Learning a Roster of Policies for Pareto-Optimal Coordination

Raghav Thakar
Oregon State University
Corvallis, OR
thakarr@oregonstate.edu

Siddarth Iyer
Oregon State University
Corvallis, OR
viswansi@oregonstate.edu

Abstract

Environmental monitoring, search and rescue operations, and providing support in healthcare settings are some promising applications of multiagent systems. In many such problems, a team may have multiple, possibly conflicting objectives. To provide choice in the trade-offs among objectives, multiagent teams must learn a range of behaviors. However, traditional multiagent learning methods may not be directly applied to these problems without scalarizing utility functions. Scalarization may require expertise, and fundamentally changes the problem. Multi-objective evolution can address this problem and can learn a Pareto front of team policies, but suffers from sample inefficiency. Gradient-based methods are sample efficient, but ill-suited in sparse reward settings. In this work, we propose a Multiagent Roster for Multi-Objective Teams (MARMOT) that learns a 'roster' of policies. Teams then formed from this roster express a wide range of Pareto-optimal trade-offs. MARMOT uses multi-objective evolution to evolve a roster on the sparse team reward, while an off-policy reinforcement learning component trains agents on a dense *local* reward. Our experiments on the Multi-Objective Rover Exploration Problem domain show boost-ups in Pareto front quality compared to classical NSGA-II.

1 Introduction

Cooperative multiagent systems have found several practical applications, ranging from environmental monitoring Athanasiadis and Mitkas [2004], Cristani et al. [2020], search and rescue operations Wu et al. [2023], to even healthcare Liu et al. [2017]. In many such problems, the system goals are often better captured by multiple, often conflicting objectives. For instance, in a multiagent environmental monitoring scenario, drones may be required to spread out and evenly search the area, simultaneously focus on specific points of interest, and also preserve battery life. In such settings, it is desirable to learn a range of team behaviors, each expressing a different *trade-off* among the conflicting objectives.

Optimizing multiple objectives is challenging without scalarizing *utility* functions that combine the objectives into one. Scalarization enforces rigid preferences over objectives, usually requires substantial domain expertise, and fundamentally changes the optimization problem. Without scalarization, traditional multiagent learning methods may not be directly applied to multi-objective problems.

Multi-objective evolutionary algorithms (MOEAs), such as NSGA-II, are a promising utility-free learning approach. These methods enable learning joint policies on the *Pareto front* – the ideal learning outcome. MOEAs are also effective when feedback from the environment is sparse, due to their indifference to the availability of rewards at each timestep. In the gradient-based paradigm, MO-MIX is the only Multiagent Reinforcement Learning (MARL) method that enables learning a Pareto front of coordinated multi-objective joint policies Hu et al. [2023].

Preprint.

On one hand, while MOEAs are utility-free, they suffer from sample inefficiency, and may take a considerable number of evaluations before convergence. On the other hand, gradient-based methods like MO-MIX are sample-efficient, but require rich, time-step level feedback, making them ill-suited when rewards are sparse or infrequent. MO-MIX, particularly, is also only designed for discrete actions spaces. Thus, no existing multiagent methods enable 1) utility-free, and 2) sample-efficient learning of Pareto-optimal joint policies under sparse rewards.

In this work, we propose a Multiagent Roster for Multi-Objective Teams (MARMOT). MARMOT learns a *roster* of policies, such that teams formed from this roster express various trade-offs. We use the Multiagent Evolutionary Reinforcement Learning (MERL) framework, combining the sample efficiency of gradient-based methods with the utility-free nature and suitability for sparse rewards of evolutionary approaches. An NSGA-II-like MOEA evolves a roster of coordinated agents, while an off-policy RL component efficiently trains individual agents to perform basic skills.

Our key insight is that a roster combines the combinatorial advantage of forming teams from a population of policies, with the *synergy* of an individually trained team, to produce high-performing yet diverse multiagent teams.The diverse teams that MARMOT is able to generate express a range of Pareto-optimal trade-offs.

In many settings, while dense multi-objective team-level rewards may be unavailable, agents may have access to a dense *local* reward, often emanating from task-specific feedback. MARMOT, leveraging the MERL paradigm, assumes this reward structure; enabling sample-efficient Pareto-optimal coordination among the agents.

