

---

# Variational Automatic Curriculum Learning for Sparse-Reward Cooperative Multi-Agent Problems

---

Jiayu Chen<sup>1‡</sup>, Yuanxin Zhang<sup>1</sup>, Yuanfan Xu<sup>1</sup>, Huimin Ma<sup>3</sup>,  
Huazhong Yang<sup>1</sup>, Jiaming Song<sup>4</sup>, Yu Wang<sup>1</sup>, Yi Wu<sup>12‡</sup>  
<sup>1</sup> Tsinghua University, <sup>2</sup> Shanghai Qi Zhi Institute,  
<sup>3</sup> University of Science and Technology Beijing, <sup>4</sup> Stanford University,  
<sup>‡</sup>jia768167535@gmail.com, <sup>‡</sup>jxwuyi@gmail.com

## Abstract

We introduce a curriculum learning algorithm, *Variational Automatic Curriculum Learning* (VACL), for solving challenging goal-conditioned cooperative multi-agent reinforcement learning problems. We motivate our paradigm through a variational perspective, where the learning objective can be decomposed into two terms: task learning on the current task distribution, and curriculum update to a new task distribution. Local optimization over the second term suggests that the curriculum should gradually expand the training tasks from easy to hard. Our VACL algorithm implements this variational paradigm with two practical components, task expansion and entity progression, which produces training curricula over both the task configurations as well as the number of entities in the task. Experiment results show that VACL solves a collection of sparse-reward problems with a large number of agents. Particularly, using a single desktop machine, VACL achieves 98% coverage rate with 100 agents in the simple-spread benchmark and reproduces the ramp-use behavior originally shown in OpenAI’s hide-and-seek project. Our project website is at <https://sites.google.com/view/vacl-neurips-2021>.

## 1 Introduction

Building intelligent agents in complex multi-agent games remains a long-standing challenge in artificial intelligence [23]. Recently, it has been a trend to apply multi-agent reinforcement learning (MARL) to extremely challenging multi-agent games, such as Dota II [5], StarCraft II [30] and Hanabi [3]. Despite these successes, learning intelligent multi-agent policies in general still remains a great RL challenge. Multi-agent games allow sophisticated interactions between agents and environment. Feasible solutions may require non-trivial intra-agent coordination, which leads to substantially more complex strategies than the single-agent setting. Moreover, as the number of agents increases, the joint action space grows at an exponential rate, which results in an exponentially large policy search space. Thus, most existing MARL applications typically require shaped rewards, or assume simplified environment dynamics, or only handle a limit number of agents.

We tackle goal-conditioned cooperative MARL problems with *sparse* rewards through a novel variational inference perspective. Assuming each task can be parameterized by a continuous representation, we introduce a variational proposal distribution over the task space and then decomposing the overall training objective into two separate terms, i.e., a policy improvement term and a task proposal update term. By treating the proposal distribution updates as the training curriculum, such a variational objective naturally suggests a curriculum learning framework by alternating between curriculum update and MARL training. In addition, we propose a continuous relaxation technique for discrete variables so that the variational curricula can be also applied over the number of agents and objects in the task, which leads to a generic and unified training paradigm for the cooperative MARL setting.

We implement our variational training paradigm through a computationally efficient algorithm, *Variational Automatic Curriculum Learning (VACL)*. VACL consists two components, *task expansion* and *entity progression*, which generate a series of effective training tasks in a hierarchical manner. Intuitively, *entity progression* leverages the inductive bias that tasks with more entities are usually more difficult in MARL and, therefore, progressively increases the entity size in the environment. *Task expansion* assumes a fixed number of entities and is motivated by Stein variational gradient descent (SVGD) [17] to efficiently expand the task distribution towards the entire task space.

We provide thorough discussions on the differences between VACL and existing works (Sec. 3.4) and empirically validate VACL using a single desktop machine in four physical multi-agent cooperative games, i.e., two in the multi-agent particle-world environment (MPE) [19] and two in the MuJoCo-based hide-and-seek environment (HnS) [2] (Sec. 4). In the MPE games, VACL learns significantly faster and better than all baselines and achieves over **98% coverage rate** with **100 agents** in the *simple-spread* testbed even using **sparse rewards**. In the HnS scenarios, VACL achieves over 90% success rates on both two games, including reproducing the *ramp use* behavior discovered in [2] and solving a sparse-reward multi-agent *Lock-and-Return* challenge where none of the baselines can produce a success rate above 15% using the same amount of training timesteps.

## 2 A Variational Perspective on Curriculum Learning

### 2.1 Preliminary

We formulate each goal-conditioned cooperative multi-agent task as a multi-agent Markov decision process (MDP),  $M(n, \phi) = \langle n, \phi, \mathcal{S}, \mathcal{A}, \mathcal{O}, O, P, R, s_\phi^0, g_\phi, \gamma \rangle$ , parameterized by a discrete variable  $n$  (e.g., the number of agents<sup>1</sup>) and a continuous parameter  $\phi \in \Phi$  (i.e., the initial state and goal; details in App. C.2).  $\mathcal{S}$  is the state space.  $\mathcal{A}$  is the shared action space for each agent.  $\mathcal{O}$  is the observation space.  $o_i = O(s; i)$  denotes the observation for agent  $i$  under state  $S$ .  $s_\phi^0$  denotes the initial state and  $g_\phi$  denotes the goal.  $P(s'|s, A)$  and  $R(s, A; g_\phi)$  are the transition probability and the goal-conditioned reward function given state  $s$  and joint actions  $A = (a_1, \dots, a_n)$ .  $\gamma$  is the discounted factor.

We consider homogeneous agents and learn a shared goal-conditioned policy  $\pi_\theta(a_i|o_i, g)$  parameterized by  $\theta$  for each agent  $i$ . The final objective is to maximize the expected accumulative reward over the entire task space, i.e.,

$$J(\theta) = \mathbb{E}_{n, \phi, a_i^t, s^t} \left[ \sum_t \gamma^t R(s^t, A^t; g_\phi) \right] = \mathbb{E}_{n, \phi} [V(n, \phi, \pi_\theta)], \quad (1)$$

where  $V(n, \phi, \pi_\theta)$  denotes the value function for  $\pi_\theta$  over task  $M(n, \phi)$ . We remark that for problems with 0/1 sparse rewards,  $V(n, \phi, \pi_\theta)$  can be also interpreted as the success rate of  $\pi_\theta$ <sup>2</sup>.

The main idea of curriculum learning is to construct a task sampler  $q(n, \phi)$  that always generates training tasks  $M(n, \phi)$  that yield the largest learning progress for  $\pi_\theta$  to efficiently maximize  $J(\theta)$ .

### 2.2 Stein Variational Inference over a Continuous Task Space

For simplicity, we ignore the discrete variable  $n$  and only focus on the continuous variable  $\phi$  in this subsection. Herein, our objective is to maximize the expectation  $\mathcal{L} = \mathbb{E}_{\phi \sim p(\phi)} [V(\phi, \pi)]$ , where  $p(\phi)$  is a uniform distribution over feasible values of  $\phi$ . The subscript  $\theta$  of  $\pi$  is omitted for conciseness.

