications (line 2). Periodically, we leverage Evolutionary Algorithms (EA) [27] and gradient-based mutations [28] to generate a population of mutated versions of the robots' policies (lines 3-6). To this end, we sample a batch $b$ from the memory buffer to compute the per-weight sensitivity $\omega$ of the network's outputs over its weights $\theta^e$ (lines 7-8 and more details in the following section). This is used to generate the population of $m$ individuals $\mathcal{P}$ with weights $\theta_{\mathcal{P}}$, to which we add a copy of the original network's weights (line 9). We evaluate $\mathcal{P}$ in a fixed set of epochs on multiple independent instances of a single-robot environment. This is used to collect the individuals average reward that represents our fitness score $F_{\mathcal{P}}$ (line 10) that is then used to select the best set of weights $\theta^*$, i.e., the one that achieves higher return (line 11):

$$\theta^* = \arg\max_{\theta^e_p} F_{\mathcal{P}} \quad (4)$$

Hence, we inject the best behaviors highlighted in the population in the policies of the multi-robot scenario, using a mean crossover operator based on Polyak averaging (lines 13-14), which showed better performance in our preliminary experiments:

$$\theta^e = \alpha\theta^* + (1-\alpha)\theta^e \quad (5)$$

where $\alpha$ is a hyper-parameter that controls the amount of information that is injected from the best individual and the MARL algorithm. We noticed that high values of $\alpha$ (i.e., $\geq 0.4$) affect the MARL network in a detrimental way, due to a large amount of information on navigating in the stationary single-agent environment that is injected. Conversely, when $\alpha < 0.4$, we found a beneficial transfer of information with a significant performance improvement (discussed in Section IV), which confirms our idea in Section I and the benefits of EPS.

**Algorithm 1** Evolutionary Policy Search

1: **Given:**
- a running MARL algorithm ▷ e.g., GDQ, IQL
- network with shared weights $\theta^e$, at epoch $e$
- periodicity $e_s$ for the EPS
- scale $\sigma$ for the baseline Gaussian mutation $\mathcal{G}$

2: During the training of the multi-robot navigation task, the robots proceed according to the chosen algorithm.
3: **if** $e \ \% \ e_s = 0$ **then**
4: $\mathcal{E} \leftarrow n+1$ instances of single-robot environments
5: $\mathcal{P} \leftarrow m+1$ copies of $\theta_e$ ▷ each with weights $\theta^e_p$
6: Compute $\mathcal{G} \leftarrow \mathcal{N}(0,\sigma)\ \forall$ weight $\in \theta^e_p,\ \forall p \in \mathcal{P}$
7: $b \leftarrow$ Sample a batch of visited states
8: Compute sensitivity $\omega$ as Eq. 6 using $b$
9: $\theta^e_p \leftarrow \theta^e_p + \frac{\mathcal{G}}{\omega}\ ,\ \forall p \in \mathcal{P}$
10: $F_{\mathcal{P}} \leftarrow$ Evaluate$(\mathcal{P}, \mathcal{E})$
11: $\theta^* \leftarrow$ best navigation policy using $F_{\mathcal{P}}$ as Eq. 4
12: **end if**
13: Combine $\theta^e$ with $\theta^*$ using Eq. 5
14: Continue the MARL training until the next EPS

### A. Gradient-based mutations

Perturbing the weights of a DNN via simple Gaussian noise can lead to disruptive policy changes [28]. Hence, we use gradient information to design mutations that avoid such detrimental behaviors, normalizing the Gaussian perturbation by a per-weight sensitivity $\omega$. We consider Gaussian noise $\mathcal{G}$ as a baseline for the perturbations and normalize it with our sensitivity $\omega$, which we compute using past visited states in the memory buffer. We then apply the resultant gradient-based mutations to the population weights. Formally, we use the per-weight magnitude of the gradient of the outputs $\mathbf{y} = f_{\theta^e}(b)$ (where $b$ is a randomly sampled batch of past visited states, and $f_{\theta^e}$ is the function represented by the network with weights $\theta^e$), to estimate the sensitivity $\omega$ to that weight with a first-order approximation:

$$\omega = \sum_{\mathbf{y}} \left( \frac{\sum_{s \in b} abs(\nabla_{\theta^e} f_{\theta^e}(s))}{|b|} \right) \frac{1}{|\mathbf{y}|} \quad (6)$$

where each sample of $b$ contributes equally to $\omega$ as to reduce the overall changes to the policy.[3] We refer to the original work on gradient-based mutations [28] for further details. However, we note that this is the first application of this operator in a MARL context.

