Coordinated Exploration via Intrinsic Rewards for Multi-Agent Reinforcement Learning

Shariq Iqbal ¹ Fei Sha ¹²

Abstract

Solving tasks with sparse rewards is one of the most important challenges in reinforcement learning. In the single-agent setting, this challenge is addressed by introducing intrinsic rewards that motivate agents to explore unseen regions of their state spaces; however, applying these techniques naively to the multi-agent setting results in agents exploring independently, without any coordination among themselves. We argue that exploration in cooperative multi-agent settings can be accelerated and improved if agents coordinate with respect to the regions of the state space they explore. In this paper we propose an approach for learning how to dynamically select between proposed intrinsic reward types which consider not just what an individual agent has explored, but all agents, such that the agents can coordinate their exploration and maximize extrinsic returns. Concretely, we formulate the approach as a hierarchical policy where a high-level controller selects among sets of policies trained on diverse intrinsic rewards and the low-level controllers learn the action policies of all agents under these specific rewards. We demonstrate the effectiveness of the proposed approach in a multi-agent GridWorld domain with sparse rewards and then show that our method scales up to more complex settings by evaluating on the VizDoom (Kempka et al., 2016) platform.

1. Introduction

Recent work in deep reinforcement learning effectively tackles challenging problems including the board game Go (Silver et al., 2016), Atari video games (Mnih et al., 2015), and simulated robotic continuous control (Lillicrap et al., 2016); however, these successful approaches often rely on frequent feedback indicating whether the learning agent is performing well, a setting known as dense rewards. In many tasks, however, dense rewards are difficult to specify without inducing locally optimal but globally sub-optimal behavior. As such, it is frequently desirable to specify only a sparse reward that simply signals whether an agent has succeeded or failed on a given task.

Nonetheless, sparse rewards introduce their own set of challenges. Determining which of an agent's actions led to a reward becomes more difficult, a phenomenon known in reinforcement learning as the credit-assignment problem. Furthermore, if rewards cannot be obtained by random actions, an agent will never receive a signal through which it can begin learning. As such, researchers have devised methods which attempt to provide agents with additional reward signals, known as intrinsic rewards, through which they can learn meaningful behavior (Oudeyer & Kaplan, 2009). A large subset of these works focus on learning intrinsic rewards that encourage exploration of the state space (Pathak et al., 2017; Houthooft et al., 2016; Burda et al., 2019; Ostrovski et al., 2017; Tang et al., 2017).

While existing work largely focuses on designing intrinsic reward functions for single agents, our work approaches the problem of designing intrinsic reward functions in the cooperative multi-agent setting, assuming that each agent has its own function that detects the novelty of their observations. In many cases, the default approach – exploring state space independently – may not be efficient. For example, consider a search-and-rescue task where multiple agents need to collectively find all missing persons spread throughout the environment. It would be inefficient for the agents to explore the same areas redundantly. Instead, it would be much more sensible for agents to "divide-and-conquer" or avoid redundant exploration. Thus, an ideal intrinsic reward for this task would encourage such behavior; however, the same behavior would not be ideal for other tasks. In some tasks agents may receive more rewards by exploring the same areas, for example when traveling in groups is safer than spreading out. Cooperative multi-agent reinforcement learning can benefit from sharing information about exploration across agents; however, what to do with that shared information needs to be adaptive to the task at hand and

¹Department of Computer Science, University of Southern California, Los Angeles, California ²Google AI, Mountain View, California. Correspondence to: Shariq Iqbal <shariqiq@usc.edu>.

exploration often needs to be coordinated.

In this paper, we present a method for learning both lowlevel policies trained on different intrinsic rewards which emphasize coordination of exploration and a meta-policy for selecting the policies which maximizes extrinsic rewards on a given task. Importantly, we learn the policies simultaneously using a shared replay buffer with off-policy methods, drastically improving sample efficiency. Moreover, the meta-policy is learned in conjunction with those lowlevel policies, effectively exploring different types of "lowlevel" exploration types. We show empirically, in both a GridWorld domain as well as in the more complex ViZ-Doom (Kempka et al., 2016) setting: first, intrinsic reward functions which coordinate across agents can be more effective than naive intrinsically motivated exploration, and finally, our approach is able to match or exceed the performance of the best coordinated intrinsic reward function (which differs across tasks) while using no more samples.

2. Related Work

Single-Agent Exploration In order to solve sparse reward problems, researchers have long worked on improving exploration in reinforcement learning. Prior works commonly propose reward bonuses that encourage agents to reach novel states. In tabular domains, reward bonuses based on the inverse state-action count have been shown to be effective in speeding up learning (Strehl & Littman, 2008). In order to scale count-based approaches to large state spaces, many recent works have focused on devising pseudo state counts to use as reward bonuses (Bellemare et al., 2016; Ostrovski et al., 2017; Tang et al., 2017). Alternatively, some work has focused on defining intrinsic rewards for exploration based on inspiration from psychology (Oudever & Kaplan, 2009; Schmidhuber, 2010). These works use various measures of novelty as intrinsic rewards including: transition dynamics prediction error (Pathak et al., 2017), information gain with respect to a learned dynamics model (Houthooft et al., 2016), and random state embedding network distillation error (Burda et al., 2019).

Multi-Agent Reinforcement Learning (MARL) Multiagent reinforcement learning introduces several unique challenges that recent work attempts to address. These challenges include: multi-agent credit assignment in cooperative tasks with shared rewards (Sunehag et al., 2018; Rashid et al., 2018; Foerster et al., 2018), non-stationarity of the environment in the presence of other learning agents (Lowe et al., 2017; Foerster et al., 2018; Iqbal & Sha, 2019), and learning of communication protocols between cooperative agents (Foerster et al., 2016; Sukhbaatar et al., 2016; Jiang & Lu, 2018).

Exploration in MARL While the fields of exploration in RL and multi-agent RL are popular, relatively little work has

been done at the intersection of both. Carmel & Markovitch (1997) consider exploration with respect to opponent strategies in competitive games, and Verbeeck et al. (2005) consider exploration of a large joint action space in a load balancing problem. Jaques et al. (2018) define an intrinsic reward function for multi-agent reinforcement learning that encourages agents to take actions which have the biggest effect on other agents' behavior, otherwise referred to as "social influence". Agogino & Tumer (2008) Define metrics for evaluating the efficacy of reward functions in multi-agent domains. These works, while important, do not address the problem of coordinating exploration in a large state space among multiple agents. A recent approach to collaborative evolutionary reinforcement learning (Khadka et al., 2019) shares some similarities with our approach. As in our work, the authors devise a method for learning a population of diverse policies with a shared replay buffer and dynamically selecting the best learner; however, their work is focused on single-agent tasks and does not incorporate any notion of intrinsic rewards. As such, this work is not applicable to sparse reward problems in MARL. Most recently, Wang et al. (2020) define influence-based rewards which encourage agents to visit regions where their actions influence other agents' transitions and rewards (e.g. one agent unlocks a door for another). In practice, they combine their approach with state-based exploration, and thus, their work is complementary to our approach where agents do not simply explore regions that are novel to them but take into account how novel all other agents consider these regions.