The main contribution of this paper is MARMOT, a split-level multi-objective multiagent learning framework that produces several, diverse Pareto-optimal team policies. We test MARMOT in the continuous Multi-Objective Rover Exploration Problem domain. Our experiments show considerable boost-ups in Pareto fronts learned compared to classical NSGA-II, and show the efficacy of leveraging rosters for multi-objective coordination problems.

2 Background and Related Works

Multi-Objective Optimisation

Several optimisation problems cannot be accurately modelled as single-objective problems. In many cases, there may be multiple objectives that require *simultaneous optimisation*. We call such problems *multi-objective* problems, and their optimisation, *multi-objective optimisation*. A key property of such problems is the presence of objectives that *conflict* with one another. Among conflicting objectives of a problem, one objective can generally only be improved upon at the cost of one or more other objectives. This creates a complex scenario where, although the overall goal is to optimise all objectives simultaneously, there exists no single optimiser solution that can do so. Thus, in multi-objective optimisation literature, we seek to control the *trade-offs* that are made by solutions, so as to achieve distinct solutions that each optimise the objectives differently. For instance, if solution A optimises *each* objective better than solution B, then A is said to *dominate* B. This means A will be preferred between the two regardless of preferences over the objectives. However, if there are two solutions such that solution A is better across some objectives, while solution C is better on the remaining, then no clear preference between A and C can be asserted. Therefore both solutions are considered *equally optimal*. Among a pool (or population) of all solutions, the set of solutions that are not dominated by any other solution(s), is called the *non-dominated*, or *Pareto-optimal* set. Each solution in the Pareto-optimal set is *equally optimal*.

Utility-Based and Utility-Free Multi-Objective Optimization

Functions that compose a 'super' objective by collapsing multiple objectives into one are called *utility* functions in the literature Rădulescu et al. [2020]. In multiagent settings, these utilities may be shared by all agents in the system, or computed and used independently by each agent. In either case, designing utility functions assumes significant prior knowledge of the domain to accurately assert compound preferences over the objectives. In many cases, agents may not have predefined preferences over objectives, or these utilities may simply not be known beforehand. Utility-based optimization is also rigid to evolving or dynamic preferences over the objectives. Therefore, it is often preferable to operate in the *utility-free* paradigm instead, and learn a set of candidate solutions

where each solution optimizes the objectives differently – presenting a Decision Maker with rich options of trade-offs to choose from.

Multi-Objective Evolutionary Algorithms

Multi-Objective Evolutionary Algorithms (MOEAs) are a popular choice in the literature for solving multi-objective optimization problems. As these methods operate on a *population* of candidate solutions, MOEAs are well-suited for achieving multiple, equally optimal solutions on the Pareto front. Of the methods proposed in the literature, the Nondominated Sorting Genetic Algorithm-II (NSGA-II) method Deb et al. [2002] is the most popular. NSGA-II utilizes the nondominated-level of candidate solutions (how many other solutions they are dominated by) along with their spatial 'uniqueness' in the objective space (captured via the crowding distance metric) to guide the evolution process. A highly desirable quality of NSGA-II (and most MOEAs) is the ability to learn a *Pareto front* of candidate solutions without requiring a Decision Maker to scalarize the multi-objective reward vector into a single scalar value a priori. Thus, this places NSGA-II in the utility-free paradigm.

Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) is a Policy Gradient Reinforcement Learning method to learn deterministic policies in domains with continuous high-dimensional action spaces Lillicrap [2015]. DDPG is an off-policy method that employs an actor-critic architecture Barto [2021] with one deterministic policy (actor) $\pi : \mathcal{S} \mapsto \mathcal{A}$ and a corresponding critic $\mathcal{Q} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$. The critic learns to accurately map state-action pairs to their values by minimizing the loss derived from the Temporal Difference error between target and predicted values. The actor attempts to maximize the predicted values for its experiences as estimated by the critic. Separate copies of the critic and actor are stored as target networks that only receive periodic soft updates for stability. Exploration of the environment is carried out by a noisy version of the policy, from which the policy gradient is sampled and used to update the actor and critic.