[3]We use the absolute value of $\nabla^e_\theta f^e_\theta(s)$ as we are interested in the magnitude (not the sign) of the slope.

### B. Limitations of Evolutionary Policy Search

We note that EPS shares a limitation with prior work [13], [14], requiring a simulator to perform the Evolutionary search. However, state-of-the-art results in DRL, robotics, and combined approaches in general, are mainly achieved using simulation and transferring the policy on real platforms [29], [30]. Moreover, our formalization of EPS in Algorithm 1 requires weight sharing in the considered MARL baseline. While this is typically used in MARL [8], [11], we note that EPS is also compatible with the scenario where each robot maintains its separate set of weights. In this case, it is sufficient to instantiate the population $\mathcal{P}$ with the copies of each robot's set of weights (i.e., the population size equals the number robots $m = n$) and compute the sensitivity separately, using individuals experiences. Finally, given that EPS works on top of existing algorithms, it is applicable with any MARL baseline.

## IV. Experiments

We first investigate the benefits of EPS applied to an independent learners (IL) algorithm based on a state-of-the-art value-based algorithm (i.e., Rainbow [31]) and GDQ [11] that recently showed superior performance over prior value-based MARL. We refer to these implementations as EPS-IL, and EPS-GDQ, respectively. Hence, we introduce the multi-robot navigation environment, where we perform a more comprehensive evaluation of up to 12 robots to confirm whether a concurrent exploration of core navigation behaviors with an evolutionary search can improve the performance of MARL.

Data are collected on a RTX 2070 and an i7-9700k, using the same network architectures and hyper-parameters reported in our prior work [11]. In the navigation tasks, we include the performance of the original QMIX [8] to have a more comprehensive overview of the performance of MARL in multi-robot navigation (note that we considered QMIX [8] and not WQMIX [10] as we obtained comparable performance in our preliminary experiments). We do not consider VDN as it is outperformed by QMIX [8], [11], and QTRAN as it typically results in poor performance [10].[4]

Given the importance of the statistical significance of the evaluation, the following plots and results consider the mean and standard deviation collected over five independent runs with different random seeds. This motivates slightly different results with respect to the original implementations.

For a fair comparison, we include the additional epochs required by EPS in all the results as this is typical in combined approaches [13]. However, we note that the training time overhead is negligible due to parallelization (i.e., each single-robot environment is strictly independent and, in our experiments, EPS-based approaches trains with an overhead of $\approx 4 \pm 3\%$ of the considered MARL baseline).

[4]We use the authors' original implementations for these baselines.

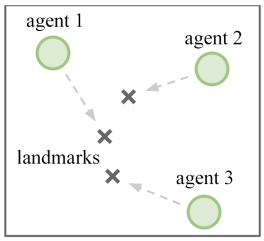


Fig. 2. Overview of the *Cooperative Navigation* scenario.

TABLE I

Avg. reward and collision % of the training phases for IL, EPS-IL, GDQ, EPS-GDQ.

| | Avg. Reward | Avg. Collision (%) |
|---|---|---|
| **IL** | -2.42 ±0.32 | 29.2 ±2.5 |
| **EPS-IL** | -2.16 ±0.25 | 25.1 ±2.6 |
| **GDQ** | -2.12 ±0.24 | 18.2 ±1.5 |
| **EPS-GDQ** | -1.95 ±0.19 | 16.3 ±1.7 |

### A. Benchmark Example

We consider *Cooperative navigation* of the *multiagent particle envs* tasks [19] as a preliminary benchmark to confirm the benefits of EPS. Figure 2 depicts this environment, where 3 agents have to learn how to navigate and cover the landmarks. They receive a positive reward based on how far each agent is from each landmark and a penalty upon collisions.