Given the objective  $\mathcal{L}$ , we motivate our automated curriculum learning procedure through the lens of variational inference. Let  $q(\phi)$  denote the current distribution of training tasks, which will vary throughout learning; we can represent curriculum learning through an objective that optimizes  $q(\phi)$ .

<sup>1</sup> $n$  can also be a discrete vector where each dimension represents the number of a particular type of objects. We now simply assume  $n$  is a single integer denoting the agent number for conciseness.

<sup>2</sup>This only rigorously holds when  $\gamma = 1$ . But let's assume it for simplicity. In practice, this statement approximately holds with a close-to-1  $\gamma$  and a relatively short horizon.

The following lower bound for the original objective  $\mathcal{L}$  can be derived:

$$\begin{aligned} \mathcal{L} &= \mathbb{E}_{\phi \sim p} [V(\phi, \pi)] = \mathbb{E}_{\phi \sim q} \left[ \frac{p(\phi)}{q(\phi)} V(\phi, \pi) \right] = \mathbb{E}_{\phi \sim q} \left[ V(\phi, \pi) + \left( \frac{p(\phi)}{q(\phi)} - 1 \right) V(\phi, \pi) \right] \quad (2) \\ &\geq \underbrace{\mathbb{E}_{\phi \sim q(\phi)} [V(\phi, \pi)]}_{\mathcal{L}_1: \text{policy update}} + \underbrace{\mathbb{E}_{\phi \sim q(\phi)} \left[ V(\phi, \pi) \log \frac{p(\phi)}{q(\phi)} \right]}_{\mathcal{L}_2: \text{curriculum update}} \quad (3) \end{aligned}$$

where the inequality is due to  $x - 1 \geq \log x$ , with equality achieved at  $p(\phi) = q(\phi)$  for all  $\phi$ .

Now, we have decomposed  $\mathcal{L}(\pi, q)$  into a lower bound that contains two terms  $\mathcal{L}_1$  and  $\mathcal{L}_2$ .  $\mathcal{L}_1$  is simply a policy update objective under the current task curriculum  $q(\phi)$  while the second term  $\mathcal{L}_2$  can be interpreted as a curriculum update objective by maximizing  $q(\phi)$  w.r.t. the policy  $\pi$ . Therefore, the decomposed lower bound naturally yields a curriculum learning framework by performing an iterative learning procedure, i.e., alternating between (1) maximizing  $\mathcal{L}_1$  with  $\pi$  (policy update) and (2) maximizing  $\mathcal{L}_2$  with  $q(\phi)$  (curriculum update). We remark that we only optimize  $\pi$  w.r.t.  $\mathcal{L}_1$  instead of  $\mathcal{L}_1 + \mathcal{L}_2$  for the purpose of stabilizing policy learning since  $q$  can be very different from  $p$  in the early stage of training. This also aligns with the curriculum learning paradigm. In the following proposition, we show that if we can perfectly optimize the RL procedure for  $\mathcal{L}_1$  under the curriculum  $q(\phi)$ , then the curriculum objective  $\mathcal{L}_2$  will encourage  $q(\phi)$  to converge to  $p(\phi)$ .

**Proposition 1.** *Assume that  $V(\phi, \pi) \in [0, 1]$  and that there exists a unique policy  $\pi^*$  such that  $V(\phi, \pi^*) = 1$  for all  $\phi$ . For any  $q \in \mathcal{P}(\Phi) \setminus \{p\}$  (i.e., any curriculum that is not the final one) and  $\pi \in \arg \max_{\pi} \mathcal{L}_1(\pi, q)$  (i.e., a policy that is optimal under the current curriculum),  $\mathcal{L}_2(\pi, q) < \mathcal{L}_2(\pi^*, p) = 0$ .*

*Proof.* First, we have that  $\mathcal{L}_1(\pi^*, p) = 1$  (definition of  $\pi^*$ ) and  $\mathcal{L}_2(\pi^*, p) = 0$ . Suppose that  $\exists \pi, q$  such that  $q \neq p$  and  $\mathcal{L}_1(\pi, q) = 1$ ; then it must be the case that whenever  $V(\phi, \pi) \neq 1$ ,  $q(\phi) = 0$ , i.e.,  $\forall \phi \in \text{supp}(q)$ ,  $V(\phi, \pi) = 1$ . Then  $\mathcal{L}_2(\phi, \pi) = -D_{\text{KL}}(q||p) < 0$ , which completes the proof.  $\square$

In other words, if our policy  $\pi$  already performs well under the current curriculum  $q(\phi)$ , then  $\mathcal{L}_2$  will encourage  $q(\phi)$  to become “closer” to  $p(\phi)$ , aligning with our eventual target of learning all tasks.

### 2.2.1 Stein Variational Gradient Descent for Curriculum Update

The most straightforward approach for optimizing  $\mathcal{L}_2$  is to learn a generative model w.r.t. the current  $\pi$ , which can be particularly sample-inefficient. Instead, we consider updates to our current curriculum via smooth updates, which is much more efficient than learning generative models.

Let us represent the task distribution under the current curriculum  $q(\phi)$  with a set of particles  $Q = \{\phi\}$ . To update each particle  $\phi$  in  $Q$ , we consider an incremental transform  $T(\phi) = \phi + \epsilon f(\phi)$ , where  $f(\phi)$  is a smooth function and  $\epsilon$  is a magnitude scalar. Our goal is then to derive suitable updates to  $Q$  that maximizes  $\mathcal{L}_2(q, \pi)$  based on the transform  $T$ . Following [9] and [27], we introduce an assumption over the generalization ability of the task space.

**Assumption 1.** *An optimal policy  $\pi$  that is trained over a specific task  $\phi$  has some generalization ability to another goal  $\phi'$  close to  $\phi$ . Specifically, if  $V(\phi, \pi) = c$  for any  $\phi \in \text{supp}(q)$ , then  $V(\phi', \pi) = c$  for all  $\phi' \in \mathcal{B}(\phi)$ , where  $\mathcal{B}(\phi)$  is a small open ball around  $\phi$ .*

In the context of curriculum learning, this assumption suggests that for any policy that is trained on a set of tasks, the performance (e.g., success rate) does not change if we only change the task parameters by an infinitesimal amount. In the following theorem, we derive the functional gradient of the curriculum update objective  $\mathcal{L}_2$  in a reproducing kernel Hilbert space (RKHS).

**Theorem 1.** *Let  $T(\phi) = \phi + \epsilon f(\phi)$  where  $f$  is an element of some vector-valued RKHS of a positive definite kernel  $k(\phi, \phi') : \Phi \times \Phi \rightarrow \mathbb{R}$ , and  $q_{[T]}$  the density of  $\psi = T(\phi)$  when  $\phi \sim q$ , then*

$$\nabla_f \mathcal{L}_2(q_{[T]}, \pi)|_{f=0} = f^*(\phi), \quad (4)$$

where  $f^*(\cdot) = \mathbb{E}_{\phi' \in Q} [V(\phi', \pi)(k(\phi', \cdot) \nabla_{\phi'} \log p(\phi') + \nabla_{\phi'} k(\phi', \cdot))]$ .