Evolutionary Reinforcement Learning

Evolutionary reinforcement learning (ERL) combines principles from Evolutionary Algorithms (EAs) and RL to address challenges in optimizing complex, high-dimensional, or partially observable environments Khadka and Tumer [2018]. In ERL, an EA is used to evolve populations of agents, leveraging its global search capabilities to explore diverse solutions. Concurrently, an RL technique is applied to refine individual policies through gradient-based learning, enabling local optimization and fine-tuning. This hybrid approach is particularly effective in introducing the benefits of fast gradient-based learning of RL to EAs, while improving RL solutions via the bias of EAs toward states with high long-term returns.

In the multiagent context, the Multiagent Evolutionary Reinforcement Learning (MERL) framework enables learning multiagent policies in a split-level reward setting Khadka et al. [2020]. MERL employs an EA to optimize a sparse, team-level reward while a DDPG loop optimizes individual agents' policies using a dense agent-specific reward. MERL stores experiences collected from the rollouts from both processes in a shared replay buffer, which is used to sample transitions by the DDPG loop and updates agent policies. MERL allows the efficient utilization of dense local rewards to achieve optimal team-level behaviors that are evaluated using sparse global rewards. As the treatment of the local and global rewards is executed separately, MERL bypasses the need for composing hybrid rewards from the local and global rewards, and thus requires little prior domain knowledge to learn. Additionally, this approach preserves the original true optimum for the problem by not modifying the reward structure.

To motivate this framework, we compare classical NSGA-II to NSGA-II + DDPG (MARMOT) on a simple instance of the Multi-Objective Rover Exploration Problem. We place three points of interest in a straight line, representing three distinct objectives. An agent is spawned closest to the central point of interest, and must navigate to observe the points. A key requirement in this problem is that the agent must plan sophisticated routes with sharp bends to observe multiple points – something that is challenging for sample-inefficient pure evolutionary approaches to learn in few generations. Figure 1 is an interesting demonstration of the value of gradient-based methods for learning key skills like navigation.

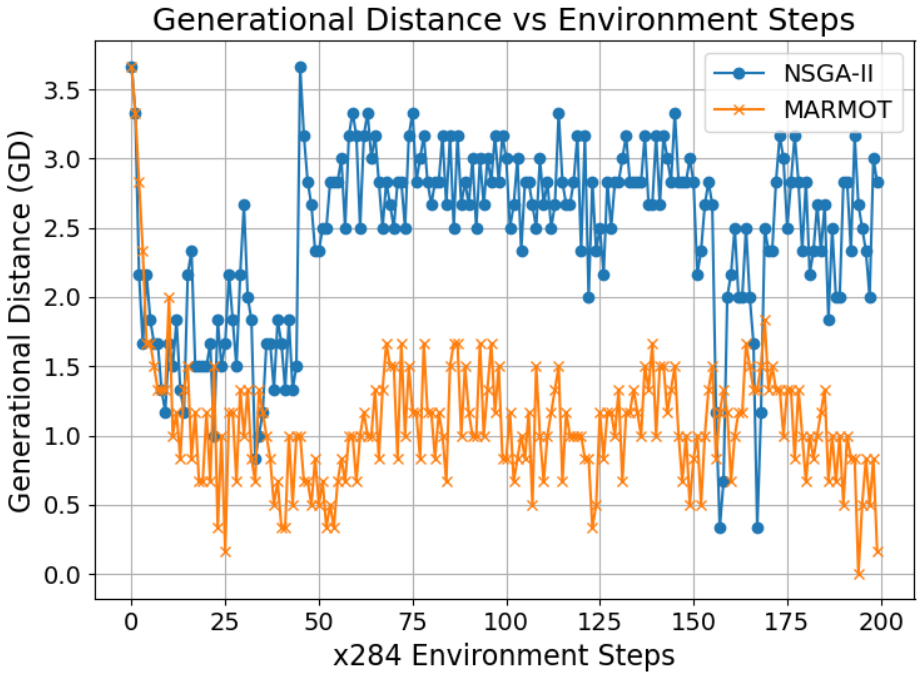

Figure 1: Generational Distance of Pareto front estimates to the ideal Pareto front (lower is better, zero is optimal).