Results in Table I shows the average reward and percentage of collisions of IL, EPS-IL, GDQ, EPS-GDQ considering the entire training phase (which lasted ≈100 minutes for all the approaches). These preliminary experiments confirm the idea behind EPS, where searching concurrently for better core navigation behaviors leads to better performance (i.e., higher reward and fewer collisions).

### B. Multi-Robot Navigation

For our robotic navigation experiments, we use a similar setup to prior DRL and MARL navigation work [11], [12].

In detail, our multi-robot task models a Turtlebot3 indoor navigation environment with up to 12 robots (our experiments employ $n = \{2, 4, 8, 12\}$ robots), where each robot has to navigate to its target and avoid collisions. The targets are randomly generated in the environment at each epoch, and are guaranteed to be obstacle-free (i.e., they do not spawn on the initial positions of the robots) and separate from each other by at least $0.5m$.

Following the Turtlebot3 specifications, we use an angular velocity of max 90 $deg/s$ and a linear velocity of max $= 0.2$ $m/s$. In every experiment, the decision-making frequency of the robot is 20Hz, to reflect the update rate of the LDS-01 lidar sensor. We use the same specifications for the single-robot environments of EPS. The only other difference in the latter environment is the presence of static obstacles, introduced to maintain key obstacle avoidance behaviors present

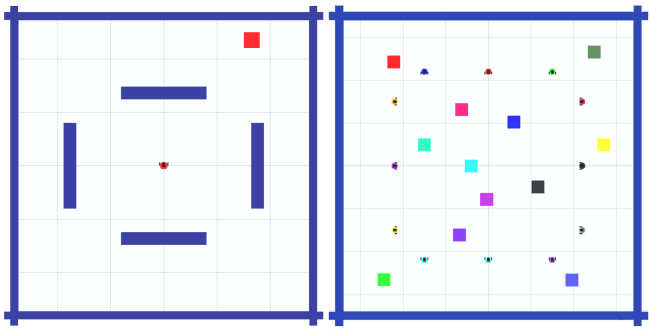

Fig. 3. Left: Single-robot environment for EPS. Obstacles and walls are depicted in blue, while the robot and its target are red. Right: Multi-robot environment for MARL, where each robot-target couple is depicted with a different color.

in the MARL policy. These two environments, depicted in Figure 3 are built with the Unity ml-agents toolkit [32] for three reasons: (i) it has a native Robot Operating System (ROS) interface to deploy the trained models onto the real robots; (ii) it allows to speed up the simulation; (iii) it is straightforward to parallelize the evaluation of EPS treating each single-robot environment independently.

Each robot receives a reward signal $r$ at each time step $t$:

$$r = \begin{cases} -1, \text{upon collision} \\ (d_{t-1} - d_t) - 0.005, \text{ if } d_t > 0.1 \\ 1, \text{ if } d_t <= 0.1 \end{cases} \quad (7)$$

i.e., two sparse values if it reaches the target within distance $d_t = 0.1m$, or colliding with other robots (or the walls), and a dense value when navigating, where $d_{t-1}$, $d_t$ is the euclidean distance between the robot and its goal at two consecutive time steps.

*1) Network Architecture:* We use a 30-dimensional vector with laser scan values sampled in a uniform distribution between $[-120, 120]$ degrees and the robots' target position (defined as the distance from the robot, and the relative heading) as input for the network. The output layer considers the combinatorial space between discrete linear velocities $\in [0.0, 0.7, 0.14, 0.21]$ $m/s$ and angular velocities $\in [-90, -45, 0, 45, 90]$ $deg/s$; this is consistent with prior work [11], [12]. We used the same network's size of our GDQ, to which we refer for further details [11].