3. Background

Dec-POMDPs In this work, we consider the setting of decentralized POMDPs (Oliehoek et al., 2016), which are used to describe cooperative multi-agent tasks. A decentralized POMDP (Dec-POMDP) is defined by a tuple: $(\mathbf{S}, \mathbf{A}, T, \mathbf{O}, O, R, n, \gamma)$. In this setting we have n total agents. S is the set of global states in the environment, while $\mathbf{O} = \bigotimes_{i \in \{1...n\}} \mathbf{O}_i$ is the set of joint observations for each agent and $\mathbf{A} = \bigotimes_{i \in \{1...n\}} \mathbf{A}_i$ is the set of possible joint actions for each agent. A specific joint action at one time step is denoted as $\mathbf{a} = \{a_1, \dots, a_n\} \in \mathbf{A}$ and a joint observation is $\mathbf{o} = \{o_1, \dots, o_n\} \in \mathbf{O}$. T is the state transition function which defines the probability $P(s'|s, \mathbf{a})$, and O is the observation function which defines the probability $P(\mathbf{o}|\mathbf{a},s')$. R is the reward function which maps the combination of state and joint actions to a single scalar reward. Importantly, this reward is shared between all agents, so Dec-POMDPs always describe cooperative problems. Finally, γ is the discount factor which determines how much the agents should favor immediate reward over long-term gain. We specifically consider a subclass of Dec-POMDPs known as factored n-agent MDPs (Oliehoek et al., 2016), where the state space can be factored into per-agent "local

states" $\mathbf{S} = \bigotimes_{i \in \{1...n\}} \mathbf{S}_i$. Importantly, we assume that the local state spaces of all agents, \mathbf{S}_i , are identical (i.e. the agents operate within a shared space).

Soft Actor-Critic Our approach uses Soft Actor-Critic (SAC) (Haarnoja et al., 2018) as its underlying algorithm. SAC incorporates an entropy term in the loss functions for both the actor and critic, in order to encourage exploration and prevent premature convergence to a sub-optimal deterministic policy. The policy gradient with an entropy term is computed as follows:

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{s \sim D, a \sim \pi} \left[\nabla_{\theta} \log \pi_{\theta}(a|s) \left(-\frac{\log \pi_{\theta}(a|s)}{\alpha} + Q_{\psi}(s, a) - b(s) \right) \right]$$
(1)

where D is a replay buffer that stores past environment transitions, ψ are the parameters of the learned critic, b(s) is a state dependent baseline (e.g. the state value function V(s)), and α is a reward scale parameter determining the amount of entropy in an optimal policy. The critic is learned with the following loss function:

$$\mathcal{L}_Q(\psi) = \mathbb{E}_{(s,a,r,s') \sim D} \left[(Q_{\psi}(s,a) - y)^2 \right]$$
 (2)

$$y = r(s, a) +$$

$$\gamma \mathbb{E}_{a' \sim \pi(s')} \left[Q_{\bar{\psi}}(s', a') - \frac{\log(\pi_{\bar{\theta}}(a'|s'))}{\alpha} \right]$$
(3)

where $\bar{\psi}$ are the parameters of the target critic which is an exponential moving average of the past critics, updated as: $\bar{\psi} \leftarrow (1-\tau)\bar{\psi} + \tau \psi$, and τ is a hyperparameter that controls the update rate.

Centralized Training with Decentralized Execution A number of works in deep multi-agent reinforcement learning have followed the paradigm of centralized training with decentralized execution (Lowe et al., 2017; Foerster et al., 2018; Sunehag et al., 2018; Rashid et al., 2018; Iqbal & Sha, 2019). This paradigm allows for agents to train while sharing information (or incorporating information that is unavailable at test time) but act using only local information, without requiring communication which may be costly at execution time. Since most reinforcement learning applications use simulation for training, communication between agents during the training phase has a relatively low cost.

4. Intrinsic Rewards for MARL Exploration

In this section we describe how to design intrinsic reward functions for exploration that are specifically tailored to multi-agent learning. The main idea is to share whether other agents have explored a region and consider it as novel. We assume that each agent (indexed by i) has a novelty function $f_i: \mathbf{O}_i \to \mathbb{R}^+$ that determines how novel an observation is to it, based on its past experience. This function can be an inverse state visit count in discrete domains, or, in large/continuous domains, it can be represented by recent approaches for developing novelty-based intrinsic rewards in complex domains, such as random network distillation (Burda et al., 2019). We assume that all agents share the same observation space so that each agent's novelty function can operate on all other agents' observations.

We define the multi-agent intrinsic reward function $g_i(\cdot)$ for agent i, which considers how novel *all* agents consider i's observation. Concretely, the function maps the vector $[f_1(o_i), \dots, f_n(o_i)]$ to a scalar reward.

Desiderata of $g_i(\cdot)$ While in theory $g_i(\cdot)$ can be in any form, we believe the following two properties are intuitive and naturally applicable to cooperative MARL:

- Coordinate-wise Monotonicity An observation becoming less novel to any individual agent should *not increase* the intrinsic reward, preventing the agent from exploring a region more as it becomes more known to other agents. Formally, $\partial g_i/\partial f_j \geq 0, \forall i,j$.
- Inner-directedness If an observation approaches having zero novelty to an agent, then the intrinsic reward should also approach zero, irrespective of other agents' novelty. This prevents the agent from repetitively exploring the same regions, at the "persuasion" of other agents.

Examples Fig. 1 visualizes a few examples of intrinsic rewards that observe the aforementioned desirable properties. INDEPENDENT rewards are analagous to single-agent approaches to exploration which define the intrinsic reward for an agent as the novelty of their own observation that occurs as a result of an action. The remainder of intrinsic reward functions that we consider use the novelty functions of other agents, in addition to their own, to further inform their exploration.

MINIMUM rewards consider how novel all agents find a specific agent's observation and rewards that agent based on the minimum. This method leads to agents only being rewarded for exploring areas that no other agent has explored, which could be advantageous in scenarios where redundancy in exploration is not useful or even harmful.

COVERING rewards agents for exploring areas that it considers more novel than the average agent. This reward results in agents shifting around the state space, only exploring regions as long as they are more novel to them than their average teammate.

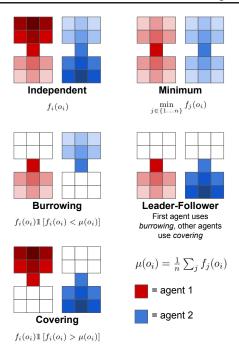


Figure 1: Multi-agent intrinsic rewards. Visualized for the 2 agent case. INDEPENDENT shows the regions that have been explored by each agent. Darker shades mean higher reward values.