Proof is in App. A. While the above statement is very similar to that used for Stein variational gradient descent [17], the update for our case also directly depends on the value  $V(\phi', \pi)$ , and thus indirectly

depends on the current policy. Intuitively, if  $V(\phi', \pi)$  is large, then  $\phi'$  will contribute more to the curriculum update, whereas if  $V(\phi', \pi) = 0$ , then  $\phi'$  will not contribute to the update.

Since we assume that  $p(\phi)$  is uniform over the task space,  $\nabla_{\phi'} \log p(\phi') = 0$  for the interior of  $\Phi$ , and the task update becomes

$$f^*(\cdot) = \mathbb{E}_{\phi' \in \mathcal{Q}} [V(\phi', \pi) \cdot \nabla_{\phi'} k(\phi', \cdot)]. \quad (5)$$

For example, we take the RBF kernel  $k(\phi, \phi') = \exp(-\frac{1}{h} \|\phi - \phi'\|_2^2)$ , and then Eq. (5) will drive  $\phi$  away from neighbouring points that have large  $k(\phi, \phi')$ , and the repelling force scales with  $V(\phi', \pi)$ . If  $V(\phi', \pi) = 0$ , i.e.,  $\phi'$  has not been solved at all, then there is no repelling force from  $\phi'$  to  $\phi$ . Intuitively, this process encourages the curriculum update to prioritize tasks that have not been fully solved yet, by exploring in the vicinity of existing tasks with high values.

### 2.3 Continuous Relaxation for Discrete Parameter

Now we switch to the discrete parameter  $n$  in our task representation, which results in a non-continuous task space and can be difficult to directly combine with continuous optimization techniques. Therefore, we propose to utilize a continuous variable alternative  $z$  to represent the discrete variable  $n$  so that the same variational inference technique for  $\phi$  can be applied here. Specifically, assuming  $n$  denotes the number of agents, we represent  $n$  with a categorical distribution  $p(n; z) = \text{Cat}(z_1, z_2, \dots, z_N)$  parametrized by a continuous vector  $z = [z_1, \dots, z_N]$  assuming the maximum possible of agents<sup>3</sup> is  $N$ .  $p(n; z)$  denotes the distribution which generates  $n$  agents with probability  $z_n$ . In this case, we can similarly introduce a target distribution  $p(z)$  over the space of  $z$  and derive the corresponding variational updates with  $V(z, \pi)$ . Since we typically wish to learn a policy that can generalize across different number of agents, such a representation over multiple  $n$ 's naturally satisfies this practical requirement. We also remark that this formulation can be directly extended to the problems with multiple discrete variables (e.g., number of both agents and objects) via a categorical distribution over all possible discrete combinations.

## 3 Variational Automatic Curriculum Learning

In this section, we implement the variational inference paradigm with computation-efficient approximations as our VACL algorithm in the context of sparse-reward cooperative MARL problems.

We will first present *Task Expansion* in Sec. 3.1, which is the most important algorithmic component assuming a continuous task space. Then, *Entity Progression*, i.e., the component specialized for discrete task parameters, will be introduced in Sec. 3.2. Finally, the full algorithm and the literature review will follow in Sec. 3.3 and Sec. 3.4.

### 3.1 Task Expansion as Approximated Stein Variational Inference

*Task Expansion* is a computation-efficient curriculum update method, which assumes a continuous task representation  $\phi$  and optimizes the variational lower bound  $\mathcal{L}_2$  w.r.t.  $q(\phi)$ . Directly optimizing  $\mathcal{L}_2$  can be computational challenging. In the following content, we will present a few critical techniques for the efficient approximation computation.

#### 3.1.1 An Ever-Expanding Particle Set

In the standard Stein variational inference framework, we typically maintain a fixed-size particle set  $\mathcal{Q}$  and gradually update the distribution of these particles using Stein variational gradient descent. Since the policy is constantly changing throughout training, we need to repeatedly update the entire set of particles to match the corresponding proposal distribution under  $\mathcal{L}_2$ , which is computationally expensive. Note that in the cooperative MARL setting, assuming a fixed amount of agents and an ever-improving policy, if we initialize  $q(\phi)$  to the proper sub-space of easiest tasks,  $q(\phi)$  would monotonically expand towards the entire task space. Hence, we can simply maintain an ever-expanding particle set by repeatedly adding novel tasks with sufficiently high values (i.e., success rates) under the current policy without the need of modifying those existing particles in  $\mathcal{Q}$ .

<sup>3</sup>If  $N$  is unbounded, we may consider parametrized discrete random processes such as the Chinese restaurant process. For all practical purposes here, this is unnecessary.

### 3.1.2 Value Quantization

Another issue for optimizing  $\mathcal{L}_2$  is that whenever  $\pi$  updates,  $V(\phi, \pi)$  changes accordingly but evaluating the value function can be particularly expensive in MARL problems. As a practical approximation, we roughly categorize all the tasks into 3 categories w.r.t. their values, i.e., a solved subspace of highest values,  $\mathcal{Q}_{\text{sol}} = \{\phi | V(\phi, \pi) > \sigma_{\text{max}}\}$ , an active subspace of moderate values,  $\mathcal{Q}_{\text{act}} = \{\phi | \sigma_{\text{min}} \leq V(\phi, \pi) \leq \sigma_{\text{max}}\}$ , and the remaining unsolved subspace. Here  $\sigma_{\text{min}}$  and  $\sigma_{\text{max}}$  are quantization thresholds. Since the value function would monotonically increase as training proceeds, we only need to verify whether an active task becomes solved or whether an unsolved task becomes active, and maintain  $\mathcal{Q}_{\text{act}}$  and  $\mathcal{Q}_{\text{sol}}$  accordingly. Once a task becomes solved, future value estimation becomes no longer necessary. In addition, w.r.t. the curriculum learning principle, when optimizing  $\pi$  under  $\mathcal{L}_1$ , we can also sample more training tasks from  $\mathcal{Q}_{\text{act}}$  for more effective training.

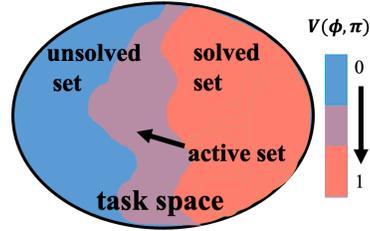


Figure 1: Task space partition.

### 3.1.3 Sampling-Based Particle Exploration

With an ever-expanding particle set  $\mathcal{Q}$ , we need to explore novel task samples in the task space and add those tasks with sufficiently high values (i.e.,  $V \geq \sigma_{\text{min}}$ ) to  $\mathcal{Q}$ . The update equation for  $q(\phi)$  in Eq.(5) naturally suggests a diversity-based task expansion scheme, namely exploring sufficiently novel tasks in the vicinity of  $\mathcal{Q}$ . Due to the difficulty in value estimation, we again follow the value quantization framework and simplify Eq.(5) as follows:

$$\tilde{f}^*(\cdot) \propto \mathbb{E}_{\phi' \in \mathcal{Q}_{\text{sol}}} [\nabla_{\phi'} k(\phi', \cdot)]. \quad (6)$$

In the simplified Eq.(6), we simply ignore tasks with moderate values (i.e., active tasks in  $\mathcal{Q}_{\text{act}}$ ) and only evaluate the expectation w.r.t. tasks with the highest values (i.e., solved tasks in  $\mathcal{Q}_{\text{sol}}$ ) as a practical approximation. Moreover, since  $\mathcal{Q}_{\text{sol}}$  is ever-expanding, Eq.(6) also suggests that we should explore novel tasks in the boundary region between  $\mathcal{Q}_{\text{sol}}$  and  $\mathcal{Q}_{\text{act}}$  (Fig. 2). Hence, in practice, whenever an active task from  $\mathcal{Q}_{\text{act}}$  becomes solved after policy update, we select them as seed tasks to derive novel exploration task samples.