*2) Empirical Evaluation:* Following our preliminary experiments on the Cooperative Navigation domain, here we only consider the best-performing algorithms (i.e., GDQ, EPS-GDQ), and QMIX to provide a more comprehensive overview of MARL in multi-robot navigation. Our goal is to investigate whether EPS favors the learning of core navigation skills during MARL training, improving performance and sample-efficiency of the combined approach.

For each experiment with $n = \{2, 4, 8, 12\}$ robots, we plot the following curves that report mean and standard deviation over the multiple runs, smoothed over 100 epochs. In more detail, we plot the average reward as the main evaluation metric, which gives a clear indication of the navigation performance. We also discuss the relative success rate that indicates how many successful collision-free trajectories (i.e., we consider a success when all the robots in the environment reach their target) are performed over 100 epochs.

Figure 4 shows the results of IL, QMIX, GDQ, and EPS-GDQ in the navigation tasks with a growing number of robots, where the goal is to learn how to reach different target positions while avoiding collisions with the walls and with each other.

We note that results are comparable when considering 2 robots, while the benefits of EPS become evident with a growing number of robots. In more detail, with $n = 4$ EPS-GDQ stabilizes at ≈35 average reward (i.e., $\approx 90\%$) in ≈100000 steps. The standard GDQ and QMIX, in contrast, reach ≈88% and ≈90% successes in 175000 and 110000 steps, respectively. The performance improvement of EPS-GDQ is evident with $n = \{8, 12\}$ robots, where QMIX does not learn how to factorize the global action-value in so few interactions with the environment, whereas EPS offer a clear performance advantage over GDQ, especially in the initial phases of the training where core navigation behaviors are learned faster.

*3) Generalization to an Unseen Scenario:* The policies trained with MARL for our multi-robot task can navigate in the environment in a decentralized fashion, exploiting only the local information discussed in the previous sections. To test whether our models generalize over crucial aspects of robotic navigation (e.g., starting and target positions and velocities), given the laser scan-based observations, we perform

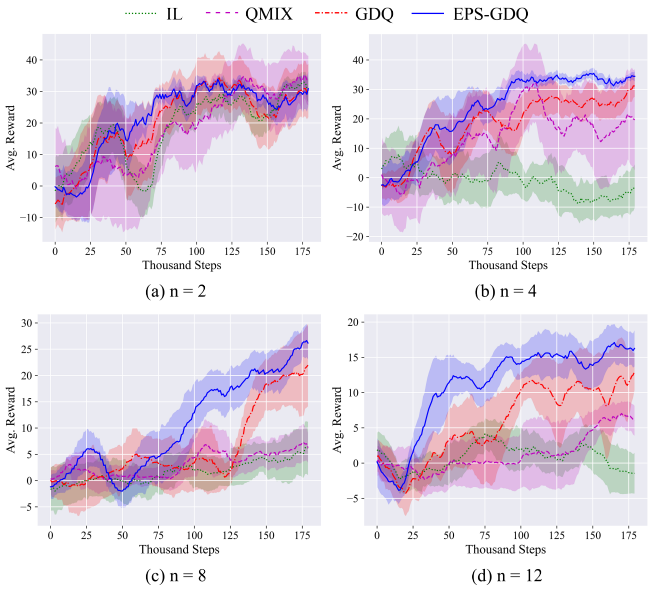


Fig. 4. Average reward for IL, QMIX, GDQ, and EPS-GDQ in our multi-robot navigation scenarios with $n = \{2, 4, 8, 12\}$ robots.

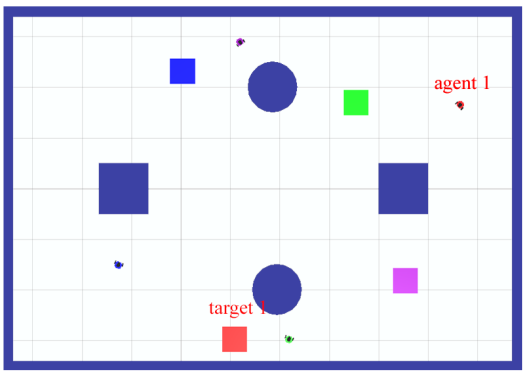


Fig. 5. Overview of the previously unseen scenario.