Related Works

MO-MIX

Learning a Pareto front of coordinated multi-objective joint policies has been notably tackled by MO-MIX Hu et al. [2023]. MO-MIX is a multi-objective extension of QMIX, a state of the art MARL algorithm for discrete action spaces. MO-MIX extends QMIX by introducing a Q-value vector, which estimates the action-value across each objective. Although this approach has been shown to work well, it carries two key limitations:

- Suboptimal sparse reward performance: MO-MIX is a gradient-based method that optimizes the global team reward directly. However, in sparse team-reward settings, gradient-based multiagent methods have been shown to perform poorly Liu et al. [2023].
- Discrete action space: MO-MIX operates in discrete action-space. However, most real-world control and coordination problems involve agents that can take highly granular actions across several dimensions. Thus, agents in continuous domains would have to be externally discretized to successfully apply MO-MIX. Discretization of the action space comes with an inherent risk of not fully capturing the complexity of the problem.

Multi-Objective Joint Policy Evolution

Another branch of research on developing Pareto-optimal coordinated joint policies utilizes utility-free MOEAs. Notable work in this area has used NSGA-II for multi-objective joint policy evolution with mechanisms to solve the credit assignment problem Yliniemi and Tumer [2014], Thakar et al. [2024]. Although these methods have shown success in learning coordinated behaviors in sparse team-reward settings, they suffer from the sample inefficiency common to MOEAs. Additionally, MOEAs cannot leverage dense feedback (if available) in the environment, as the evolution process may only act upon aggregated reward values at the end of an episode – discarding useful timestep-level information about the rewards.

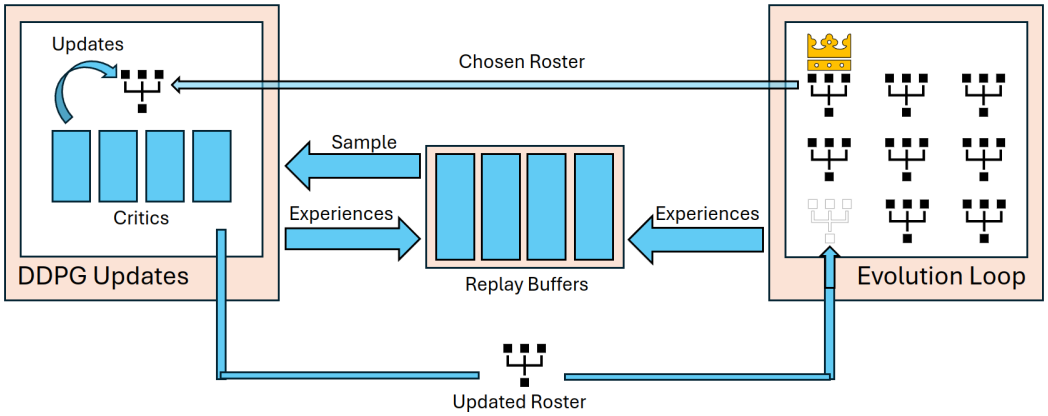

Figure 2: Complete procedure for MARMOT. The chosen roster is copied from the evolutionary population to the RL component for updates. Experiences from the evaluation of the evolutionary population, along with noisy policy rollouts from RL add diverse experiences to the Replay Buffer. The RL component samples the replay buffer to update the chosen roster. The updated roster is inserted back into the evolutionary population for the next generation.

3 Method

We now describe Multiagent Roster for Multi-Objective Teams (MARMOT), a split-level, multi-objective multiagent learning framework that learns a *roster* of policies from which a suite of teams, each expressing a different trade-off, can be composed.

3.1 A Multiagent Roster for Multi-Objective Teams

At a high level, MARMOT comprises two main components:

1. Gradient-based skill-building: We use DDPG to train indiviudal agents with a dense, agent-specific local reward that promotes basic skill-building.
2. Evolutionary multi-objective optimization: We use the gradient and utility-free NSGA-II algorithm to optimize team policies for the sparse, multi-objective global team reward.