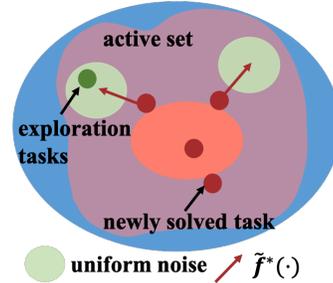


Figure 2: We explore novel tasks in the boundary region between  $\mathcal{Q}_{\text{sol}}$  and  $\mathcal{Q}_{\text{act}}$

Eq.(6) suggests purely gradient-based exploration, which may cause practical issues in curriculum learning since the feasible task parameterization space is often highly constrained. Fig. 5(b) illustrates a maze scenario where agents need to navigate towards the landmarks and avoid the black obstacles. The task parameterization  $\phi$  includes the positions of agents and landmarks. The feasible subspace of  $\phi$  is constrained by the obstacles while these feasibility constraints are only accessible as a black box in the curriculum learning setting. Therefore, directly computing a constraint-free gradient can easily lead to an exploration direction towards infeasible regions. Here we suggest a simple sampling-based enhancement for such highly-constrained problems, which promotes effective task-space exploration and can be viewed as a zero-th order approximation of the projected gradient.

**Rejection-Sampling Exploration:** We explore novel tasks  $\phi_{\text{exp}}$  by adding small uniform noise to the gradient of seed tasks, namely  $\phi_{\text{exp}} \leftarrow \phi_{\text{seed}} + \epsilon \tilde{f}^*(\phi_{\text{seed}}) + \text{Unif}(-\delta, \delta)$ . Then we reject those infeasible tasks by querying the environment for configuration feasibility.

---

**Algorithm 1:** The VACL Algorithm

---

**Input:**  $\theta, n_0, N, B, B_{\text{exp}}$ ; //  $B$ : training batch size;  $B_{\text{exp}}$ : exploration size  
**Output:** final policy  $\pi_\theta$ ;  
 $k \leftarrow 0, \mathcal{Q}_{\text{sol}}^0 \leftarrow \{\}, \mathcal{Q}_{\text{act}}^0 \leftarrow \mathbf{GetEasy}(n_0)$ ; // use easy tasks to initialize  $q(\phi)$   
**repeat**  
     $z \leftarrow 1, n_{k+1} \leftarrow \mathbf{Inc}(n_k), \mathcal{Q}_{\text{sol}}^{k+1} \leftarrow \{\}, \mathcal{Q}_{\text{act}}^{k+1} \leftarrow \mathbf{GetEasy}(n_{k+1})$ ;  
    **while not converge with  $n_k$  do**  
        // Policy Update  
         $\mathcal{M}_{\text{train}} \leftarrow \mathbf{Sample}(B \times z, \mathcal{Q}^k) + \mathbf{Sample}(B \times (1 - z), \mathcal{Q}^{k+1})$ ;  
        Train and evaluate  $\pi_\theta$  on  $\mathcal{M}_{\text{train}}$  via MARL;  
        // Task Expansion  
        expand  $\mathcal{Q}_{\text{sol}}^{\{k, k+1\}}$  and  $\mathcal{Q}_{\text{act}}^{\{k, k+1\}}$  using the value estimates on  $\mathcal{M}_{\text{train}}$ ;  
        // generate novel exploration tasks  
         $\mathcal{M}_{\text{seed}} \leftarrow$  active tasks from  $\mathcal{M}_{\text{train}}$  that just become solved under  $\pi_\theta$  ;  
         $\mathcal{M}_{\text{exp}} \leftarrow$  generate  $B_{\text{exp}}$  exploration task samples from seeding tasks in  $\mathcal{M}_{\text{seed}}$ ;  
        Add exploration tasks  $\mathcal{M}_{\text{exp}}$  to  $\mathcal{Q}_{\text{act}}^{\{k, k+1\}}$ ;  
        // Entity Progression  
        Decay  $z$ ;  
     $k \leftarrow k + 1$ ;  
**until**  $n_k = N$ ;

---

### 3.2 Entity Progression for Massive Multi-Agent Learning

Directly applying the Stein variational update (Eq.(5)) over the relaxation  $z$  of the discrete variable  $n$  results in a uniform expansion over the entire  $N$ -dimensional space of  $z$ , which can be computational challenging when  $N$  is particularly large. We leverage an important inductive bias in cooperative MARL that problems with more agents are generally harder than fewer agents, which implies a monotonic update scheme over the  $z$  space by starting with  $z_{n_0} = 1$ , where  $n_0$  is the smallest possible number of agents, and gradually increasing the probability mass of  $z_k$  for larger  $k$  values.

*Entity Progression* simplifies this procedure further by restricting  $z$  to only have two non-zero entries, i.e.,  $z_k + z_{k'} = 1$  with  $k' > k$ , which suggests a smooth and progressive transition from a simpler problem with  $k$  agents (i.e.,  $z_k = 1$ ) to a harder problem with  $k'$  agents (i.e.,  $z_{k'} = 1$ ). Note that we can increase the agent number incrementally (i.e.,  $k' = k + 1$ ) or at an exponential rate (i.e.,  $k' = 2k$ ), the latter of which is typically more preferred in practice. We remark that entity progression can be directly applied to problems with multiple discrete parameters by viewing  $k$  as a discrete vector.

### 3.3 VACL: A Hierarchical Automatic Curriculum Learning Algorithm

Since the change of discrete variable  $n$  may cause a drastic distribution shift of value functions, instead of using a joint continuous representation unifying both  $n$  and  $\phi$ , VACL combines *Entity Progression* and *Task Expansion* in a hierarchical manner: at the low level, task expansion maintains a separate curriculum  $q^n(\phi)$  for each  $n$  and performs particle-based curriculum update under a fixed  $n$ ; at the high level, entity progression progressively increases  $n$  to  $n'$  by smoothly switching training distribution from  $q^n(\phi)$  to  $q^{n'}(\phi)$ . The full procedure is summarized in Algo. 1.

### 3.4 Connection to Existing Works

We focus on goal-conditioned multi-agent cooperative problems and assume a parameterized task space. This is a common setting in the recent Automatic Curriculum Learning (ACL) literature [4, 25]. Some works also consider a fixed set of tasks [20], which are typically referred to as multi-task learning [34]. The core idea of ACL is to train the policy using tasks with moderate difficulty, i.e., neither too hard nor too easy, which naturally emerges as a consequence of our variational paradigm.

A popular class of ACL methods learns a generative model, such as VAE [26], GAN [9, 7] and Gaussian mixture model [24, 12, 22], over the task space with density concentrated on tasks with

desired difficulties. These methods can be viewed as a generic solution under our variational objective (Eq. 3) by representing  $q(\phi)$  as a generative model and directly optimizing  $\mathcal{L}_2$  whenever the policy is updated. However, it can be extremely sample-inefficient to repeatedly learn  $q(\phi)$  from scratch without leveraging the fact that  $q(\phi)$  is ever-expanding in cooperative MARL problems.

