Novelty Seeking Multiagent Evolutionary Reinforcement Learning

Ayhan Alp Aydeniz
Collaborative Robotics and Intelligent Systems Institute
Oregon State University
Corvallis, Oregon, USA
aydeniza@oregonstate.edu

Robert Loftin
Department of Intelligent Systems
Delft University of Technology
Delft, The Netherlands
r.t.loftin@tudelft.nl

Kagan Tumer
Collaborative Robotics and Intelligent Systems Institute
Oregon State University
Corvallis, Oregon, USA
kagan.tumer@oregonstate.edu

ABSTRACT

Coevolving teams of agents promises effective solutions for many coordination tasks such as search and rescue missions or deep ocean exploration. Good team performance in such domains generally relies on agents discovering complex joint policies, which is particularly difficult when the fitness functions are sparse (where many joint policies return the same or even zero fitness values). In this paper, we introduce Novelty Seeking Multiagent Evolutionary Reinforcement Learning (NS-MERL), which enables agents to more efficiently explore their joint strategy space. The key insight of NS-MERL is to promote good exploratory behaviors for individual agents using a dense, novelty-based fitness function. Though the overall team-level performance is still evaluated via a sparse fitness function, agents using NS-MERL more efficiently explore their joint action space and more readily discover good joint policies. Our results in complex coordination tasks show that teams of agents trained with NS-MERL perform significantly better than agents trained solely with task-specific fitnesses.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent systems**; **Multi-agent reinforcement learning**.

KEYWORDS

Multiagent Learning, Fitness Shaping, Evolutionary RL, Exploration

ACM Reference Format:
Ayhan Alp Aydeniz, Robert Loftin, and Kagan Tumer. 2023. Novelty Seeking Multiagent Evolutionary Reinforcement Learning. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583131.3590428

1 INTRODUCTION

Multiagent systems (MAS) have been successfully applied to many real-world applications from air-traffic management [9], to multi-UAV surveillance systems [1]. In these applications, it is critical for agents to closely coordinate their actions so that they can achieve their common objective. This makes it difficult to discover desirable joint actions because all agents need to take the *correct* actions simultaneously. Evolutionary MAS algorithms that directly maximize a scalar—and team-based—fitness often struggle to explore the joint-state space sufficiently to unearth these key joint-actions.

Novelty-search methods [7, 14, 15, 19, 36] alleviate this problem by explicitly searching for diverse behaviors. They therefore cover a larger portion of the solution space than would methods directly aimed at achieving a particular objective. However, they require a pre-defined diversity metric to distinguish behaviors based on their contribution to diversity. In MAS, deriving a diversity metric that balances the need to maintain both similar and distinct behaviors in a population is problematic, and incorrect choices can lead significant reduction of behavioral diversity. Evolutionary Reinforcement Learning (ERL) [29] aims to balance this by using a policy-gradient algorithm to focus on locally defined objectives. The Evolutionary Algorithm (EA) then incorporates the Reinforcement Learning (RL) policies into its population to ensure both local and global objectives are met. However, this approach is brittle and fails to learn coordination in complex tasks, primarily due to the vastly differing convergences [21, 23, 27, 28] between the RL and EA modules.

In this paper, we introduce Novelty Seeking Multiagent Evolutionary Reinforcement Learning (NS-MERL) that explores the agents' joint-state space through both providing diversity-based fitnesses to the gradient-based algorithms and seeking to increase diversity directly during evolutionary learning.

NS-MERL relies on a count-based estimate of the novelty of states to construct the fitness function. Individually exploring the state space more widely leads agents to discover those joint-states that represent coordination with their teammates. This increases the likelihood of receiving non-zero team fitness values.

Our main contribution is to balance exploration at agent-level and exploitation at team-level within the framework of Multiagent ERL (MERL) [29]. NS-MERL provides novelty-seeking fitness shaping for MAS that enables agents to explore their joint behavior space efficiently. Our experiments show that our agents out-perform current state-of-the-art methods in the tightly-coupled multi-rover exploration domain where agents need to coordinate to observe points of interest (POIs). We show that our fitnesses (which encourage exploration of the environment rather than task performance) result in an evolutionary search which is resilient to increasing task complexity. The performance of behaviors learned with the aide of our method is robust to increases in the numbers of agents, and the degree to which the success of an individual agent's strategy depend on the strategies of its teammates.

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590428

2 BACKGROUND AND RELATED WORK

We consider cooperative multiagent tasks formalized as *decentralized partially-observable Markov decision processes* or Dec-POMDPs [33]. We focus specifically on *tightly-coupled* tasks, by which tasks in which several agents must select compatible strategies in order to succeed [2, 10, 22, 26, 30]. Here the *coupling factor* of a task will refer to the number of agents that need to coordinate their strategies to solve the task (or a specific sub-task).

A Dec-POMDP is defined by a tuple $\mathcal{J} = \{N, \mathcal{S}, \mathcal{A}, P, \mathcal{O}, G, b_0, h, l, \pi\}$, where N is the number of agents in the environment, $\mathcal{S}$ is the state space, $\mathcal{A} = \prod_{i \in N} \mathcal{A}_i$ is the *joint* action space and $\mathcal{O} = \prod_{i \in N} \mathcal{O}_i$ is the joint observation space. $P(s^{t+1}|s^t, a^t)$ is the state transition distribution, $O(o^t|s, a)$ is the joint observation distribution, and $b_0(s^0)$ is the initial state distribution. All episodes end after l steps, with a total fitness of $\hat{G} = \sum_{t=0}^{h-1} G(s^t)$, where G is the per-step fitness function. However, $\hat{G}$ is often available at the end of l steps, so it is a sparse fitness. Agents do not directly observe the state, but instead only have access to their individual observations o_i^t. Therefore, agent $i \in N$ can only condition their actions on their individual action-observation history up to time t, which we denote as $h_i^t = (o_i^0, a_i^0, \ldots, o_i^{t-1}, a_i^{t-1}, o_i^t)$. As states, observations and actions may all be stochastic, the fitness of a team is defined as the expectation of $\hat{G}$. Our goal in Dec-POMDPs is to learn a joint policy $\pi = (\pi_1, \ldots, \pi_N)$ for all agents that maximizes this expectation.