TABLE II

AVG. REWARD AND COLLISIONS % FOR QMIX, GDQ, EPS-GDQ IN THE EVALUATION IN A PREVIOUSLY UNSEEN SCENARIO.

| | **Avg. Reward** | **Collision (%)** |
|---|---|---|
| **QMIX** | 26.3 ±3.1 | 1.4 ±2.6 |
| **GDQ** | 30.1 ±2.7 | 0.8 ±1.8 |
| **EPS-GDQ** | 32.4 ±2.5 | 0.9 ±1.0 |

an additional experiment with 4 robots in a previously unseen scenario. This testing environment is depicted in Figure 5, where each color characterizes an agent and its target.[5] For a fair evaluation, we selected a sequence of targets that were reachable by all the models. Results in Table II confirm the trend highlighted during the training, where EPS-GDQ outperforms the baseline in terms of average reward. We also measured the percentage of collisions detected, further confirming the performance improvement obtained with our evolutionary search, which returns comparable collisions over GDQ, but with a higher reward.

### C. Ablation Study

To further confirm our idea that EPS improves exploration and sample efficiency of prior algorithms, we performed an additional ablation study in the scenario with 4 robots. In detail, we initially train a navigation policy in the single-robot scenario (pre-training). Hence, we initialize the network's weights of GDQ with the resultant set of weights of the pre-training (which we refer to as Pre-GDQ), in a similar fashion to transfer learning. Our goal is to verify whether the EPS periodical search offers improved performance over the more intuitive solution of Pre-GDQ.

We performed three experiments with different initialization at different stages of the pre-training (i.e., at around 50, 75, and 95% of success rate for the single-robot pre training, which corresponds to $\approx 12, 20, 30$ average reward, respectively). Figure 6 shows the average reward over different runs for Pre-GDQ initialized at ≈75% successes, while Table III shows the performance of our trials with different

[5]We consider 4 robots due to the good performance of the models trained with the MARL baselines.


Fig. 6. Avg. reward for the single-agent pre training (left) and Pre-GDQ (right) initialized with a single-robot policy that reached ≈20 avg. reward. Pre-GDQ reaches ≈30 avg. reward in around 210000 steps.

TABLE III

AVERAGE RESULTS WITH DIFFERENT PRE TRAINING FOR GDQ. EPS-GDQ OUTPERFORM PRE-GDQ IN ANY INIZIALITATION, ACHIEVING ≈35 AVG. REWARD IN ≈100000 STEPS

| **Pre Training** | | **Pre-GDQ** | | **EPS-GDQ** | |
|---|---|---|---|---|---|
| Steps | Reward | Steps | Reward | Steps | Reward |
| 75 | 12 | 200 | 25 | | |
| 110 | 20 | 210 | 30 | 100 | 35 |
| 170 | 30 | 280 | 34 | | |

initializations (where we indicate the steps in thousands). Crucially, every experiment confirms our intuition behind EPS as our EPS-GDQ achieves ≈35 average reward in 100000 steps (i.e., approximately two times fewer steps than the best performing Pre-GDQ).

## V. DISCUSSION

We presented EPS, a novel approach for multi-robot navigation that works on top of existing algorithms, maintaining their CTDE fashion. The idea is to perform a periodical evolutionary search to find better core navigation behaviors to inject into the MARL training. Our preliminary evaluation in Cooperative Navigation highlighted the performance improvement when using EPS with prior MARL algorithms (i.e., IL, GDQ). Hence, we evaluated our framework in a multi-robot robotic navigation scenario with up to 12 robots. This empirical evaluation shows that EPS-GDQ significantly improves the performance of prior MARL approaches, especially with a growing number of robots. This work paves the way for several research directions as the application of EAs to multi-agent scenarios is a novel way to foster desired behaviors into the training loop, possibly enabling the exploration of safer behaviors to address Safe MARL. Another interesting direction follows the recent trend of using Formal Verification [33], [34] to quantify and foster safety behaviors of the agents, which we believe is a crucial aspect in complex multi-agent tasks.