BURROWING rewards do the opposite, only rewarding agents for exploring areas that it considers less novel than the average agent. While seemingly counterintuitive, these rewards encourage agents to further explore areas they have already explored with the hope that they will discover new regions that few or no other agents have seen, which they will then consider less novel than average and continue to explore. As such, these rewards result in agents continuing to explore until they exhaust all possible intrinsic rewards from a given region (i.e. hit a dead end), somewhat akin to a depth-first search.

LEADER-FOLLOWER uses burrowing rewards for the first agent, and covering rewards for the rest of the agents. This leads to an agent exploring a space thoroughly, and the rest of the agents following along and trying to cover that space.

Note that these are not meant to be a comprehensive set of intrinsic reward functions applicable to all cooperative multiagent tasks. In fact, any nonnegative combination of those rewards is a reward that is consistent with the desiderata. Our focus instead is to use a "primitive" basis of functions and show that these functions induce coordinated exploration. Then, we show that our method, detailed in the following section, is able to match or exceed the performance of the best individual reward function while using the same number of samples.

5. Learning for Multi-Agent Exploration

For many tasks, it is impossible to know a priori which intrinsic rewards will be the most helpful one. Furthermore, the type of reward that is most helpful could change over the course of training if the task is sufficiently complex. In this section we present our approach for simultaneously learning policies trained with different types of intrinsic rewards and dynamically selecting the best one.

Simultaneous Policy Learning In order to learn policies for various types of intrinsic rewards in parallel, we utilize a shared replay buffer and off-policy learning to maximize sample efficiency. In other words, we learn policies and value functions for all intrinsic reward types from all collected data, regardless of which policies it was collected by. This parallel learning is made possible by the fact that we can compute our novelty functions off-policy, given the observations for each agent after each environment transition, which are saved in a replay buffer. For each type of reward, we learn a different "head" for our policies and critics. In other words, we learn a single network for each agent's set of policies that shares early layers and branches out into different heads for each reward type. For critics, we learn a single network across all agents that shares early layers and branches out into separate heads for each agent and reward type. We learn separate heads for intrinsic and extrinsic rewards, as in Burda et al. (2019). We provide a diagram of our model architecture in Figure 2. Our specific learning algorithm is adapted from the multi-agent Soft-Actor-Critic method presented in Iqbal & Sha (2019).

We index agents by $i \in \{1 \dots n\}$ and intrinsic reward types by $j \in \{1 \dots m\}$ where m is the total number of intrinsic reward types that we are considering. The policy for agent i, trained using reward j (in addition to extrinsic rewards), is represented by π_i^j . It takes as input agent i's observation, o_i , and outputs a distribution from which we can sample the action a_i . The parameters of this policy are $\Theta_i^j = \{\theta_i^{\text{share}}, \theta_i^j\}$, where θ_i^{share} is a shared base/input (for agent i) in a neural network and θ_i^j is a head/output specific to this reward type.

The extrinsic critic for policy head π_i^j is represented by $Q_{i,j}^{\mathrm{ex}}$. It takes as input the global state s and the actions of all other agents $\mathbf{a}_{\backslash i}$, and it outputs the expected returns under policy π_i^j for each possible action that agent i can take, given all other agents' actions. The parameters of this critic are $\Psi_{i,j}^{\mathrm{ex}} = \{\psi^{\mathrm{share}}, \psi_{i,j}^{\mathrm{ex}}\}$ where ψ^{share} is a shared base across all agents and reward types. A critic with similar structure exists for predicting the intrinsic returns of actions taken by π_i^j , represented by $Q_{i,j}^{\mathrm{in}}$, which uses the parameters: $\Psi_{i,j}^{\mathrm{in}} = \{\psi^{\mathrm{share}}, \psi_{i,j}^{\mathrm{in}}\}$. Note that the intrinsic critics share the same base parameters ψ^{share} .

We remove the symbols representing the parameters of the

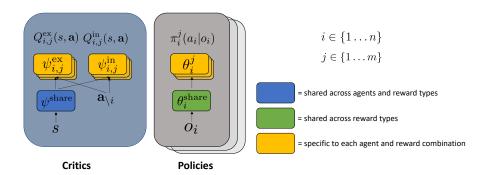


Figure 2: Diagram of our model architecture, showing how parameters for actors and critics are shared. i indexes agents, while j indexes reward types. Pseudo-code for the full network architecture is provided in the supplementary material.

policies (Θ) and the critics (Ψ) for readability. In our notation we use the absence of a subscript or superscript to refer to a group. For example π^j , refers to *all* agents' policies trained on intrinsic reward j. We train our critics with the soft actor-critic Q-function loss (Equation 2), using the following targets for the extrinsic and intrinsic critics:

$$\begin{split} y_{i,j}^{\text{ex}} &= r^{\text{ex}}(s, \mathbf{a}) + \\ &\gamma \mathbb{E}_{\mathbf{a}' \sim \bar{\pi}^{j}(\mathbf{o}')} \left[\bar{Q}_{i,j}^{\text{ex}}(s', \mathbf{a}') - \frac{\log(\bar{\pi}_{i}^{j}(a'_{i}|o'_{i}))}{\alpha} \right] \quad (4) \\ y_{i,j}^{\text{in}} &= r_{i,j}^{\text{in}}(o'_{i}) + \\ &\gamma \mathbb{E}_{\mathbf{a}' \sim \bar{\pi}^{j}(\mathbf{o}')} \left[\bar{Q}_{i,j}^{\text{in}}(s', \mathbf{a}') - \frac{\log(\bar{\pi}_{i}^{j}(a'_{i}|o'_{i}))}{\alpha} \right] \quad (5) \end{split}$$

where \bar{Q} refers to the target Q-function, an exponential weighted average of the past Q-functions, used for stability, and $\bar{\pi}$ are similarly updated target policies. The intrinsic rewards laid out in Figure 1 are represented as a function of the observations that results from the action taken, $r_{i,j}^{\rm in}(o_i')$ where j specifies the type of reward. Importantly, we can calculate these loss functions for expected intrinsic and extrinsic returns for all policies given a single environment transition, allowing us to learn multiple policies for each agent in parallel. We train each policy head with the following gradient:

$$\nabla_{\boldsymbol{\Theta}_{i}^{j}} J(\boldsymbol{\pi}_{i}^{j}) = \mathbb{E}_{(s,\mathbf{o}) \sim D, \mathbf{a} \sim \boldsymbol{\pi}^{j}} \left[\nabla_{\boldsymbol{\Theta}_{i}^{j}} \log \boldsymbol{\pi}_{i}^{j}(a_{i} | o_{i}) \right]$$