2.1 Evolutionary Reinforcement Learning

Evolutionary algorithms (EAs) are advantageous in reinforcement learning (RL) tasks with sparse fitness functions and/or challenging credit assignment problems, as their behavior depends only on the final expected fitness of a policy [35]. At the same time, in high-dimensional policy spaces they can be slow to converge relative to RL methods that leverage the structural information provided by the gradients of the policy. The *evolutionary reinforcement learning* (ERL) framework [23] combines the benefits of both approaches, utilizing a gradient-based RL algorithm to train policies based on dense, heuristic fitness functions [24], while using and EA to fine-tune the final, sparse fitness of the policy.

Multiagent Evolutionary Reinforcement Learning (MERL) [29] applies ERL to MAS, with the EA optimizing the true, global fitness function, while the RL algorithm learns from denser and more informative local fitness functions defined for each individual agent. Our implementation of MERL uses the same configuration as [29], and we use the same neuroevolution mechanism [16] to optimize the global fitness of the joint policy. ERL (and by extension MERL) rely on dense, task-specific fitness functions that can be thought of as heuristics to help the agents learn useful skills and representations. In this work, we consider how task-independent individual fitness functions can be used to encourage good exploratory behavior. We summarize the MERL framework in Figure 1.

2.2 Related Work

In the literature, novelty can be searched at various levels. On one hand, some works [11, 14, 20, 36] achieve diversity within a set of behaviors by searching in the action-space. On the other hand, there is a good amount of research, *e.g.* [3, 4, 7, 39], achieving exploration by searching for novel states that are observed not as frequently as the other visited states. Overall, although these techniques search for novelty at different levels, they all aim to enhance the information gain during training.

In our method, the main driving force of exploration is count-based exploration. It is a popular novelty search method that pioneered the literature on intrinsic motivation [5]. These methods often aim to reduce the frequency of states throughout an episode [7] or training [4] (or both), so that agents are encouraged to detect more novel states or state transitions that allow them to learn behaviors that are of high fitness. Depending on the nature of a task, some state transitions can be more significant than the others. These events are often referred as salient events and these events can be incorporated into the *count* function counting the number of times a state has been visited through the errors in the prediction of salient events and significant changes during state transitions [6, 7]. In continuous domains where we might have raw images, sensor values, or other high-dimensional state-spaces, applying a standard count-based method does not function well, since most states will likely occur once. Some methods [3, 4, 7, 17, 34, 39] use different types of discretization techniques, such as dynamic hashing [39], nearest neighbor search [4] or with density models [34]. However, counting states (or observations) throughout an episode does not consider occurrences of the same states, the work [4] counts the occurrences of states during training as well, so that their algorithm is able to compare different episodes according to their novelty. In MAS, counting states is of difficulties, due to the dynamic nature of state-space affected by the actions of multiple agents. In addition, every agent or sub-teams of agents can desire different set of novel states and keeping a track of this during training is another challenging avenue. In the RL framework, count-based fitnesses have a non-Markovian nature, but it has been shown that they help agents reduce the uncertainty about the environment in the work [7].

In the algorithms searching for novel behaviors explicitly during training has been gaining traction especially in the recent years. Among EA-based methods, Quality Diversity (QD) methods offer a path to novelty search by specifically searching for diverse behaviors that are of high quality [11, 31, 36]. However, for multiagent problems, searching exhaustively through a large behavior space is often not tractable and requires the acceptance of significant loss of information by reducing the dimensions of a behavior space. In continuous RL settings, some works utilized information theory, *e.g.* [14, 20, 43], to define novelty through a more probabilistic metric.

3 NOVELTY-SEEKING AGENTS

While dense local fitness functions can help individual agents learn useful skills, they do not necessarily address the challenge of coordination between agents in tightly-coupled tasks. For example, in the Rover domain described in Section 4.1, we can provide a fitness to agents for moving towards the nearest point of interest (POI), but this may conflict with the need for multiple rovers to converge on a single POI, or the need to spread out to cover multiple POIs. In order to identify these types of coordinated behaviors, during training the agents need to effectively *explore* the joint strategy space. In this section we describe how an intra-episode *novelty-seeking* local fitness can be used to encourage such exploration.

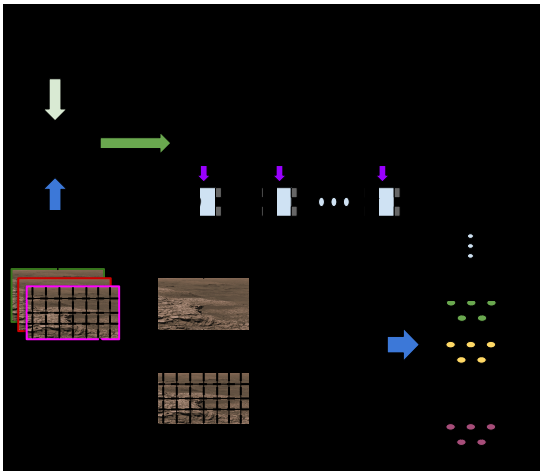


Figure 1: MERL with novelty seeking agents. Individual policies are trained via local fitnesses. The final joint policies are optimized via an EA optimizing the lobal fitness. [29]

3.1 Novelty-Seeking Fitness Function

A novelty-seeking local fitness function encourages each agent to efficiently explore its own observation space within a single episode. For discrete observation spaces, the fitness for agent i is

$$f_{\text{ns}}(h_i^t) = \frac{1}{\text{count}(o_i^t, h_i^t)} \tag{1}$$

where $\text{count}(o_i^t, h_i^t)$ is the number of times i's most recent observation o_i^t occurs in its observation history up to time t. Therefore, if o_i^t has never been observed previously, $f_{\text{ns}}(h_i^t) = 1$. This function is maximized by policies that cover as much of the observation space as possible within an episode. For continuous, vectored observation spaces, we found that using a coarse discretization to estimate $\text{count}(o_i^t, h_i^t)$ was sufficient to encourage exploration. This modification is only applied to compute the fitness value, the states provided to the policy networks do not go through this change.