Another type of ACL methods is particle-based, which maintains a set of solvable task samples and gradually expands the set towards the entire space. [10, 11] assume a fixed goal and generate tasks with starting states increasingly far away from the goal; [1, 21] consider a task space over simulator configurations and train from the easiest settings to the hardest for sim-to-real adaptation. These methods are conceptually similar to the *Task Expansion* component in VACL for its particle-based nature and can be viewed as a random search procedure for optimizing  $\mathcal{L}_2$  in a gradient-free manner. Our method provides a principled interpretation of curriculum updates, and achieves superior performance in practice. Besides, [31] learn an individual policy paired with each task for discovering diverse behaviors while our work learns a single goal-conditioned policy for the entire task space; [33] defines the active set as those parts of the task space in which the value function exhibits the largest gradient w.r.t. the task variable and proposes to meta-learn  $f^*(\phi)$  to more effectively expand the curriculum distribution, which is complementary to our work. [14, 15] propose novel optimization methods based on the value function to minimize the KL-Divergence between  $q(\phi)$  and  $p(\phi)$  while our work provides a gradient-based exploration method to expand  $q(\phi)$  to the entire task space.

[18, 32] propose *population curriculum*, which trains entity-invariant policies over training tasks with a growing number of agents. [8] adopt a similar curriculum learning method for Sokoban with an increasing number of boxes. These methods can be viewed as a special case of our *Entity Progression* component by applying a hard transfer from  $z_k = 1$  to  $z_{k'} = 1$  while our method smoothly switches between different number of entities and also yields better empirically performances.

## 4 Experiment

We consider four tasks over two environments, *Simple-Spread* and *Push-Ball* in the multi-agent particle-world environment (MPE) [19], and *Ramp-Use* and *Lock-and-Return* in the hide-and-seek environment (HnS) [2]. Every experiment is repeated over 3 seeds and performed on a desktop machine with one 64-core CPU and one 2080-Ti GPU, which is used for forward action computation and training updates. More details can be found in Appendix.

### 4.1 The Multi-Agent Particle-World Environment

The two tasks are illustrated in Fig. 3. In *Simple-Spread*, there are  $n$  agents and  $n$  landmarks and the agents need to occupy all the landmarks. The agents are penalized for collisions and only receive a positive reward when all the landmarks are covered. In *Push-Ball*, there are  $n$  agents,  $n$  balls and  $n$  landmarks. The agents need to physically push the balls to get every landmark covered. A success reward is given after all landmarks are covered. We scale  $n$  at an exponential rate following [18].

#### 4.1.1 Main Result

We present most important results in this section and more analysis can be found in App. B. We compare VACL with 5 baselines, including (1) multi-agent PPO [35] with uniform task sampling (*Uniform*), (2) population curriculum only (*PC-Unif*), (3) reverse curriculum generation (*RCG*) [10], (4) automatic goal generation (*GoalGAN*) [9], which uses a GAN to generate training tasks, (5) adversarially motivated intrinsic goals (*AMIGo*) [6], which learns a teacher to generate increasingly challenging goals. We test all methods in *Simple-Spread* with  $n = 8$  and *Push-Ball* with  $n = 4$ . For *PC-Unif* and VACL, we start with  $n_0 = 4$  in *Simple-Spread* and  $n_0 = 2$  in *Push-Ball* and then switch to the desired agent number. Evaluation is always performed on the final  $N$ . Results are shown in Fig. 4, where VACL outperforms all baselines with a clear margin. In *Simple-Spread*, all baselines fail to solve the task: Uniform and PC-Unif fail to gain sufficient progress due to sparse rewards; RCG quickly runs out of active tasks; GoalGAN and AMIGo takes an extremely long time to train a good neural goal-generator. In *Push-Ball*, Uniform and PC-Unif learn much slower, RCG quickly converges to a local minimum without further policy improvements, GoalGAN and AMIGo again consumes a large number of samples to make training progress.

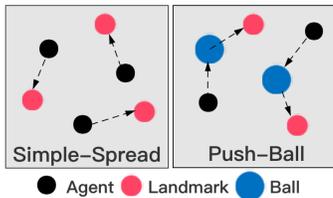


Figure 3: MPE tasks.

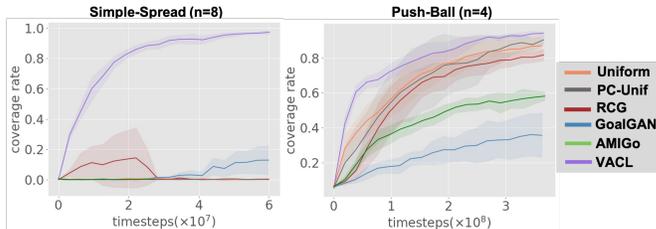


Figure 4: Comparison of VACL and baselines on MPE.

In addition to the baselines above, we also consider the setting of **massive agents** on *Simple-Spread* and compare our VACL method with other coverage numbers in the existing literature. We consider two existing works. The *attentional communication* (ATOC) method [13], which adopts intra-agent communication channels and trains  $n = 50$  and  $n = 100$  agents with *dense* rewards. ATOC is not open-sourced<sup>4</sup>. We re-train VACL on the same environment configuration (i.e., same agent/landmark size and room size) as ATOC for a fair comparison with the number reported in [13]. The other work is evolutionary population curriculum (EPC) [18], which reports the coverage rate on the standard *Simple-Spread* task with  $n = 24$  agents. EPC adopts dense rewards and separate policies for each agent while VACL trains a shared policy with *sparse 0/1* rewards. The results are reported in Tab. 1. VACL achieves a 98% coverage rate with  $n = 100$  agents, which outperforms the highest number ever reported to the best of our knowledge.

Table 1: The best coverage rate ever reported on *Simple-Spread*.

$n$	EPC	ATOC	VACL
24	56.8%	/	<b>97.6%</b>
50	/	92%	<b>98.5%</b>
100	/	89%	<b>98%</b>

#### 4.1.2 Ablation Study

**Choice of seed task:** As implied by Eq. (5) and Eq. (6), VACL explores from the boundary between  $\mathcal{Q}_{act}$  and  $\mathcal{Q}_{sol}$  by taking newly solved tasks as seeds to generate exploration tasks. A naive alternative is to simply take active tasks from  $\mathcal{Q}_{act}$  as seeds (*Exp.Act.*), which is similar to the exploration scheme in the RCG algorithm [10]. Note that exploration from active tasks may possibly produce unsolvable tasks, we consider an additional enhanced variant (*Exp.Act w. eval*), which take active tasks as seeds but also perform value estimates (and training) on those exploration tasks to filter out unsolvable tasks before adding them to  $\mathcal{Q}_{act}$ . Results of these variants are shown in Fig. 5(a). Using active tasks as seeds (*Exp.Act.*) leads to a clear failure since the aggressive exploration brings too many unsolvable tasks to  $\mathcal{Q}_{act}$ . Although additional evaluation on exploration tasks (*Exp.Act.w.eval*) significantly stabilizes training, it still yields much worse training progress than the standard VACL. This suggests that it is critical to explore from the vicinity of solved subspace.