$$\left(-\frac{\log \boldsymbol{\pi}_{i}^{j}(a_{i} | o_{i})}{\alpha} + A_{i}^{j}(s, \mathbf{a}) \right)$$

$$A_{i}^{j}(s, \mathbf{a}) = Q_{i,j}^{\text{ex}}(s, \mathbf{a}) + \beta Q_{i,j}^{\text{in}}(s, \mathbf{a}) - V_{i}^{j}(s, \mathbf{a}_{\backslash i})$$

$$V_{i}^{j}(s, \mathbf{a}_{\backslash i}) = \sum_{a_{i}^{\prime} \in \mathbf{A}_{i}} \boldsymbol{\pi}_{i}^{j}(a_{i}^{\prime} | o_{i}) (Q_{i,j}^{\text{ex}}(s, \{a_{i}^{\prime}, \mathbf{a}_{\backslash i}\}) + \beta Q_{i,j}^{\text{in}}(s, \{a_{i}^{\prime}, \mathbf{a}_{\backslash i}\}))$$

$$\beta Q_{i,j}^{\text{in}}(s, \{a_{i}^{\prime}, \mathbf{a}_{\backslash i}\})$$

$$(8)$$

where β is a scalar that determines the weight of the intrinsic rewards, relative to extrinsic rewards, and A_i^j is a multiagent advantage function (Foerster et al., 2018; Iqbal & Sha, 2019), used for helping with multi-agent credit assignment.

Dynamic Policy Selection Now that we have established a method for simultaneously learning policies using different intrinsic reward types, we must devise a means of selecting between these policies. In order to select policies to use for environment rollouts, we must consider which policies maximize extrinsic returns, while taking into account the fact that there may still be "unknown unknowns," or regions that the agents have not seen yet where they may be able to further increase their extrinsic returns. As such, we must learn a meta-policy that, at the beginning of each episode, selects between the different sets of policies trained on different intrinsic rewards and maximizes extrinsic returns without collapsing to a single set of policies too early. We parameterize the selector policy Π with a vector, ϕ , that contains an entry for every reward type. The probability of sampling head j is: $\Pi(j) \propto \exp(\phi[j])$. Unlike the action policies, this high-level policy does not take any inputs, as we simply want to learn which set of policies trained on the individual intrinsic reward functions has the highest expected extrinsic returns from the beginning of the episode.

The most sensible metric for selecting policies is the expected extrinsic returns given by each policy head. We can use policy gradients to train the policy selector, Π , to maximize this value using the returns received when performing rollouts in the environment. We use the following gradient to train Π :

$$\nabla_{\phi} J(\Pi) = \mathbb{E}_{h \sim \Pi} \left[\nabla_{\phi} \log \Pi(h) - \left(-\frac{\log \Pi(h)}{\eta} + R_h^{\text{ex}} - b_{\Pi} \right) \right]$$
(9)
$$R_h^{\text{ex}} = \sum_{t=0}^{T} \gamma^t r^{\text{ex}}(s_t, \mathbf{a}_t) |\mathbf{a} \sim \pi^h(\mathbf{o}_t),$$

$$b_{\Pi} = \sum_{h'}^{m} \Pi(h') \mu_{h'}$$
(10)

where μ_h is a running mean of the returns received by head h in the past, and η is a parameter similar to α for the

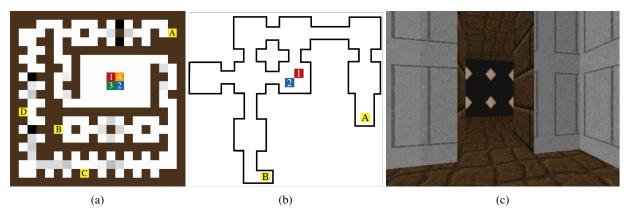


Figure 3: (Left) Rendering of our gridworld domain. Agents start each episode in the central room and must complete various tasks related to collecting the yellow treasures placed around the map. (Center) Top-Down view of VizDoom "My Way Home" map, modified for multi-agent experiments (Right) Egocentric view in VizDoom used for agents' observations

low-level policies, which promotes entropy in the selector policy. Entropy in the policy selector is important in order to prevent it from collapsing onto a single exploration type that does well at first but does not continue to explore as effectively as others.

6. Experiments

We begin by describing our evaluation domains and then present experimental results which demonstrate the effectiveness of our approach. We provide additional details in the supplementary material as well as code for both the algorithm and environments.

Domains for testing single-agent exploration typically revolve around navigation of an environment with sparsely distributed rewards (e.g. Montezuma's Revenge (Ostrovski et al., 2017; Tang et al., 2017; Burda et al., 2019), Viz-Doom (Pathak et al., 2017), etc). In the multi-agent setting, we define several tasks which also consider navigation with very sparse rewards, while requiring coordination across agents. Our tasks involve collecting the items spread around a map (displayed in yellow in Figure 3): TASK 1 Agents must cooperatively collect all treasure on the map in order to complete the task. Ideally, agents would spread out in order to solve the task effectively. TASK 2 Agents must all collect the same treasure. In this case agents would ideally explore similar regions concurrently. TASK 3 Agents must all collect the specific treasure assigned to them, requiring no coordination across agents. The two agent version of each task uses agents 1-2 and treasure A-B, while the three agent versions use 1-3, A-C, and the four agent versions use 1-4, A-D. Agents receive a negative time penalty towards extrinsic rewards at each step, so they are motivated to complete the task as quickly as possible. The only positive extrinsic reward comes from any agent collecting a treasure allowed by the specific task, and rewards are shared between all agents.

6.1. Gridworld Domain

We first test our approach using a multi-agent gridworld domain (pictured in Fig. 3a), which allows us to design environments where the primary challenge lies in a combination of exploring the state space efficiently and coordinating behaviors, without the added difficulty of learning in a high-dimensional space.

The environment includes two sources of stochasticity: random transitions and wormholes. At each step there is a 10% chance of an agent's action being replaced by a random one. Furthermore, there are several "wormholes" placed around the map which have a probability of opening at each time step. This probability, which agents can observe, changes at each step using a biased random walk such that it moves toward one, until the hole opens and it resets to zero. If an agent steps into an open wormhole, they will be sent back to their starting position. The spaces colored as black are open wormholes, while the gray spaces are holes with the possibility of opening at the next step (the darker they are, the higher the probability).

The novelty function for each agent f_i , which is used for calculating the intrinsic rewards in Figure 1, is defined as $\frac{1}{N^{\zeta}}$, where N is the number of times the agent has visited its current cell and ζ is a decay rate selected as a hyperparameter (we find $\zeta=0.7$ works well for our purposes).