During training, NSGA-II is run on a population of rosters. At each generation, a Pareto-optimal team is sampled, and it corresponding roster is shared with the DDPG component to receive directional policy updates. On the DDPG side, in the roster, agents of the sampled team receive policy updates from their own critics. We then reintroduce this updated roster in the evolutionary population, and allow the evolutionary component to operate on the updated population of rosters. In Figure 2, we depict this high-level procedure of MARMOT.

Our key insight is that by learning more policies than the actual size of the team, we allow the flexible composition of teams with varying trade-offs across the objectives. A key requirement for the success of this method is that agents on the roster learn different, yet *synergistic* behaviors. This would allow composing teams with diverse behaviors, comprising agents that are likely to succeed with each other. We achieve this by maintaining a distinct replay buffer for each agent. This preserves diversity in the experiences collected by each agent, and allows them to learn distinct behaviors. Each agent also has its own distinct critic, which operates only on the experiences and policies that agent. It is important to note that while agents have distinct critics, an agent exists in each roster. Thus, the critic learns a value function on the experiences of its agent from each roster – enabling powerful data sharing that causes an agent's behaviors across rosters to homogenize with each generation.

3.2 Policy Network Architecture

We represent a roster as a Multi-head policy Silver et al. [2017], Khadka et al. [2020], Aydeniz et al. [2023]. Each head (or output node) of the multi-headed network, represents a single agent's policy.

Thus, all agents in a roster share a 'trunk' (the lower layers of the network). This parameter-sharing makes training more efficient, and allows agents to benefit from globally beneficial mutations made to the network during evolution. Additionally, each agent k has its own replay buffer. This buffer is shared among agents at index k across all rosters. An important distinction between previous methods that employ the multi-headed network and ours is that networks have more heads than agents on a team. By making this modification, we can create and train a roster of agents which can later be leveraged to create several Pareto-optimal teams.

3.3 Team-Rewards Optimization (NSGA-II)

NSGA-II forms the basis for optimizing the team-level reward vector. However, the application of classical NSGA-II for evolving multiagent rosters is exceptionally challenging, as evaluation via policy rollouts is not possible with more agents than in the problem description. Thus, assigning a multi-objective 'fitness' for rosters in the NSGA-II population is nontrivial. MARMOT circumvents this problem by computing a fitness for rosters *by proxy*. Instead of attempting to perform rollouts with the whole roster, we probabilistically sample a fixed number of teams from a each roster in the population. We perform rollouts with each of these sampled teams. Then, we create a ranked list of the sampled teams using their non-domination levels and crowding distance, much like NSGA-II. We then use the Borda Count–a voting method that allots points to each candidate according to their standing–to determine which roster produced the 'worst' sampled teams. We then discard low-performing rosters from the population to free up space to introduce updated rosters from the gradient-based component and mutated copies of high performing rosters.

To determine which roster should be sent for gradient-based policy updates, we simply select a Pareto-optimal sampled team at random, and choose its associated roster. We then provide this roster, along with the exact composition of the team from this roster that was chosen, to the gradient-based optimization component.

3.4 Local-Reward Optimization (DDPG)

The gradient-based local-reward optimization process utilizes the standard DDPG algorithm to update the chosen roster network. This process receives both – the multi-headed roster network as well as the Pareto-optimal team from this roster that was selected by the evolutionary process. A noisy version of the team is used to conduct rollouts to explore the environment, and its experiences are added to the replay buffer. Crucially, due to the shared trunk parameters of the multi-headed network, all heads (or agents) in the roster receive updates even if the specific agent is not in the selected team. This property allows for all agents in the roster to improve in terms of the local, skill-building reward. After the completion of DDPG, the new multi-headed roster network is injected back into the evolutionary population to replace the lowest performing roster.

4 Experimental Setup

Variations of the Continuous Multi-objective Rover Exploration Problem (MO-Rover) are used to test MARMOT Thakar et al. [2024]. This serves as a proxy for real-world problems such as environmental monitoring Athanasiadis and Mitkas [2004] or ocean exploration in which a team of agents must learn to balance multiple objectives.