As maximizing intra-episode novelty alone may not lead to effective behavior under the global fitness function, we combine the novelty-seeking fitness with a task-specific heuristic local fitness function. Heuristic fitnesses ease the search by rewarding specific states, so they improve the effectiveness of the search in $\mathcal{S}$ [8]. The resulting fitness function encourages exploration over *salient* parts of the state space. Let $V(h)$ be a heuristic local fitness function for agent i, we then define the combined local fitness

$$f_{\text{salient}}(h_i^t) = \begin{cases} f_{\text{ns}}(h_i^t)V(h_i^t), & \text{if } V(h_i^t) > 1 \\ f_{\text{ns}}(h_i^t), & \text{otherwise} \end{cases} \tag{2}$$

Incorporating this information via multiplication allows agents to experience the values of salient events during an episode. This leads agents to experience a saliency as a value discounting over time to encourage exploration of other salient events. While the true state s^t is not observable during deployment, and so the policies themselves cannot depend on it, it is available during training, and so in principle the heuristic function could depend on s^t directly. If $V(h_i^t)$ is less than 1, the value needs to be scaled in some domians.

Difference Evaluation Functions. As the heuristic fitness functions do not directly capture a team's ability to coordinate, we also consider combining the novelty-seeking fitness with a signal derived from the global fitness. We use *difference evaluation functions* (DEFs) [2] to evaluate each agent based on its individual contribution to the global (team) fitness. DEFs compute the difference between the true global fitness and the fitness that would have been achieved without the contribution of a given agent, and have been shown to be effective in a number of different settings [2, 12, 32, 40–42]. While DEFs help each agent evaluate their own actions, they are only non-zero when the global fitness is also non-zero, and so are as sparse as the global fitness. We therefore combine them with both the novelty-seeking and heuristic fitness functions. In general, the difference evaluation function for agent $i \in N$ is

$$D_i(s) = G(s) - G(s_{-i} \cup c_i) \tag{3}$$

where $s_{-i} \cup c_i$ is a version of state s in which agent i's contribution has been removed and replaced with their counterfactual contribution under the default behavior c_i. In our experiments this is calculated by simply removing agent i from the environment altogether. The local fitness incorporating the DEF is then

$$f_D^i(s^t, h_i^t) = \begin{cases} f_{salient}((h_i^t)\beta D_i(s^t), & \text{if } D_i(s^t) > 0 \\ f_{salient}((h_i^t), & \text{otherwise} \end{cases} \tag{4}$$

where β is scaling factor to not interrupt the effect of salient events. DEFs may be a value between 0 and 1, so we scale them to a value that does not decrease the value obtained by $f_{\text{salient}}(h_i^t)$. As the DEF will typically be sparse, we only include it when its value is non-zero. β is set to 10 for all experiments. We summarize the calculation of local fitness for a single agent in Algorithm 1

Algorithm 1: Computes a sequence of local novelty-seeking fitnesses for agent i over a single episode.

```
Initialize state s^0, history h_i^0 = {o_i^0}.
for t ∈ [0, l − 1] do
    Retrieve action a_i^t, state s^{t+1} and observation o_i^{t+1}
    count ← 1
    for o_i ∈ h_i^t do
        if quantize(o_i^{t+1}) == quantize(o_i) then
            count ← count + 1
    fitness ← 1/count
    h_i^{t+1} ← h_i^t ∪ {a_i^t, o_i^{t+1}}
    if V(h_i^{t+1}) then
        fitness ← fitness * V(h_i^{t+1}) (from Eq. 2)
    if use_DEF and D_i(s^{t+1}) > 0 then
        fitness ← fitness * D_i(s^{t+1}) (from Eq. 4)
    Yield fitness
```

3.2 MERL Algorithm

A key idea behind our approach is that efficient exploration of the joint strategy space can be achieved by encouraging agents to explore their individual strategy spaces. Good exploratory behavior,

however, is often very different from the behavior that maximizes team-level fitness. The MERL algorithm provides a natural framework for resolving this conflict. Within our implementation of MERL, the gradient-based TD3 algorithm [18] learns exploratory policies for individual agents based on our novelty-seeking fitness function, while the evolutionary algorithm [16] coordinates team-level behavior to maximize global fitness.

Algorithm 2: Multiagent Evolutionary Reinforcement Learning (MERL) [29] with Novelty Shaped Fitnesses

```
Initialize a population of M multi-head actor teams each
  having N agents, pop_π (see Figure 1)
Initialize a set of N replay buffers and N local TD3 agents
for gen ∈ [1, ∞] do
    foreach team π ∈ pop_π do
        Fitness = 0
        for t ∈ [0, Timesteps] do
            foreach agent A ∈ A_1, ..., A_N do
                Compute local fitness, f^t,
                    (via Algorithm 1)
                foreach ReplayBuffer R ∈ R_1, ..., R_N do
                    append (o^t, o^{t+1}, a^t, f^t) to R
        Compute team fitness G
        Assign G as Fitness of team π
    // Evolve pop_π
    Rank lineage according to the fitness of each team
    Sample elites via a roulette wheel [25]
    Mutate Sampled Elites
    Return the Champion
    Send the policy gradient team to pop_π replace with the
      team having the lowest fitness
```

In Algorithm 2, each team is represented by a multi-headed actor with N heads, which allows each agent to learn a distinct policy, while still sharing learned state representations. MERL maintains an evolutionary population pop_π of M teams, along with a team policy that is updated using TD3. We define separate *replay buffers* for each agent, which store histories of observations, actions, and fitness signals on which the RL algorithm will train. A mini-batch of each agent's transitions, (o^t, o^{t+1}, a^t, f^t), from each team of pop_π, are stored in each agent's replay buffer. In each generation of the EA, each member of the population is evaluated by "rolling-out" the joint policy multiple times in the environment, and observing the average fitness. Experience data from these rollouts is also added to the replay buffers of each agent, along with local fitness values computed during each rollout. After the evaluation data is collected, TD3 updates are then performed on the RL policy using data sampled from the replay buffers, training each agent's policy to maximize its own local fitness. Neuroevolution is then used to update the EA population based on the global fitness. Finally, the team with the lowest fitness is discarded, after ranking, we sample group of teams (teams with higher fitness values), then we choose the best team (the *champion*) for evaluation.

4 EXPERIMENTS

Our experiments seek to answer the following questions:

- How does the coupling factor affect the performance of our approach relative to other methods?
- How well does our method scale as the numbers of agents and the size of the joint-state space grow?
- How does the coupling factor affect the behavior of the exploration policies learned with our method?