**SVGD-Principled Update:** Our Stein variational paradigm suggests a gradient-based update rule to explore novel tasks in Sec. 3.1.3. However, most existing particle-based algorithms adopts a much simplified gradient-free alternative via uniform random noise, i.e.,  $\phi_{exp} \leftarrow \phi_{seed} + \text{Unif}(-\delta, \delta)$ . To illustrate the effectiveness of our SVGD-principled update rule, we consider a challenging *Hard-Spread* scenario in Fig. 5(b) with 4 agents and 4 landmarks. Obstacles are in black and landmarks can only be placed in the right side. We initialize  $\mathcal{Q}_{act}$  with easy tasks where each landmark has an agent nearby and evaluate the performance of the learning policy on those hardest tasks in Fig. 5(b). We remark that due to the highly constrained task space, it becomes particularly challenging for the ACL method to gradually guide the agents to learn to go through the narrow doors between obstacles. We evaluate the test performances of VACL and the simplified gradient-free version (*Unif. noise*) in Fig. 5(c). Our principled rule produces a stable learning curve with fast convergence while the gradient-free version fails to solve the task.

**Rejection Sampling:** To investigate the effectiveness of rejection-sampling-based exploration (*R.J.*), we turn off rejection-sampling (i.e., strictly following Eq (6)) and conduct experiments on *Hard-Spread* (i.e., highly constrained task space) and *Simple-Spread* (i.e., less constrained task space) in Tab. 2.

<sup>4</sup>We contacted the authors but did not get a response.

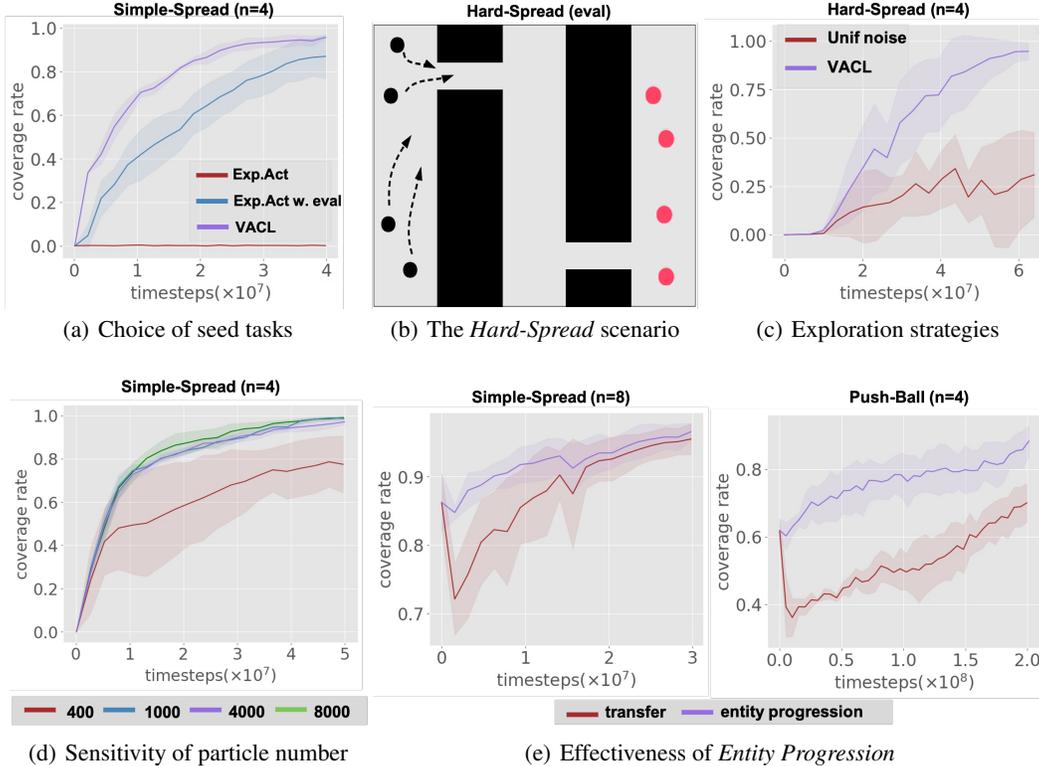


Figure 5: Ablation studies. (a) We evaluate different choices of seed tasks in *Simple-Spread*. (b) We select the hardest tasks to evaluate the performance of different exploration strategies in the *Hard-Spread* scenario, which has a highly constrained task space. (c) We compare our SVGD-principled exploration strategy with random exploration from seed tasks in *Hard-Spread*. (d) We perform experiments in *Simple-Spread* using different numbers of particles to represent  $\mathcal{Q}$ . (e) We compare entity progression with direct entity transfer in standard *Simple-Spread* and *Push-Ball*.

Rejection-sampling-based exploration clearly improves the final performance, particularly in *Hard-Spread*, where purely gradient-based expansion (i.e., Eq.(6)) can hardly discover feasible novel tasks. It is worth mentioning that these feasibility constraints are shared across all the experiments, so all the baselines, including RCG, GoalGAN, and AMIGO, have access to the feasibility check and are able to produce feasible training tasks.

**Sensitivity of Particle Number:** We perform experiments with different numbers of particles in *Simple-Spread* with  $n = 4$  and the results in Fig. 5(d) show that we do need sufficient particles to approximate  $q(\phi)$ . Insufficient particle size (e.g., 400) results in inaccurate estimation and poor performance. The results are robust when particle size is at least 1000. We choose 4000 particles to approximate  $q(\phi)$  in all our experiments.

**Entity Progression:** We compare *entity progression* with naive population curriculum, i.e., directly transfer the trained policy on  $n_{k-1}$  to the next entity size  $n_k$  (*transfer*). We train with the two methods on *Simple-Spread* from  $n_0 = 4$  to  $N = 8$  and on *Push-Ball* from  $n_0 = 2$  to  $N = 4$ . The results are shown in Fig. 5(e). In both two tasks, direct transfer yields a significant performance drop, which we believe due to the drastic training distribution shift. For *entity progression*, due to its smooth transition between the continuous relaxation of  $z_k = 1$  and  $z'_k = 1$ , it produces a much stable training curves. *Entity progression* also yields the much better final performances particularly on the more challenging *Push-Ball* scenario.

Table 2: Ablation study on the rejection-sampling technique in Task Expansion on *Simple-Spread* and *Hard-Spread* with  $n = 4$

coverage rate %	w. R.J.	w.o. R.J.
<i>Simple-Spread</i>	96.0 ± 1.9	94.1 ± 4.9
<i>Hard-Spread</i>	97.2 ± 1.4	0

## 4.2 The Hide-and-Seek Environment

We consider two tasks in the hide-and-seek environment (HnS) as shown in Fig. 6, i.e., a simplified hide-and-seek game called *Ramp-Use* and a multi-agent and sparse-reward variant of the *Lock-and-Return* task from the transfer task suite in [2]. *Ramp-Use* is a single-agent scenario for the ramp use strategy, with 1 ramps, 1 movable seeker and 1 fixed hider in the quadrant room. We need to train a seeker policy to use the ramp to get into the quadrant room for positive rewards. In *Lock-and-Return*, the discrete parameter  $n$  becomes a 2-dimensional vector and each dimension represents the number of agents and boxes respective. Agents need to lock all the boxes and return to their birthplaces. Agents get a reward when *all* boxes are locked, and another success reward when the task is finished, i.e., all boxes locked and all agents back to the birthplaces.