6.2. VizDoom Domain

In order to test our method's ability to scale to more complex environments with similarly challenging exploration tasks, we implement tasks analogous to those in our gridworld environment (i.e. extrinsic rewards are defined identically) in the VizDoom framework (Kempka et al., 2016). We use the "My Way Home" map, which has been used as a test bed for single agent exploration techniques (Pathak et al., 2017), and modify it for multi-agent tasks (pictured in Figures 3b

and 3c). Since the agents are moved to a central location closer to their rewards than in the original map, we lower the action repeat from 4 to 2, in order to force agents to take twice as many steps in order to explore the same areas, maintaining the challenging nature of exploration in the original task.

As in the gridworld setting, we use count-based intrinsic rewards for VizDoom; however, since VizDoom is not a discrete domain, we separate agents' (x,y) positions into discrete bins and use the counts for these bins. We again find $\zeta=0.7$ to work well in our experiments.

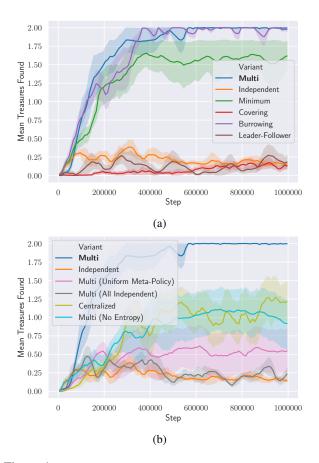


Figure 4: (Top) Mean number of trasures found per episode on TASK 1 with 2 agents in the gridworld domain. Shaded region is a 68% confidence interval across 6 runs of the running mean over the past 100 episodes. Our approach (MULTI-EXPLORATION) is competitive with the best individual intrinsic reward function, using the same number of environment samples without any prior knowledge provided. (Bottom) Ablations and baselines in the same setting. We show both aspects of our approach (the meta-policy selector and the diverse intrinsic reward functions) are crucial for successful completion of exploration tasks requiring coordination. Furthermore, we show that the intrinsic rewards developed within our framework are advantageous when compared to a naive centralized approach to exploration.

6.3. Main Results

Figure 4a demonstrates the results of our approach over the course of training on the 2 agent version of TASK 1in gridworld, and the final results on each task/agent/domain combination can be found in Table 1. The full training curves for all settings can be found in the supplementary material. We train a team of agents using each of the multi-agent intrinsic reward functions defined in Figure 1 individually, and then test our dynamic policy selection approach. We find our approach is competitive with, or outperforms the best performing individual exploration method in nearly all tasks. This performance is exciting since our method receives no prior information about which type of exploration would work best, while each type carries its own bias. Furthermore, we find our results on the more complex VizDoom domain mirror those in the gridworld, indicating our methods are not limited to discrete domains, assuming a reliable way for measuring the novelty of observations exists.

Interestingly, our approach is sometimes able to significantly surpass the performance of the best individual reward function on TASK 3, indicating that the evolving meta-policy may provide a sort of automatic curriculum for learning in certain tasks. We explore this behavior in more detail in the supplementary material. We find our approach is unable to match the performance of the best individual method on TASK 2 in some settings (gridworld with 3 agents and VizDoom). This lack of success may be an indication that the exploration strategies which perform well in these settings require commitment early on in training, highlighting a limitation of our approach. Our method requires testing out all policies until we find one that reaches high extrinsic rewards, which can dilute the effectiveness of exploration early on.

6.4. Analysis

Characteristics of Different Intrinsic Rewards In order to better understand how each reward type encourages agents to explore the state space, we visualize their exploration in videos, viewable at the anonymized link below. Independent rewards, as expected, result in agents exploring the whole state space without taking other agents into consideration. As a result, on TASK 1, which requires coordination between agents to spread out and explore different areas, Independent rewards struggle; however, on TASK 3, where agents receive individualized goals, independent exploration usually performs better, relative to the other methods. TASK 2 also requires coordination, but the rate of wormholes opening in the gridworld version is lower on that task, making exploration easier. As a result, Independent pendent rewards perform well on TASK 2; however, we find

¹https://sites.google.com/view/multi-exploration-icml2020/home

Table 1: # of treasures found with standard deviation across 6 runs. Scores are bolded where the best mean score falls within one standard deviation.

Gridworld									
Intrinsic reward type (fixed or adaptive as in our approach MULTI)									
Task	n	INDEPENDENT	MINIMUM	COVERING	Burrowing	LEAD-FOLLOW	MULTI		
1	2	0.14 ± 0.05	1.62 ± 0.59	0.13 ± 0.12	$\textbf{1.98} \pm \textbf{0.06}$	0.18 ± 0.24	2.00 ± 0.00		
	3	1.16 ± 0.11	$\textbf{1.49} \pm \textbf{0.76}$	0.00 ± 0.00	$\textbf{2.06} \pm \textbf{1.05}$	0.34 ± 0.45	2.23 ± 0.73		
	4	0.84 ± 0.29	$\textbf{1.78} \pm \textbf{0.44}$	0.00 ± 0.00	$\textbf{1.90} \pm \textbf{0.49}$	1.17 ± 0.39	$\textbf{2.04} \pm \textbf{0.61}$		
2	2	$\textbf{2.00} \pm \textbf{0.00}$	0.92 ± 0.10	1.11 ± 0.99	0.98 ± 0.05	$\textbf{1.73} \pm \textbf{0.66}$	1.83 ± 0.41		
	3	$\textbf{2.66} \pm \textbf{0.80}$	1.11 ± 0.29	0.54 ± 0.80	1.80 ± 0.29	$\textbf{3.00} \pm \textbf{0.00}$	1.80 ± 0.71		
	4	$\textbf{1.83} \pm \textbf{1.08}$	0.93 ± 0.13	0.22 ± 0.18	$\textbf{1.99} \pm \textbf{0.67}$	$\textbf{2.66} \pm \textbf{2.06}$	$\textbf{2.54} \pm \textbf{1.21}$		
3	2	1.39 ± 0.94	0.67 ± 1.03	0.29 ± 0.37	0.67 ± 1.03	0.83 ± 0.67	2.00 ± 0.00		
	3	$\textbf{1.68} \pm \textbf{0.70}$	0.60 ± 0.73	0.09 ± 0.08	$\textbf{1.35} \pm \textbf{1.16}$	$\textbf{1.59} \pm \textbf{0.83}$	$\textbf{2.21} \pm \textbf{0.91}$		
	4	1.12 ± 0.47	1.36 ± 0.71	0.05 ± 0.05	$\textbf{2.14} \pm \textbf{1.49}$	0.68 ± 0.53	$\textbf{1.73} \pm \textbf{0.47}$		
				VizDoom	1				
1	2	0.94 ± 0.54	$\textbf{1.57} \pm \textbf{0.74}$	0.16 ± 0.17	$\textbf{1.94} \pm \textbf{0.10}$	0.61 ± 0.43	1.98 ± 0.03		
2		$\textbf{1.52} \pm \textbf{0.75}$	$\textbf{1.53} \pm \textbf{0.74}$	0.70 ± 1.00	0.63 ± 0.04	$\textbf{1.93} \pm \textbf{0.10}$	1.23 ± 0.65		
3		0.18 ± 0.19	$\textbf{0.64} \pm \textbf{1.05}$	0.45 ± 0.46	0.29 ± 0.25	0.20 ± 0.17	$\textbf{1.64} \pm \textbf{0.63}$		

LEADER-FOLLOWER also performs well on this task, expecially when more agents are involved, indicating that these rewards do a good job of biasing agents toward exploring similar regions of the environment.