We conduct our experiments in several configurations of the multi-rover exploration domain, a tightly-coupled cooperative task in which agents must organize themselves into smaller teams to solve different sub-tasks. The rover domain allows us to vary the number of agents and sub-tasks, as well as the *coupling factor*, that is, the number of agents required to solve each sub-task.

4.1 Multi-rover Exploration Domain

In this well-established domain [37], a team of rovers must learn to navigate to several different *points of interest* (POIs) in a continuous 2D environment. The domain is tightly coupled because for a single POI to be considered "explored" it must be observed simultaneously by multiple rovers. To observe a POI, a rover must be within the activation radius. The number of rovers required is the same for every POI, and we will refer to this as the coupling factor. When a POI is observed it is removed from the environment.

The rover domain is partially observable, with each agent only observing the environment through its own sensors. Each rover has 8 sensors, 4 rover sensors and 4 POI sensors. These sensors are positioned in 4 quadrants covering 90°each, and each quadrant has one POI and one rover sensor. The sensors can only detect if the objects are within a fixed distance. If there are more than one POI or rover within their observation area, they can only detect the objects resulting in the maximum value by functions in Eqs 6.

If the distance to an object is smaller than a fixed value, it is rounded to that value. This is given in the equation 5, where x and y are the points and d is a fixed minimum value like 0.1.

$$\delta(x, y) = max\{||x - y||^2, d^2\} \tag{5}$$

POI sensors' readings are calculated as in equation 6, W_{POIj} is the worth of a POI_j in a quadrant from the sensors of an agent, A_i.

$$s_{POI} = max(\frac{W_{POIj}}{\delta(A_i, POIj)}), s_{rover} = max(\frac{1}{\delta(A_i, A_j)}) \tag{6}$$

A rover A_i's sensors convert the input signal reflected by a teammate A_j to values as in equation 6.

The sensor readings of each rover forms the joint state of the environment. The global fitness is computed via Equation 7 where G represents the global fitness, N is the number of POIs, W_{POI_k} is the value of the kth POI, and $I(POI_k)$ is the Boolean function that returns 1, if POI_k is observed, 0, otherwise. The ratio between the sum of the values of observed POIs by the sum of the all POI values gives the G of a team. To achieve this fitness partially, agents need to learn to coordinate to accomplish tasks, otherwise, it stays as 0.

$$G = \frac{\sum_{k=1}^{N} W_{POI_k} I(POI_k)}{\sum_{i=1}^{N} W_{POI_j}} \tag{7}$$

4.2 Hyperparameters

The hyper-parameters of MERL are provided in Table 1. These parameters are consistent with the parameters used in the paper [29]. Our main claim is that agents trained via novelty-oriented fitnesses have the potential to perform even under high complexity. We use mainly **coupling factor** and **number of agents** as our parameters influencing the complexity level within a global task. **Task diversity** (in terms of their values) and **task distribution** are used as environmental parameters. Those are to set a level-of-difficulty for exploration in the environment.

Table 1: Hyper-parameters used for MERL and other two components, EA and RL

Component	Hyper-parameter	Setting
MERL	Population Size	10
	Rollout Size	50
	Actor Learning Rate	5e-5
	Critic Learning Rate	0.1
	Replay Buffer Size	1e5
	Batch Size	512
EA	Number of elites	4
Specifics	Number of anchors	5
	CCEA Reduction	Leniency
	Mutation Prob	0.9
	Crossover Prob	0.1
RL Actor	Actor Architecture	[100, 100]
Specifics	Critic Architecture	[100, 100]
	TD3 Noise variance	0.2
	TD3 Noise Clip	0.5
	TD3 Update Frequency	2
Rover Domain	State Dim (8sens+ID+vel)	11
	Action Dim	2

Baseline Methods for Comparisons: In multi-rover exploration domain, the objective is to observe POIs as a team of rovers; therefore, Khadka *et al.* [29] defines the objective function, $f_{extrinsic}$, to minimize the distance to the closest POI [29].

$$f_{extrinsic} = \frac{R_{act}}{d_{POI_c}} \tag{8}$$

where R_{act} is the activation radius of a POI, d_{POI_c} is the distance to the closest POI (cannot be captured by the sensors), POI_c. Here, we refer $f_{extrinsic}$ as task-oriented objective, because the global objective is to observe POIs within a certain radius and it is based on maximizing a specific extrinsic objective. This is a ***greedy*** approach, because they only construct a fitness toward the closest (the easiest to experience) saliency. Never Give Up (NGU) [4] intrinsic fitness function is a state-of-the-art novelty-oriented fitness for single-agent systems. Its episodic fitness function utilizes an embedding network to embed states into the history of states given in Sec. 3.1 to count high-dimensional states. We compare with NGU fitness function to show that having an embedding network is difficult to train in tightly-coupled MAS without a pre-training phase on task-oriented fitnesses. We use the same hyperparameters as in the paper [4]. We only compare our coarse discretization based counting to KNN search based counting of embedded states (by a neural network), so we do not consider the general learning framework of NGU.

Our agents are challenged under four main configurations testing them against, *increasing coupling factor*, *increasing team and task complexity*, *increasing growth in the state-space*, and *densely distributed tasks*. In the first set of configurations agents are tested with the same valued and randomly distributed POIs. We keep the number of agents, and POIs constant and increase the coupling factor. The second set is to test agents' exploratory behaviors under different couplings. There are two types of POIs, low-valued, and high-valued. Low-valued POIs are randomly distributed on an inner circle and high-valued POIs are distributed on an outer circle. Agents are expected to learn a behavior allowing them not to get stuck around low-valued POIs, so that they discover high-valued POIs. This setup includes experiments under both different coupling factors and numbers of agents. The third configuration has a constant coupling factor, but we change the number of available sub-tasks (POIs) and the number of agents (to grow state-space). Last setup has a dense distribution of POIs with different numbers of coupling, where agents can discover policies in a wider range.

To introduce salient events to our novelty-shaped fitness (see Eq. 2), we include the value of the POI, W of Eq. 6, as $V(h_i^t)$, when an agent enters its the activation radius in multi-rover domain. The coarse discretization mentioned in Sec. 3.1 is done by converting the values of observations to binary values. The sensor values not detecting any object are converted to 0, any detection's value is 1.