4.1 Problem Description and Reward Structure

A set of homogeneous rovers on a two-dimensional plane must learn to navigate to a set of points of interest (POIs). This variation of MO-Rover is a tri-objective problem in which there exist POIs of type A, B, and C. Each POI requires an agent to observe it by being inside the POI's observation radius. At the next timestep, the POI disappears and the team receives a unit of reward for each POI that is observed. A reward vector includes the normalized sum of rewards for observing POIs of type A, B, and C. Each team of rovers can express a different trade-off between these three objectives.

The environmental configurations for the experiments are presented in Table 1.

Property	Rover$_{multi}$
Size	14 x 20 units
Episode length	9 timesteps
Number of Type A POIs	2
Number of Type B POIs	2
Number of Type C POIs	2
Type A Obs. Radius	1
Type B Obs. Radius	1
Type A Reward	+1
Type B Reward	+1
Type C Reward	+1
Rovers in a Team	2
Roster Size	2, 3

Table 1: Experimental configurations for Rover$_{multi}$

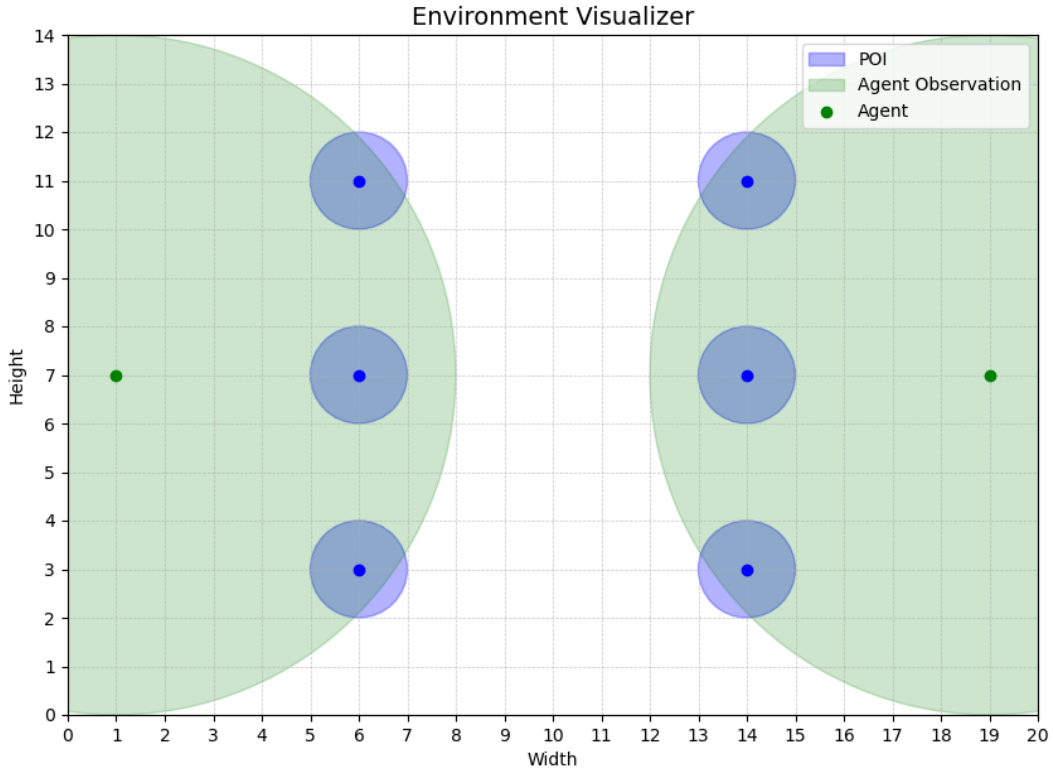

Figure 3: Environmental configuration for our experiment. The short episode length coupled with the sophisticated navigation requirements make this problem exceptionally difficult to solve in isolation with RL or Evolutionary methods.

4.2 Baselines

We compare the performance of MARMOT against classical NSGA-II. The final Pareto front of each method is plotted, and both the number of points on each method's Pareto front as well as which method's Pareto front dominates the others is evaluated. One statistical run is performed for each method.