MIMIMUM rewards prevent agents from exploring the same regions redundantly but can lead to situations where one of the agents is the first to explore all regions that provide sparse extrinsic rewards. In these cases, other agents are not aware of the extrinsic rewards and are also not motivated to explore their regions since another agent has already done so. COVERING rewards, as expected, lead to behavior where agents are constantly switching up the regions they explore. While this behavior does not prove to be useful in the tasks we test since the switching slows down overall exploration progress, it may be useful in other scenarios. Finally, BUR-ROWING rewards cause agents to each explore different subregions and continue to explore those regions until they exhaust their options. This behavior is particularly effective on TASK 1, where agents are best served by spreading out and exploring the whole map in a mutually exclusive fashion.

Ablations and Baselines We first compare to a metapolicy which simply selects the action policies uniformly at random. We find our approach is significantly superior to this baseline (see Figure 4b *Multi (Uniform Meta-Policy)*). We also find that our approach outperforms itself without an entropy bonus on the policy selector (*Multi (No Entropy)*), demonstrating that our method effectively balances exploration and exploitation in terms of policy selection. Furthermore, we test a version of our method where all policies (with different random initializations) are trained on independent rewards (*Multi (All Independent)*). The purpose of this ablation is to test the degree to which the specific multi-

agent intrinsic reward functions are helpful, as opposed to simply providing multiple different options at each episode. Again, we find our method outperforms the baseline, indicating that both aspects of our approach (diverse intrinsic reward functions which share information across agents, and a meta-policy selector that maximizes extrinsic rewards while continuing to explore other option) are crucial for success in multi-agent exploration tasks.

Finally, in (*Centralized*) we compute intrinsic rewards under the assumption that all agents are treated as one agent. In other words, we use the inverse count of the number of times **all** agents have jointly taken up their combined positions, rather than considering agents independently. While this reward function will ensure the global state space is thoroughly searched, it lacks the inductive biases toward spatial coordination that our reward functions incorporate. As such, it does not learn as efficiently as our method.

7. Conclusion

We propose a set of multi-agent intrinsic reward functions with differing properties, and compare them both qualitatively and quantitatively on several multi-agent exploration tasks in both a gridworld domain as well as in VizDoom. Overall, we can see that navigation-based cooperative multi-agent tasks can, in many cases, benefit from intrinsic rewards that take into account what other agents have explored, but there are various ways to incorporate that information, each with differing properties. We show that our method is capable of matching or surpassing the performance of the best performing intrinsic reward type on various tasks while using the same number of samples collected from the environment.

References

- Agogino, A. K. and Tumer, K. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, 17(2):320–338, 2008.
- Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.
- Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.
- Carmel, D. and Markovitch, S. Exploration and adaptation in multiagent systems: A model-based approach. In *IJCAI* (1), pp. 606–611, 1997.
- Foerster, J., Assael, I. A., de Freitas, N., and Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016.
- Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, 2018.
- Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of* the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018.
- Houthooft, R., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pp. 1109–1117, 2016.
- Iqbal, S. and Sha, F. Actor-attention-critic for multiagent reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2961–2970, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/iqbal19a.html.
- Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P. A., Strouse, D. J., Leibo, J. Z., and de Freitas, N. Social influence as intrinsic motivation for Multi-Agent deep reinforcement learning. arXiv preprint arXiv:1810.08647, October 2018.

- Jiang, J. and Lu, Z. Learning attentional communication for multi-agent cooperation. arXiv preprint arXiv:1805.07733, 2018.
- Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pp. 341–348, Santorini, Greece, Sep 2016. IEEE. URL http://arxiv.org/abs/1605.02097. The best paper award.
- Khadka, S., Majumdar, S., Miret, S., Tumer, E., Nassar, T., Dwiel, Z., Liu, Y., and Tumer, K. Collaborative evolutionary reinforcement learning. *arXiv preprint arXiv:1905.00976*, 2019.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2014.
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.
- Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pp. 6382–6393, 2017.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.
- Oliehoek, F. A., Amato, C., et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.
- Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2721–2730. JMLR. org, 2017.
- Oudeyer, P.-Y. and Kaplan, F. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.
- Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In Precup, D. and Teh, Y. W. (eds.), Proceedings of the 34th International Conference on Machine Learning, volume 70 of Proceedings of Machine Learning Research, pp. 2778–2787, International Convention Centre, Sydney, Australia, 06–11 Aug

- 2017. PMLR. URL http://proceedings.mlr.press/v70/pathak17a.html.
- Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4295–4304, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018.
- Schmidhuber, J. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- Strehl, A. L. and Littman, M. L. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309 1331, 2008. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2007.08.009. URL http://www.sciencedirect.com/science/article/pii/S0022000008000767. Learning Theory 2005.
- Sukhbaatar, S., Fergus, R., et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pp. 2244–2252, 2016.
- Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pp. 2085–2087, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.
- Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, O. X., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural informa*tion processing systems, pp. 2753–2762, 2017.
- Verbeeck, K., Nowé, A., and Tuyls, K. Coordinated exploration in multi-agent reinforcement learning: an application to load-balancing. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pp. 1105–1106. ACM, 2005.

Wang, T., Wang, J., Wu, Y., and Zhang, C. Influence-based multi-agent exploration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJgy96EYvr.

A. Environment Details

A.1. Gridworld

The wormholes which send agents back to their starting positions if they are stepped into are an important aspect of the environment, as they add difficulty to exploration. The probability, ρ , of a wormhole opening at each step, t, evolves as such: $\rho_{t+1} = \rho_t + \mathcal{N}(\mu, \sigma)$, where $\mu = \sigma = 0.05$ for **TASK 1** and $\mu = \sigma = 0.005$ for **2** and **3**.

Agents observe their global position in (x,y) coordinates (scalars), as well as local information regarding walls in adjacent spaces, the probability of their adjacent spaces opening into a wormhole, the relative position of other agents (if they are within 3 spaces), as well as information about which treasures the agent has already collected in the given episode. The global state is represented by the (x,y) coordinates of all agents, as one-hot encoded vectors for x and y separately, as well as the local information of all agents regarding wormholes, walls, and treasures collected. Each agent's action space consists of the 4 cardinal directions as well as an option to not move, which is helpful in cases where an agent is waiting for a wormhole to be safe to cross.