## REFERENCES


[1] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real drl: Continuous control of mobile robots for mapless navigation," in *IROS*, 2017.

[2] E. Marchesini, D. Corsi, and A. Farinelli, "Benchmarking aquatic navigation using deep reinforcement learning and formal verification," in *IROS*, 2021.

[3] J. L. Baxter, E. K. Burke, J. M. Garibaldi, and M. Norman, "Multi-robot search and rescue: A potential field based approach," in *Autonomous Robots and Agents*, 2007.

[4] K. Tuyls and G. Weiss, "Multiagent learning: Basics, challenges, and prospects," in *AI Magazine*, 2012.

[5] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *NIPS*, 2017.

[6] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *AAAI*, 2018.

[7] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning," in *AAMAS*, 2018.

[8] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. N. Foerster, and S. Whiteson, "QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning," in *ICML*, 2018.

[9] K. Son, D. Kim, W. J. Kang, D. Hostallero, and Y. Yi, "QTRAN: learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *ICML*, 2019.

[10] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *NeurIPS*, 2020.

[11] E. Marchesini and A. Farinelli, "Centralizing state values in dueling architectures for multi-agent reinforcement learning navigation," in *IROS*, 2021.

[12] ——, "Discrete deep reinforcement learning for mapless navigation," in *ICRA*, 2020.

[13] S. Khadka and K. Tumer, "Evolution-guided policy gradient in reinforcement learning," in *NeurIPS*, 2018.

[14] S. Pourchot, "CEM-RL: Combining evolutionary and gradient-based methods for policy search," in *ICLR*, 2019.

[15] E. Marchesini and A. Farinelli, "Genetic deep reinforcement learning for mapless navigation," in *AAMAS*, 2020.

[16] E. Marchesini, D. Corsi, and A. Farinelli, "Genetic soft updates for policy evolution in deep reinforcement learning," in *ICLR*, 2021.

[17] ——, "Exploring safer behaviors for deep reinforcement learning," in *AAAI*, 2022.

[18] "Learning for a robot: Deep reinforcement learning, imitation learning, transfer learning," *Sensors*, 2021.

[19] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *arXiv*, 2017.

[20] F. A. Oliehoek and C. Amato, "A concise introduction to decentralized pomdps," in *SpringerBriefs in Intellingece Systems*, 2016.

[21] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *IROS*, 2017.

[22] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," in *AAAI*, 2018.

[23] T. Wiele, D. Warde-Farley, A. Mnih, and V. Mnih, "Q-learning in enormous action spaces via amortized approximate maximization," in *NeurIPS Workshop*, 2018.

[24] G. Dulac-Arnold, R. Evans, P. Sunehag, and B. Coppin, "Reinforcement learning in large discrete action spaces," in *CoRR*, 2015.

[25] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson, "MAVEN: multi-agent variational exploration," in *NeuIPS*, 2019.

[26] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," in *ICML*, 2016.

[27] D. Fogel, "Toward a new philosophy of machine intelligence," in *Evolutionary computation 3. ed.*, 2006.

[28] J. Lehman, J. Chen, J. Clune, and K. O. Stanley, "Safe mutations for deep and recurrent neural networks through output gradients," in *GECCO*, 2018.

[29] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *IEEE Symposium Series on Computational Intelligence*, 2020.

[30] Z. Ding, N. F. Lepora, and E. Johns, "Sim-to-real transfer for optical tactile sensing," in *ICRA*, 2020.

[31] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in drl," in *AAAI*, 2018.

[32] A. Juliani, V. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A platform for intelligent agents," in *CoRR*, 2018.

[33] D. Corsi, E. Marchesini, and A. Farinelli, "Formal verification of neural networks for safety-critical tasks in deep reinforcement learning," in *UAI*, 2021.

[34] D. Corsi, E. Marchesini, A. Farinelli, and P. Fiorini, "Formal verification for safe deep reinforcement learning in trajectory generation," in *IRC*, 2020.