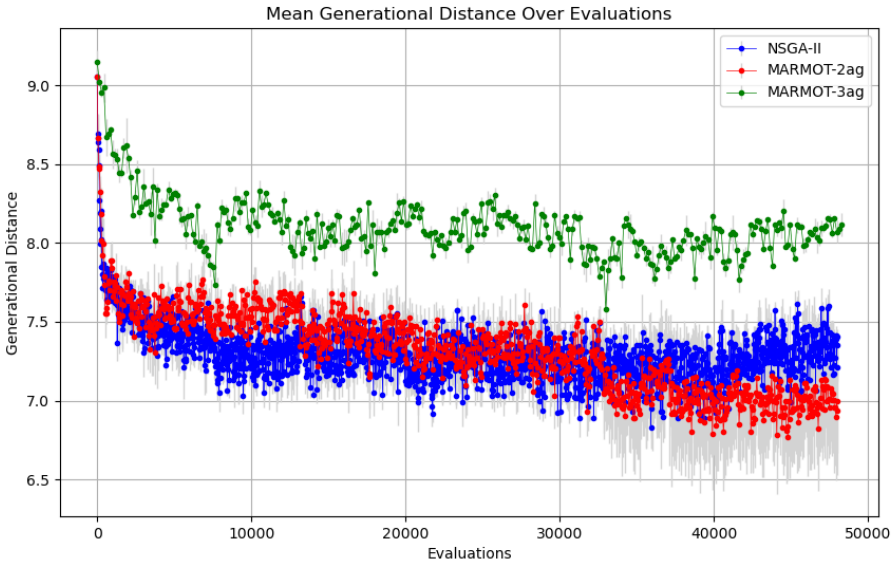

Figure 4: Comparison of the achieved Generational Distance across total policy evaluations. Lower is better (zero is optimal).

5 Results

In this section, we present the findings from our experiment in the multi-objective rover exploration problem domain. MARMOT-2ag is MARMOT with a roster size of 2, while MARMOT-3ag is MARMOT with a roster size of 3. Please note that the problem requires a team of 2 agents. Our results plot the mean statistic over 3 stat runs along with the Standard Error of the Mean (SEM). We use MARMOT-2ag to isolate the effect of introducing DDPG to purely Evolutionary Multi-Objective Learning, and use MARMOT-3ag to study the impact of training a roster of more agents than in the problem description.

Figure 4 shows the Generational Distance from the true Pareto front, achieved using each method. As is clear from the chart, MARMOT-2ag learns a Pareto front estimate that is closest to the true Pareto front. NSGA-II provides closely competetive solutions as well. The DDPG-introduced enhanced nature to track points of interests in the environment enables MARMOT-2ag to more closely perform Pareto-optimal behaviors and yield better Pareto front estimates.

A surprising result is the 'poor' performance of MARMOT-3ag, considering it contains the most sophisticated mechanisms for promoting diverse trade-offs in the solutions. Our explanation for this result is the suboptimal roster preservation mechanisms in MARMOT's evolutionary step. As we use the Borda count to evaluate rosters, it is possible that rosters that provide singular dominant solutions, but otherwise poor solutions on average, get discarded after the roster fitness assignment step. This is undesirable in our experimental setup with only three optimal trade-offs. When the Pareto front is sparse, and there are a limited number of team performances that any trajectory of agents may provide, it becomes exceptionally important to explicitly preserve dominant solutions. With MARMOT-2ag, this is not an issue, as each roster is evaluated according to the one team it supplies, resulting in an accurate mapping of team performance to roster fitnesses. Our intuition is that MARMOT-3ag is likely to succeed in settings with a vast range of Pareto-optimal trade-offs and more granular team performances, where the notion of rosters being evaluated on the 'overall' quality of their teams is more robust.

We would like to point out that even though MARMOT-3ag performs worse compared to other methods, it still demonstrates a clear trending – a promising start to performing new experiments to back its validity.