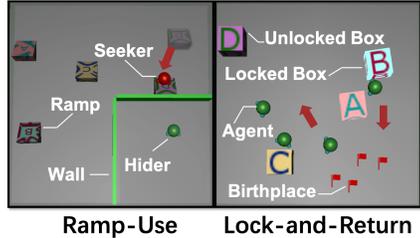


Figure 6: Tasks in the hide-and-seek environment (HnS).

The results of different methods are shown in Tab. 3. We train each algorithm on *Ramp-Use* for 25M timesteps and on *Lock-and-Return* with  $n = (2, 2)$  for 180M timesteps. For *Lock-and-Return* with  $n = (4, 4)$ , we train VACL for a total number of 230M timesteps. We report the average success rate over 3 seeds. VACL is the only algorithm able to solve both two tasks.

Table 3: Results of VACL and baselines in HnS tasks.

		Uniform	RCG	GoalGAN	AMIGo	VACL
Ramp-Use	$n = 1$	15.3% $\pm$ 16.1%	76.7% $\pm$ 2.1%	6.4% $\pm$ 7.5%	40.8% $\pm$ 29.2%	99.7% $\pm$ 0.5%
Lock-and-Return	$n = (2, 2)$	<1%	5.0% $\pm$ 5.1%	<1%	< 2%	97.3% $\pm$ 0.1%
	$n = (4, 4)$	/	/	/	/	97.0% $\pm$ 1.6%

In *Lock-and-Return*, we start with the easy setting of  $n_0 = (2, 2)$  and proceed towards  $N = (4, 4)$ . Note that  $n$  is a 2-dimensional discrete vector in this task, there are multiple paths to transit from the continuous relaxation  $z_{(2,2)} = 1$  to  $z_{(4,4)} = 1$ . One is to double the number of agents first and keep the box number unchanged, and then double the number of boxes, i.e.,  $z_{n_0} = 1 \rightarrow z_{(4,2)} = 1 \rightarrow z_N = 1$  (*agent-first*). The other is to first double the number of boxes and then increase the agent number, i.e.,  $z_{n_0} = 1 \rightarrow z_{(2,4)} = 1 \rightarrow z_N = 1$  (*box-first*). Although both two paths are feasible transitions, they lead to different practical performances. The results are shown in Fig. 7, where we clearly observe that the strategy of increasing agent number first substantially outperforms the box-first strategy. Therefore, we generally suggest to increase the number of agents before objects in entity progression. We remark that since no baseline achieves a success rate above 15% even in the easy case of  $n = (2, 2)$ , we do not evaluate them in the hard setting.

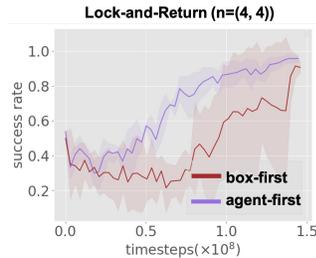


Figure 7: Results of different entity switching strategies.

## 5 Conclusion

In this paper, we propose a novel curriculum learning paradigm following the principle of Stein variational inference and develop an effective algorithm, Variational Automatic Curriculum Learning (VACL), which solves a collection of sparse-reward multi-agent cooperative problems. In this work, we consider cooperative tasks for the clear measurement of training progress. We also remark that VACL assumes a monotonic expansion of  $q(\phi)$ , which only holds in the fully cooperative setting. There are also existing works that study ACL for competitive games [28, 16, 2], which we leave as future work. Our framework tackles an important research problem in the literature while experiments

are conducted on open-source environments with student licences, so we believe our work does not produce any negative impact to the society.

## Acknowledgements

This work is supported by National Science Foundation of China (No. U20A20334, U19B2019, 61832007), National Key R&D Program of China (No.2019YFF0301505), Tsinghua EE Xilinx AI Research Fund and 2030 Innovation Megaprojects of China (Programme on New Generation Artificial Intelligence) Grant No. 2021AAA0150000. We appreciate Jiantao Qiu and Xuefei Ning for their reviews of an early draft. We would like to thank Chao Yu for her support and input during this project. Yi Wu would also thank Bowen Baker and Ingmar Kanitscheider since some of the very initial ideas are inspired by their early discussion at OpenAI. Finally, we thank all anonymous reviewers for their constructive comments.

## References

- [1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik’s cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.
- [2] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2020.
- [3] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.
- [4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.
- [5] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.
- [6] Andres Campero, Roberta Raileanu, Heinrich Kuttler, Joshua B. Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with {amig}o: Adversarially motivated intrinsic goals. In *International Conference on Learning Representations*, 2021.
- [7] Kuan Fang, Yuke Zhu, Silvio Savarese, and Fei-Fei Li. Adaptive procedural task generation for hard-exploration problems. In *International Conference on Learning Representations*, 2021.
- [8] Dieqiao Feng, Carla Gomes, and Bart Selman. Solving hard ai planning instances using curriculum-driven deep reinforcement learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020.
- [9] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.
- [10] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.
- [11] Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 15–21. IEEE, 2019.
- [12] Allan Jabri, Kyle Hsu, Abhishek Gupta, Ben Eysenbach, Sergey Levine, and Chelsea Finn. Unsupervised curricula for visual meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

- [13] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7254–7264. Curran Associates, Inc., 2018.
- [14] Pascal Klink, Hany Abdulsamad, Boris Belousov, and Jan Peters. Self-paced contextual reinforcement learning. In *CoRL*, 2019.
- [15] Pascal Klink, Carlo D’Eramo, Jan Peters, and Joni Pajarinen. Self-paced deep reinforcement learning. In *NeurIPS*, 2020.
- [16] Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019.
- [17] Qiang Liu and Dilin Wang. Stein variational gradient descent: a general purpose bayesian inference algorithm. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2378–2386, 2016.
- [18] Qian Long, Zihan Zhou, Abhinav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. Evolutionary population curriculum for scaling multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2020.
- [19] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.
- [20] Tabet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*, 2019.
- [21] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.
- [22] Steven Morad, Roberto Mecca, Rudra Poudel, Stephan Liwicki, and Roberto Cipolla. Embodied visual navigation with automatic curriculum learning in real environments. *IEEE Robotics and Automation Letters*, 2021.
- [23] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.
- [24] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep RL in continuously parameterized environments. *arXiv preprint arXiv:1910.07224*, 2019.
- [25] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.
- [26] Sebastien Racaniere, Andrew Lampinen, Adam Santoro, David Reichert, Vlad Firoiu, and Timothy Lillicrap. Automated curriculum generation through setter-solver interactions. In *International Conference on Learning Representations*, 2020.
- [27] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. *arXiv preprint arXiv:1906.04279*, 2019.
- [28] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.
- [29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

- [30] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.
- [31] Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International Conference on Machine Learning*, pages 9940–9951. PMLR, 2020.
- [32] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning, 2019.
- [33] Jan Wöhlke, Felix Schmitt, and Herke van Hoof. A performance-based start state curriculum framework for reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1503–1511, 2020.
- [34] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems*, 2020.
- [35] Chao Yu, Akash Velu, Eugene Vinyals, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2021.