5 RESULTS

We report the global team performance of a multiagent team by computing as: *Champion* of Alg. 2, the team with the highest fitness, is selected (in Algorithm 2) in each generation, then it is tested on 10 rollouts of the test environment. We record the average score of these 10 rollouts achieved by the champion. The x-axis shows the number of gradient steps (number of times local fitnesses are provided to agents) during training. These metrics are determined according to the work [29]. Each plot is average of 5 statistical runs.

Note that our novelty seeking agents are represented as NS (using the fitness given in Eq. 2), and NS_D (given in Eq. 4) in the plots. Task-oriented agents (using the fitness of Eq. 8) are represented as TO, D and NGU appears only on one set of figures (Figure 2). D represents the case when we use pure DEF (of Eq. 3), and NGU represents when we use an embedding network and a KNN search to count states without any discretization.

5.1 Increasing Coupling

In our primary set of experiments, we analyze the behavior of our agents for different coupling factors. This dependency certainly affects the exploratory behavior of the team, because only a small portion of the joint-state space yields a positive team fitness. In this setup, we compare our baseline with 1) other fitness construction strategies, D, and TO, 2) a state-of-the-art novelty-search strategy, NGU. The first setup has 4 randomly distributed and equally valued POIs. Figure 2 shows the results for the degrees of coupling 1, 2, and 4. In the simplest task, where only one agent is required to successfully observe a POI, the team task is easy to learn for the embedding network, and NGU outperforms NS agents. Our agents achieve the same performance only by injection of DEFs into the fitness function represented as NS_D. As the coupling factor increases, the embedding network fails to achieve an effective performance to embed states, and this results in failure of all agents. All methods

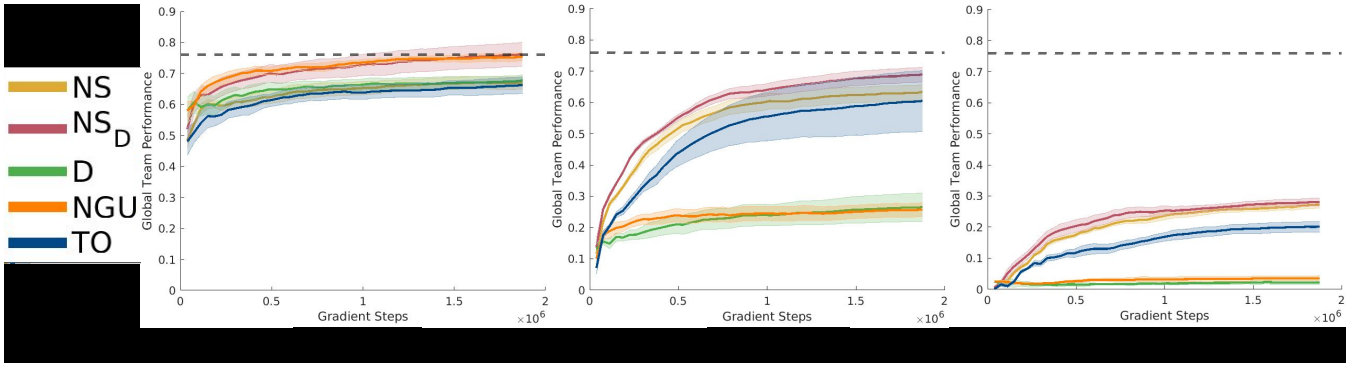


Figure 2: Performances achieved during 250 generations of EA's training in the first environmental setup - 6 rovers and 4 randomly distributed and equally valued POIs on a 15x15 map and episode length is 50. Dashed (–) lines represent the threshold for maximum G (Global Team Performance). Novelty seeking (NS) and NS with DEFs agents outperform the task-oriented (TO) agents, pure DEF (D), and Never Give Up (NGU) in each configuration.

generally achieves a similar score, but only NS_D and NGU agents reach the full team performance. As coupling increases, pure D agents do not exhibit an effective performance in general, because DEFs provide a sparse fitness that agents only achieve when they contribute to a collaborative task.

5.2 Increasing Team Size and Coupling

In these experiments we examine the effects of increasing the number of agents and the number of POIs. The environment has 3 POIs valued 2 and placed on an inner circle, and 3 other POIs valued 5 are distributed on an outer circle. Each circle represents equally distant three POIs. So, we expect agents to first observe the POIs on the inner circle, then the outer ones. Parameters setting the level of complexity are both the degree of coupling (1, 2, 3, 4, 5, 6) and the number of agents (3, 6, 9, 12, 15, 18) (Figure 3). We set the number of agents according to the coupling requirement. The effect of DEFs can be seen for the simplest scenario (as in the first setup). TO agents outperform both NS and NS-D agents in simpler scenarios, for the coupling factors, 1 and 2. With the increase in both the degree of coupling and number of agents, we observe that TO agents get affected significantly more than NS agents. Figure 3a shows that NS agents' performances stay almost constant.