6 Discussion and Future Work

In this work we proposed MARMOT, a utility-free multi-objective multiagent learning framework that is able to produce a range of Pareto-optimal trade-offs in team performance. We presented a detailed methodology, and tested MARMOT against NSGA-II in an of the Multi-Objective Rover Exploration Problem. We observe that MARMOT is able to leverage the quick skill-building capabilities of RL for navigation, along with the density-agnostic global reward maximization power of EAs to produce high-performing teams. Our results show that MARMOT learns consistently, and lay the foundation for extensive experimentation on the method. We now discuss some limitations and future work. As MARMOT combines two sophisticated learning methodologies – RL and Evolution, it is highly dependent on the algorithmic configuration and hyperparameters. Owing to this, our testing of MARMOT in several domains, and ability to extract its peak performance was limited in this work. The potential 'misalignment' between local and team rewards is also worth investigating, as we are unable to make claims on performance in diverse reward structures.

For future work, we aim to perform substantial testing and tune the performance of our method. We would also like to test MARMOT against MO-MIX, as this comparison would serve as a benchmark against the current state of the art. Lastly, we would like to introduce MARMOT in the MO-Starcraft domain to test its performance in dynamic cooperative multi-objective settings.

References

Ioannis Athanasiadis and Pericles Mitkas. An agent-based intelligent environmental monitoring system. *Management of Environmental Quality*, 15, 09 2004. doi: 10.1108/14777830410531216.

Ayhan Alp Aydeniz, Robert Loftin, and Kagan Tumer. Novelty seeking multiagent evolutionary reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 402–410, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi: 10.1145/3583131.3590428. URL `https://doi.org/10.1145/3583131.3590428`.

Andrew G Barto. Reinforcement learning: An introduction. by richard's sutton. *SIAM Rev*, 6(2):423, 2021.

Matteo Cristani, Luca Pasetto, and Claudio Tomazzoli. Protecting the environment: a multi-agent approach to environmental monitoring. *Procedia Computer Science*, 176:3636–3644, 2020. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.09.336. URL `https://www.sciencedirect.com/science/article/pii/S1877050920322547`. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 24th International Conference KES2020.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

Tianmeng Hu, Biao Luo, Chunhua Yang, and Tingwen Huang. Mo-mix: Multi-objective multi-agent cooperative decision-making with deep reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12098–12112, 2023. doi: 10.1109/TPAMI.2023.3283537.

Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Shauharda Khadka, Somdeb Majumdar, Santiago Miret, Stephen McAleer, and Kagan Tumer. Evolutionary reinforcement learning for sample-efficient multiagent coordination. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

TP Lillicrap. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Boyin Liu, Zhiqiang Pu, Yi Pan, Jianqiang Yi, Yanyan Liang, and D. Zhang. Lazy agents: A new perspective on solving sparse reward problem in multi-agent reinforcement learning. In Andreas

Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 21937–21950. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/liu23ac.html.

Ying Liu, Brent Logan, Ning Liu, Zhiyuan Xu, Jian Tang, and Yangzhi Wang. Deep reinforcement learning for dynamic treatment regimes on medical registry data. In *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 380–385, 2017. doi: 10.1109/ICHI.2017.45.

Roxana Rădulescu, Patrick Mannion, Diederik M Roijers, and Ann Nowé. Multi-objective multi-agent decision making: a utility-based analysis and survey. *Autonomous Agents and Multi-Agent Systems*, 34(1):10, 2020.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017. URL https://arxiv.org/abs/1712.01815.

Raghav Thakar, Gaurav Dixit, and Kagan Tumer. Multi-objective credit assignment for multiagent systems. *Multi-Objective Decision Making Workshop at European Conference on Artificial Intelligence*, 2024.

Yan Wu, Mingtao Nie, Xiaolei Ma, Yicong Guo, and Xiaoxiong Liu. Co-evolutionary algorithm-based multi-unmanned aerial vehicle cooperative path planning. *Drones*, 7(10), 2023. ISSN 2504-446X. doi: 10.3390/drones7100606. URL https://www.mdpi.com/2504-446X/7/10/606.

Logan Yliniemi and Kagan Tumer. Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. In Grant Dick, Will N. Browne, Peter Whigham, Mengjie Zhang, Lam Thu Bui, Hisao Ishibuchi, Yaochu Jin, Xiaodong Li, Yuhui Shi, Pramod Singh, Kay Chen Tan, and Ke Tang, editors, *Simulated Evolution and Learning*, pages 407–418, Cham, 2014. Springer International Publishing. ISBN 978-3-319-13563-2.