A.2. VizDoom

Agents receive their egocentric view in the form of 48x48 grayscale images as observations along with an indicator of which agents (if any) have collected each reward, and we use a vector based global state which includes all agents' (x, y) positions and velocities, their orientations, as well as the same indicator of which agent has collected each reward. The policies do not receive the global state as input, as it is only used for the centralized critics. As in the gridworld setting, we use count-based intrinsic rewards for VizDoom; however, since VizDoom is not a discrete domain, we separate agents' (x, y) positions into discrete bins and use the counts for these bins. There are 30 bins in the x dimension and 26 in the y dimension. (x, y) positions in the global state are represented both as scalars and onehot vectors indicating which bin the agents are currently occupying. Each agent can choose from 3 actions at each time step: turn left, turn right, or go forward.

B. Training Details

The training procedure is detailed in Algorithm 1, and all hyperparameters are listed in Tables 2 and 3. Hyperparameters were selected by tuning one parameter at a time through

intuition on task 1 with 2 agents and then applying to the rest of the settings with minimal changes. Individual runs used a single NVIDIA Titan Xp GPU for training.

Algorithm 1 Training Procedure for Multi-Explore w/ Soft Actor-Critic (Haarnoja et al., 2018)

```
1: Initialize environment with n agents
 2: Initialize replay buffer, D
 3: t_{\text{update}} \leftarrow 0
 4: t_{ep} \leftarrow \max ep length
 5: for t = 1 \dots total steps do
        if episode done or t_{ep} == \max ep length then
 6:
           for j = 1 \dots num updates do
 7:
 8:
               UpdateSelector(R, h) {Eqs 9-10 in main text}
 9:
           end for
10:
           s, \mathbf{o} \leftarrow ResetEnv()
           h \sim \Pi {Sample policy head}
11:
12:
           R \leftarrow 0
13:
14:
        end if
        Select actions a_i \sim \pi_i^h(\cdot|o_i) for each agent, i
15:
        Send actions to environment and get s, o, r
16:
        R \leftarrow R + \gamma^{t_{\text{ep}}} r
17:
        Store transitions for all environments in D
18:
19:
        t_{\text{update}} + = 1
        t_{ep}+=1
20:
21:
        if t_{update} == steps per update then
           for j = 1 \dots num updates do
22:
23:
              Sample minibatch, B
              UpdateCritic(B) {Eqs 2,4,5 in main text}
24:
25:
              UpdatePolicies(B) {Eqs 6-8 in main text}
              Update target parameters:
26:
                              \bar{\Psi} = \tau \bar{\Psi} + (1 - \tau)\Psi
                              \bar{\Theta} = \tau \bar{\Theta} + (1 - \tau)\Theta
           end for
27:
28:
           t_{\text{update}} \leftarrow 0
29:
        end if
30: end for
```

C. Network Architectures

In this section we list, in pseudo-code, the architectures we used for all policies and critics

C.1. Gridworld

 θ_i^{share} (shared for policy heads):

```
obs_size = observations.shape[1]
fc1 = Linear(in_dim=obs_size, out_dim=128)
nl1 = ReLU()
```

 θ_i^j (specific to each policy head):

```
n_acs = actions.shape[1]
fc2 = Linear(in_dim=fc1.out_dim,
    out_dim=32)
n12 = ReLU()
fc3 = Linear(in_dim=fc2.out_sim,
    out_dim=n_acs)
```

 ψ^{share} (shared across critics for all agents and reward types):

```
state_size = states.shape[1]
fc1 = Linear(in_dim=state_size,
   out_dim=128)
nl1 = ReLU()
```

 $\psi_{i,j}$ (specific to each agent/policy head combination, same architecture for extrinsic and intrinsic critics):

C.2. VizDoom

 θ_i^{share} (shared for policy heads belonging to one agent):

```
# vector observation encoder
vect_obs_size =
   vector_observations.shape[1]
vect_fc = Linear(in_dim=obs_size,
   out_dim=32)
vect_nl = ReLU()
# image observation encoder
img_obs_channels =
    image_observations.shape[1]
pad1 = ReflectionPadding(size=1)
conv1 =
   Conv2D(in_channels=img_obs_channels,
   out_channels=32, filter_size=3,
   stride=2)
conv_nl1 = ReLU()
pad2 = ReflectionPadding(size=1)
conv2 =
    Conv2D(in_channels=conv1.out_channels,
   out_channels=32, filter_size=3,
   stride=2)
conv_nl2 = ReLU()
pad3 = ReflectionPadding(size=1)
conv3 =
   Conv2D(in_channels=conv2.out_channels,
   out_channels=32, filter_size=3,
   stride=2)
conv_nl3 = ReLU()
pad4 = ReflectionPadding(size=1)
conv4 =
   Conv2D(in_channels=conv3.out_channels,
```

Table 2: Hyperparameter settings across all runs in gridworld.

Name	Description	Value
\overline{Q} lr	learning rate for centralized critic	0.001
Q optimizer	optimizer for centralized critic	Adam (Kingma & Ba, 2014)
π lr	learning rate for decentralized policies	0.001
π optimizer	optimizer for decentralized policies	Adam
Π lr	learning rate for policy selector	0.04
Π optimizer	optimizer for policy selector	SGD
au	target function update rate	0.005
bs	batch size	1024
total steps	number of total environment steps	1e6
steps per update	number of environment steps between updates	100
niters	number of iterations per update	50
max ep length	maximum length of an episode before resetting	500
Ψ penalty	coefficient for weight decay on parameters of Q-function	0.001
Θ penalty	coefficient on L_2 penalty on pre-softmax output of policies	0.001
θ penalty	coefficient for weight decay on parameters of policy selector	0.001
D	maximum size of replay buffer	1e6
α	action policy reward scale	100
η	selector policy reward scale	5
γ	discount factor	0.99
β	relative weight of intrisic rewards to extrinsic	0.1
ζ	decay rate of count-based rewards	0.7

```
out_channels=32, filter_size=3,
    stride=2)
conv_n14 = ReLU()
conv_flatten = Flatten() # flatten output
    of conv layers
conv_fc =
    Linear(in_dim=conv_flatten.out_dim,
    out_dim=128)
conv_fc_n1 = ReLU()
```

θ_i^j (specific to each policy head):

```
n_acs = actions.shape[1]
# takes concatenation of image and vector
    encodings as input
fc_out1 = Linear(in_dim=conv_fc.out_dim +
        vect_fc.out_dim, out_dim=32)
fc_out_nl = ReLU()
fc_out2 = Linear(in_dim=fc_out1.out_dim,
        out_dim=n_acs)
```

ψ^{share} (shared across critics for all agents and reward types):

```
state_size = states.shape[1]
fc1 = Linear(in_dim=state_size,
```

```
out_dim=256)
nl1 = ReLU()
```

$\psi_{i,j}$ (specific to each agent/policy head combination, same architecture for extrinsic and intrinsic critics):

Table 3: Hyperparameter settings across all runs in VizDoom (only where different from Table 2).