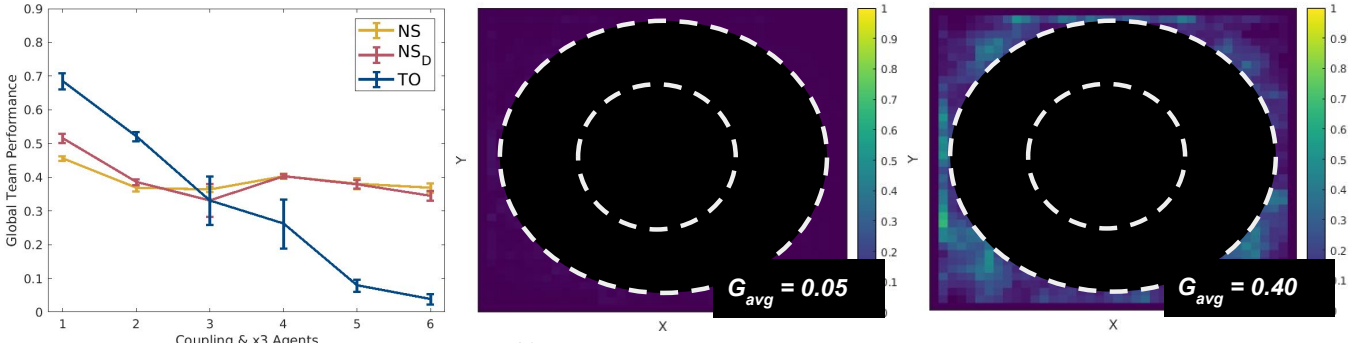
The second setup requires agents to learn an exploratory behavior. To analyze this we use heat-maps to compare our agents with TO agents. In Figure 3c, agents learning via our method expand over the environment and POIs more uniformly. The warmer parts of the figure represent the circles where we locate the POIs on. POIs are all equally distant from each other; therefore, we can see a uniform distribution of agents over the tasks. The heat is more concentrated on the inner circle where POIs are closer to each other, whereas this heat is more widely distributed over the outer circle. As we know from Figure 3a, TO agents fail under couplings 4, 5 and 6. Figure 3b shows that TO agents get stuck around center of the environment mostly and they are not able to expand in a useful manner for the whole team. TO agents first try to observe the inner POIs; however, due to high dependency on each other's behavior, they fail. NS agents are exploration driven; hence, RL agents provide policies that observe outer POIs, EA learns to observe all POIs successfully.

5.3 Scalability

Scalability to tasks with different numbers of agents and numbers of POIs is important to see how agents' performances are affected by the growth in their joint-state space. In our third set of experiment, we use a fixed coupling factor of 4 and change the number of POIs and the number of agents. We still us a circular distribution of POIs; however, we now vary the number of POIs on each circles For instance, the case with the 2 sets of POIs, the inner circle has 2 POIs valued 2, and 2 outer POIs valuing 5. The POIs are equally distant from each other; however, when we locate the POIs on the outer circle we shift their positions within a randomly chosen angle between 0°and 30°. When there are only for 4 agents, and only one set of POIs, a promising behavior would be to first observe the closest POI, then the outer POI. However, when there are POIs on multiple directions, then expanding over those POIs gets more difficult, because agents need to learn to expand uniformly over POIs. Therefore, we see a much higher performance when there is a set of POIs as Figure 4 shows. NS agents scale better as their state-space gets larger, whereas we see a slight, but constant decay in the performance of TO agents.

5.4 Densely Distributed POIs

Our fourth setup challenges agents' both exploratory and exploitative skills by requiring agents to learn to visit as many POIs as possible. They should not do this greedily, but more efficiently. We do not expect them to learn to observe all of the POIs, due to time limitation. Hence, we show the normalized scores of teams of agents. The environment has 3 layers of POIs on a 60°chord of a circle. The closest chord to the center has 3 POIs with the value of 2, the middle one has 3 POIs valuing 5, and the farthest one has another 3 that worth 10 each. Because POIs are distributed on three chords, as they move away from the center, the distance between the POIs of every chord increases. Therefore, we expect agents to not observe the all of the POIs on the low valued chords, but to observe a POI with a higher value. This setup allows agents to learn from a larger policy space that results in non-zero team fitness, because a team can learn to observe various combinations of POIs.



(a) Scaling over Task Complexity

(b) Coupling 5 - Task-Oriented Agents

(c) Coupling 5 - Novelty Seeking Agents

Figure 3: Scaling of Novelty seeking (NS) and Task-Oriented (TO) Agents over different numbers of agents and coupling. Performances (Fig. 3a) in second experimental setup - 6 randomly distributed POIs on two circles on a 20x20 map, the number of agents increases as the coupling requirement increases and the episode length is 40. POIs located on the inner circle have the value of 2, and the outer POIs value 5. Heatmaps represent how the map is explored during 300 generations of EA's training when there are 15 agents and coupling is 5. Dashed white circles represent the circles where POIs are on. Exploration by TO agents is on Fig. 3b, by NS agents is provided on Fig. 3c

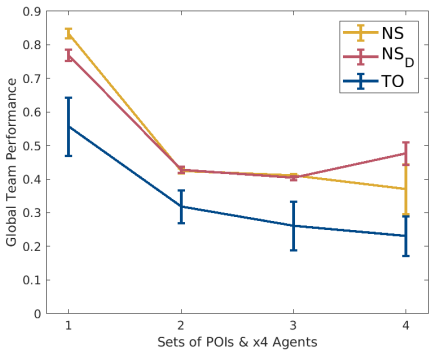


Figure 4: Performances achieved after 300 generations of EA's training under the third settings. A set of POIs represents a low valued POI and an accordingly positioned high valued POI. The degree of coupling is constant, 4, but we have varying numbers of agents, 4, 8, 12, 16 and sets of POIs, 1, 2, 3, 4 on a 20x20 map and duration is 40 time-steps.

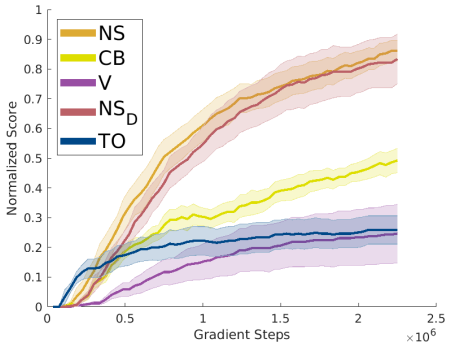
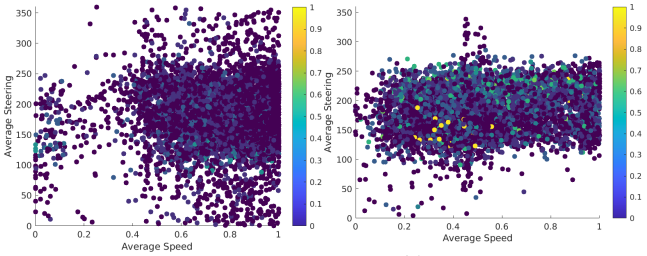


Figure 5: Performances achieved after 300 generations of EA in the forth experimental setup - 8 agents, Densely Distributed 9 POIs on three layers and the episode length is 50. The inner-most layer has the POIs with the lowest values, the outer-most layer has the highest values on a 20x20 map.

We test agents against a high coupling factor and Figure 5 shows that both NS and NS_D agents achieve similar performances. TO agents achieves much less than our agents.

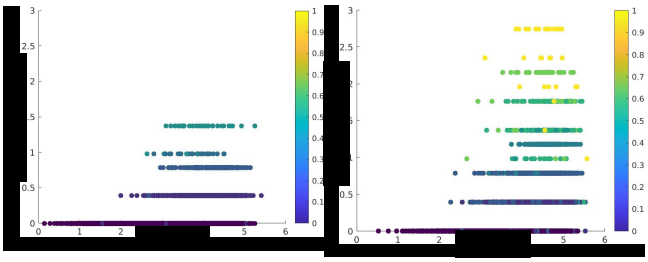
We also test the effect of the different components of our novelty-seeking fitness function. When we train V agents solely on the heuristic fitness, value of saliency (only V of Eq. 2), EA learns slower, but it finally achieves a score similar to TO agents. Pure count-based (CB) bonus (f_{NS} of Eq. 1) is certainly a fitness resulting in better search, but we see that NS agents experiencing salient events via their fitnesses (f_{salient} of Eq. 2), achieves much better.

In this paper, we are not generating diversity by searching on a behavior map that has a lower dimension than the agents' actual behavior-space, but we represent behaviors generated on two different behavior spaces to understand what kind of behaviors our agents generate that are different than TO agents. We expect NS agents to search in a larger part of behavior space than the TO agents do during the EA's training. To visualize the behavior diversity, we represent behaviors in a 2D plot (Figure 6). The work [13] scatters a behavior via the average speed and the observation radius of agents. However, in the traditional settings of the multi-rover domain, observation radius stays constant, but we expect agents to maneuver to observe POIs on different chords. Therefore, we adopt their average speed axis and replace average observation radius with average steering. We do rollouts for each behavior in the population of every generation. All rollouts are conducted in the same configuration of the environmental setup. NS agents cover a wider portion of the behavior-space, while TO agents are concentrated on the high-speed region. Also, behaviors generated within the teams with higher fitness are mostly distributed in the lower average speed region, but TO agents do not cover those areas. Particularly, we expect agents to generate behaviors on a wider average speed



(a) Task-Oriented (b) Novelty Seeking

Figure 6: Behavior spaces covered during the 300 generations of EA's training. There are 8 agents and coupling factor is 7. Each behavior is represented by the average speed and steering by a single agent throughout an episode. Covered space by task-oriented (TO) agents is on the *Left*, by novelty seeking (NS) agents is provided on the *Right*. The color gradient represents the team fitness of a behavior's episode



(a) Task-Oriented (b) Novelty Seeking

Figure 7: Analysis of the behaviors (in Figure 6). Each behavior is represented by the cumulative contribution to its team and entropy representing the state diversity covered throughout an episode. Generated behaviors by task-oriented (TO) agents is on the *Left*, by novelty seeking (NS) agents is provided on the *Right*. The color gradient represents the team fitness of a behavior's episode

scale, because of our mathematical implementation of the fitness function. We combine the value of saliency with our exploration bonus (Eq. 2) that enables agents to continuously explore. Ergo, agents receive feedback according to the ratio between the time experiencing a saliency and the value of that saliency.

Additionally, we analyze generated behaviors through a novel behavior diversity representation that we present in this paper. We utilize DEFs as a measure to analyze how a behavior contributes to its team's success. Agents are not necessarily provided DEFs as their fitness, but we only use it as a measure here. On the other axes, we measure the diversity of the states visited during the generation of a behavior. This diversity is measured via Shannon entropy [38] and we use the distributions of quantized states. Figure 7 represents the behaviors with these two measures and the team fitness values (color gradient). Self-evidently, behaviors that have higher contribution to their teams' success are mostly present in the teams with higher fitnesses. A key fact is that behaviors having higher contribution to their teams visit more distinct states.

6 DISCUSSION

Based on these results, we can answer the questions raised in Section 4. Our experiments show that the coupling factor has the largest effect on the team behavior of agents. In our first two sets of experiments we varied the coupling factor. The first setup (Figure 2) has the coupling factor as the only changing parameter. Because coupling requires agents to be present at certain states simultaneously, as the coupling increases the global task gets much more difficult to solve unless we add more agents to the system. We see that TO (greedy) agents do well when tasks are simple; however, they are affected more dramatically than NS agents by the increase in this parameter. When we introduce more agents to the system, even when systems gets more tightly-coupled, the performance of the NS agents is much more robust than the TO agents'(Figure 3).

We argue that the greater robustness of the NS agents to larger tasks and higher coupling factors stems from their ability to provide near uniform coverage of the space of POIs (Figure 3). For higher coupling factors, coverage of the task space increases the likelihood that a sufficient number of agents will visit each POI simultaneously at some point during an episode. Without this coverage, agents trained with task-oriented local fitness are prone to concentrating on a small number of POIs, and missing out on the opportunity to observe other POIs as well.

In environments with higher-dimensional observation spaces, we need methods to approximate the visitation counts used to compute the novelty-seeking fitness(e.g. [4, 7, 39]). In Figure 2 we evaluate MERL with a Never-Give-Up [4] style pseudocount estimator, but found that it does not preserve the performance of the discretized count estimate for larger coupling factors. Finally, our results show that while incorporating a DEF into the local fitness function helps in simple scenarios, its effect is minimal, and diminishes as the task becomes more complex. This likely reflects the sparsity of the DEF signal in more tightly-coupled tasks.

7 CONCLUSION AND FUTURE WORK

We proposed a general and principled approach to novelty-search in MAS that balances exploration at agent-level with exploitation at team-level. Our fitness structure enables teams to perform well even with large numbers of agents and high coupling factors. We argue that being exploration-driven at team level and experiencing saliency increases team performance significantly in tightly-coupled systems. When the degree of coupling between the agents is low, greedy individual behavior may lead to better performance in simple tasks, but as we increase task complexity, these task-oriented individual policies fail to sufficiently explore the joint-strategy space in the way that our novelty-seeking agents do. In these more complex settings our approach is therefore superior.

As a direction for future work, we suggest that an embedding network (like that used in NGU [4]) can be trained via novelty-oriented fitnesses, so that it can be incorporated into our framework for tightly-coupled MAS with high-dimensional state-spaces.