Name	Description	Value
$\overline{Q \ln}$	learning rate for centralized critic	0.0005
π lr	learning rate for decentralized policies	0.0005
bs	batch size	128
$\overline{ D }$	maximum size of replay buffer	5e5

D. Training Curves

D.1. Gridworld

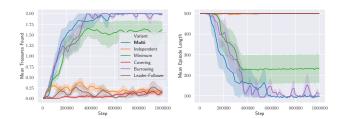


Figure 5: Results on Task 1 in Gridworld with 2 agents.

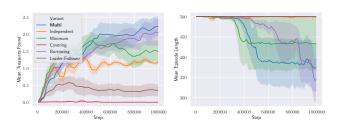


Figure 6: Results on Task 1 in Gridworld with 3 agents.

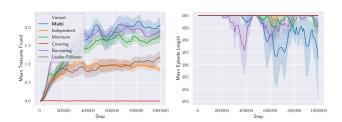


Figure 7: Results on Task 1 in Gridworld with 4 agents.

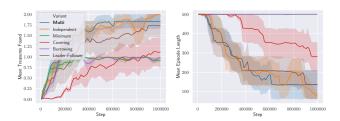


Figure 8: Results on Task 2 in Gridworld with 2 agents.

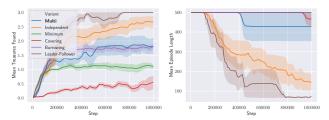


Figure 9: Results on Task 2 in Gridworld with 3 agents.

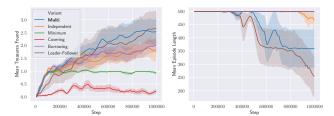


Figure 10: Results on Task 2 in Gridworld with 4 agents.

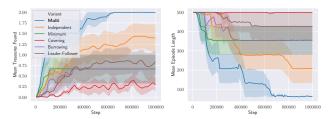


Figure 11: Results on Task 3 in Gridworld with 2 agents.

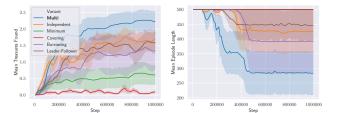


Figure 12: Results on Task 3 in Gridworld with 3 agents.

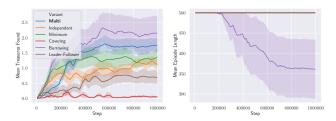


Figure 13: Results on Task 3 in Gridworld with 4 agents.

D.2. Vizdoom

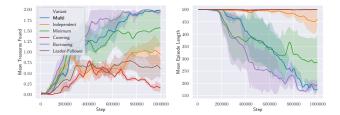


Figure 14: Results on Task 1 in Vizdoom.

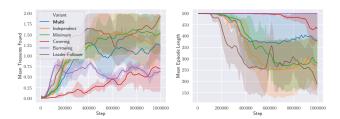


Figure 15: Results on Task 2 in Vizdoom.

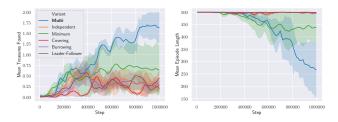


Figure 16: Results on Task 3 in Vizdoom.

E. More Ablations

In this section we consider two ablations/comparisons to our model across all three tasks in the 2 agent version of gridworld. In the first (*Centralized*) we compute intrinsic rewards under the assumption that all agents are treated as one agent. In other words, we use the inverse count of the number of times that **all** agents have jointly taken up their combined positions, rather than considering agents independently. While this reward function will ensure that the global

state space is thoroughly searched, it lacks the inductive biases toward spatial coordination that our reward functions incorporate. As such, it does not learn as efficiently as our method in any of the three tasks. In the second (*Multi (No Entropy)*) we remove the entropy term from the head selector loss function in order to test its importance. We find that this ablation is unable to match the performance of the full method, indicating that entropy is crucial in making sure that our method does not converge early to a suboptimal policy selector.

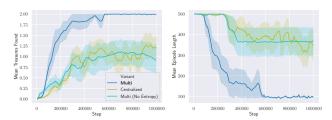


Figure 17: Ablations on Task 1 in Gridworld with 2 agents.

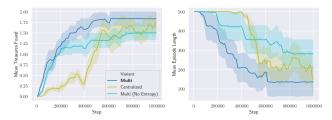


Figure 18: Ablations on Task 2 in Gridworld with 2 agents.

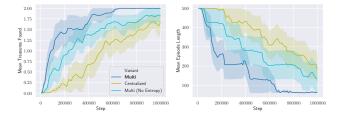


Figure 19: Ablations on Task 3 in Gridworld with 2 agents.

F. Analyzing Meta-Policy

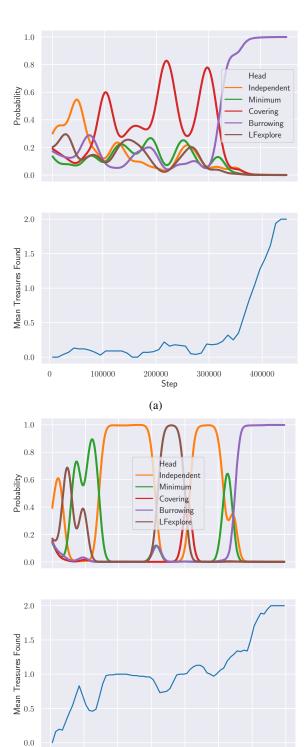


Figure 20: Two runs of our method on GridWorld Task 3 with 2 agents. Top row shows the evolution of the meta-policy's probability of selecting each policy head. Bottom row shows the number of treasures found per episode.

Step
(b)

100000 200000

300000 400000 500000 600000 700000

In Figure 20 we analyze the behavior of the meta-policy in two separate runs. We evaluate on Task 3, since we find that our method is able to surpass the best individual reward function. This task assigns specific goals to each agent. As such, one might expect that independent exploration would work most effectively in this setting. While independent exploration is effective (see Figure 11), we find that our method can outperform it. In both runs, we find that burrowing rewards are selected when the agents finally learn how to solve the task; however, we find that burrowing rewards are not necessarily successful when deployed on their own. This lack of success is likely due to the fact that these rewards cause the agents to pick a region and commit to exploring it for the duration of training. As such, the agents may pick the "wrong" region at first and never be able to recover. On the other hand, using our methods, the meta-policy can wait until the burrowing exploration regions align with the assigned rewards and then select the policies trained on these rewards. This usually ends up being more efficient than waiting for the agents to explore the whole map using independent rewards.