ACKNOWLEDGMENTS

This work was partially supported by the National Science Foundation grant IIS-1815886 and the Air Force Office of Scientific Research grant No. FA9550-19-1-0195.

REFERENCES

[1] Adrian Agogino, Chris HolmesParker, and Kagan Tumer. 2012. Evolving large scale UAV communication system. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. 1023–1030.

[2] Adrian K Agogino and Kagan Tumer. 2004. Unifying temporal and structural credit assignment problems. In *Autonomous Agents and Multi-Agent Systems Conference*.

[3] Ayhan Alp Aydeniz, Anna Nickelson, and Kagan Tumer. 2022. Entropy-based local fitnesses for evolutionary multiagent systems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 212–215.

[4] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, et al. 2020. Never give up: Learning directed exploration strategies. *arXiv preprint arXiv:2002.06038* (2020).

[5] Andrew G Barto and Ozgür Simsek. 2005. Intrinsic motivation for reinforcement learning systems. In *Proceedings of the Thirteenth Yale Workshop on Adaptive and Learning Systems*. Citeseer, 113–118.

[6] Andrew G Barto, Satinder Singh, Nuttapong Chentanez, et al. 2004. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*. Piscataway, NJ, 112–19.

[7] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. 2016. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

[8] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. 2021. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 13550–13563.

[9] Jen Jen Chung, Carrie Rebhuhn, Connor Yates, Geoffrey A Hollinger, and Kagan Tumer. 2019. A multiagent framework for learning dynamic traffic management strategies. *Autonomous Robots* 43 (2019), 1375–1391.

[10] Jeffrey S Cox and Edmund H Durfee. 2005. An efficient algorithm for multiagent plan coordination. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. 828–835.

[11] Antoine Cully. 2019. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 81–89.

[12] Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer. 2014. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*. 165–172.

[13] Gaurav Dixit and Kagan Tumer. 2022. Balancing teams with quality-diversity for heterogeneous multiagent coordination. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 236–239.

[14] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2018. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070* (2018).

[15] Iztok Fister, Andres Iglesias, Akemi Galvez, Javier Del Ser, Eneko Osaba, and Iztok Fister. 2018. Using novelty search in differential evolution. In *Highlights of Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection: International Workshops of PAAMS 2018, Toledo, Spain, June 20–22, 2018, Proceedings 16*. Springer, 534–542.

[16] David B Fogel. 2006. *Evolutionary computation: toward a new philosophy of machine intelligence*. Vol. 1. John Wiley & Sons.

[17] Justin Fu, John Co-Reyes, and Sergey Levine. 2017. Ex2: Exploration with exemplar models for deep reinforcement learning. *Advances in neural information processing systems* 30 (2017).

[18] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[19] Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. 2017. Novelty-driven cooperative coevolution. *Evolutionary computation* 25, 2 (2017), 275–307.

[20] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*. PMLR, 1861–1870.

[21] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905* (2018).

[22] Shivaram Kalyanakrishnan and Peter Stone. 2009. Learning complementary multiagent behaviors: A case study. In *Robot Soccer World Cup*. Springer, 153–165.

[23] Shauharda Khadka and Kagan Tumer. 2018. Evolution-guided policy gradient in reinforcement learning. *Advances in Neural Information Processing Systems* 31 (2018).

[24] Xiao Li, Cristian-Ioan Vasile, and Calin Belta. 2017. Reinforcement learning with temporal logic rewards. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3834–3839.

[25] Adam Lipowski and Dorota Lipowska. 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications* 391, 6 (2012), 2193–2196.

[26] Jyi-Shane Liu and Katia P Sycara. 1996. Multiagent coordination in tightly coupled task scheduling. In *Proceedings of the Second International Conference on Multi-Agent Systems*. 181–188.

[27] Robert Loftin, Aadirupa Saha, Sam Devlin, and Katja Hofmann. 2021. Strategically efficient exploration in competitive multi-agent reinforcement learning. In *Uncertainty in Artificial Intelligence*. PMLR, 1587–1596.

[28] Hamid Maei, Csaba Szepesvari, Shalabh Bhatnagar, Doina Precup, David Silver, and Richard S Sutton. 2009. Convergent temporal-difference learning with arbitrary smooth function approximation. *Advances in neural information processing systems* 22 (2009).

[29] Somdeb Majumdar, Shauharda Khadka, Santiago Miret, Stephen McAleer, and Kagan Tumer. 2020. Evolutionary reinforcement learning for sample-efficient multiagent coordination. In *International Conference on Machine Learning*. PMLR, 6651–6660.

[30] Maja J Mataric. 1998. Using communication to reduce locality in distributed multiagent learning. *Journal of experimental & theoretical artificial intelligence* 10, 3 (1998), 357–369.

[31] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

[32] MohammadJavad NoroozOliaee, Bechir Hamdaoui, and Kagan Tumer. 2012. Efficient objective functions for coordinated learning in large-scale distributed osa systems. *IEEE Transactions on Mobile Computing* 12, 5 (2012), 931–944.

[33] Frans A Oliehoek and Christopher Amato. 2016. *A concise introduction to decentralized POMDPs*. Springer.

[34] Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. 2017. Count-based exploration with neural density models. In *International conference on machine learning*. PMLR, 2721–2730.

[35] Aloïs Pourchot and Olivier Sigaud. 2018. CEM-RL: Combining evolutionary and gradient-based methods for policy search. *arXiv preprint arXiv:1810.01222* (2018).

[36] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI* (2016), 40.

[37] Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. 2016. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 4424–4429.

[38] Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal* 27, 3 (1948), 379–423.

[39] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. 2017. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems* 30 (2017).

[40] Kagan Tumer, Adrian K Agogino, and David H Wolpert. 2002. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. 378–385.

[41] Kagan Tumer, Zachary T Welch, and Adrian Agogino. 2008. Aligning social welfare and agent preferences to alleviate traffic congestion. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*. Citeseer, 655–662.

[42] Matteo Vasirani and Sascha Ossowski. 2009. A market-inspired approach to reservation-based urban road traffic management. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. Citeseer, 617–624.

[43] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. 2008. Maximum entropy inverse reinforcement learning.. In *Aaai*, Vol. 8. Chicago, IL, USA, 1433–1438.