## References

- [1] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik’s cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.
- [2] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autotutorials. In *International Conference on Learning Representations*, 2020.
- [3] Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020.
- [4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.
- [5] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.
- [6] Andres Campero, Roberta Raileanu, Heinrich Kuttler, Joshua B. Tenenbaum, Tim Rocktäschel, and Edward Grefenstette. Learning with {amig}o: Adversarially motivated intrinsic goals. In *International Conference on Learning Representations*, 2021.
- [7] Kuan Fang, Yuke Zhu, Silvio Savarese, and Fei-Fei Li. Adaptive procedural task generation for hard-exploration problems. In *International Conference on Learning Representations*, 2021.
- [8] Dieqiao Feng, Carla Gomes, and Bart Selman. Solving hard ai planning instances using curriculum-driven deep reinforcement learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020.
- [9] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, pages 1515–1528. PMLR, 2018.

- [10] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, pages 482–495. PMLR, 2017.
- [11] Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 15–21. IEEE, 2019.
- [12] Allan Jabri, Kyle Hsu, Abhishek Gupta, Ben Eysenbach, Sergey Levine, and Chelsea Finn. Unsupervised curricula for visual meta-reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.
- [13] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7254–7264. Curran Associates, Inc., 2018.
- [14] Pascal Klink, Hany Abdulsamad, Boris Belousov, and Jan Peters. Self-paced contextual reinforcement learning. In *CoRL*, 2019.
- [15] Pascal Klink, Carlo D’Eramo, Jan Peters, and Joni Pajarinen. Self-paced deep reinforcement learning. In *NeurIPS*, 2020.
- [16] Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019.
- [17] Qiang Liu and Dilin Wang. Stein variational gradient descent: a general purpose bayesian inference algorithm. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2378–2386, 2016.
- [18] Qian Long, Zihan Zhou, Abhinav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. Evolutionary population curriculum for scaling multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2020.
- [19] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.
- [20] Tabet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*, 2019.
- [21] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J Pal, and Liam Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.
- [22] Steven Morad, Roberto Mecca, Rudra Poudel, Stephan Liwicki, and Roberto Cipolla. Embodied visual navigation with automatic curriculum learning in real environments. *IEEE Robotics and Automation Letters*, 2021.
- [23] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.
- [24] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep RL in continuously parameterized environments. *arXiv preprint arXiv:1910.07224*, 2019.
- [25] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.
- [26] Sebastien Racaniere, Andrew Lampinen, Adam Santoro, David Reichert, Vlad Firoiu, and Timothy Lillicrap. Automated curriculum generation through setter-solver interactions. In *International Conference on Learning Representations*, 2020.

- [27] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. *arXiv preprint arXiv:1906.04279*, 2019.
- [28] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. In *International Conference on Learning Representations*, 2018.
- [29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- [30] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.
- [31] Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. Enhanced poet: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International Conference on Machine Learning*, pages 9940–9951. PMLR, 2020.
- [32] Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning, 2019.
- [33] Jan Wöhlke, Felix Schmitt, and Herke van Hoof. A performance-based start state curriculum framework for reinforcement learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1503–1511, 2020.
- [34] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems*, 2020.
- [35] Chao Yu, Akash Velu, Eugene Vinyals, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2021.

All the source code can be found at our project website <https://sites.google.com/view/va1-neurips-2021>.

## A Proofs

In order to prove Theorem 1, we introduce the following lemma, which uses Assumption 1.

**Lemma 1.** *Let  $T(\phi) = \phi + \epsilon f(\phi)$  and  $q_{[T]}(\psi)$  the density of  $\psi = T(\phi)$ . Then*

$$\nabla_{\epsilon} \mathcal{L}_2(\pi, q_{[T]})|_{\epsilon=0} = -\mathbb{E}_{\phi \sim q(\phi)} [V(\psi, \pi) \cdot \text{trace}(\nabla_{\phi} \log p(\phi) f(\phi)^{\top} + \nabla_{\phi} f(\phi))] \quad (7)$$

with  $\mathcal{L}_2(\phi, \pi)$  as defined in Eq.(3).

*Proof.* The proof is largely based on [17]. Denote by  $p_{T^{-1}}(\phi)$  the density of  $\phi = T^{-1}(\psi)$  when  $\psi \sim p(\psi)$ , then:

$$\mathcal{L}_2(\pi, q_{[T]}) = \mathbb{E}_{\psi \sim q_{[T]}(\psi)} \left[ V(\psi, \pi) \log \frac{p(\psi)}{q_{[T]}(\psi)} \right] = \mathbb{E}_{\phi \sim q(\phi)} \left[ V(T(\phi), \pi) \log \frac{p_{[T^{-1}]}(\phi)}{q(\phi)} \right] \quad (8)$$

and since  $\epsilon$  only depends on  $T$ , we have:

$$\nabla_{\epsilon} \mathcal{L}_2(\pi, q_{[T]}) = \mathbb{E}_{\phi \sim q(\phi)} \left[ V(T(\phi), \pi) \nabla_{\epsilon} \log p_{[T^{-1}]}(\phi) + \log \frac{p_{[T^{-1}]}(\phi)}{q(\phi)} \nabla_{\epsilon} V(T(\phi), \pi) \right] \quad (9)$$

From Assumption 1,  $V(\phi', \pi)$  is constant within a small vicinity of  $\phi$ ; thus  $\nabla_{\epsilon} V(T(\phi), \pi)|_{\epsilon=0} = 0$ ; hence

$$\nabla_{\epsilon} \mathcal{L}_2(\pi, q_{[T]})|_{\epsilon=0} = \mathbb{E}_{\phi \sim q(\phi)} [V(T(\phi), \pi) \cdot \nabla_{\epsilon} \log p_{[T^{-1}]}(\phi)|_{\epsilon=0}] \quad (10)$$

Define  $s_p(\phi) = \nabla_{\phi} \log p(\phi)$ ; we have

$$\nabla_{\epsilon} \log p_{[T^{-1}]}(\phi) = s_p(T(\phi))^{\top} \nabla_{\epsilon} T(\phi) + \text{trace}((\nabla_{\phi} T(\phi))^{-1} \cdot \nabla_{\epsilon} \nabla_{\phi} T(\phi)). \quad (11)$$

When  $T(\phi) = \phi + \epsilon f(\phi)$  and  $\epsilon = 0$ , we have

$$T(\phi) = \phi, \quad \nabla_{\epsilon} T(\phi) = f(\phi), \quad \nabla_{\phi} T(\phi) = I, \quad \nabla_{\epsilon} \nabla_{\phi} T(\phi) = \nabla_{\phi} f(\phi), \quad (12)$$

where  $I$  is the identity matrix. Therefore:

$$\nabla_{\epsilon} \mathcal{L}_2(\pi, q_{[T]})|_{\epsilon=0} = \mathbb{E}_{\phi \sim q(\phi)} [V(T(\phi), \pi) (\nabla_{\phi} \log p(\phi)^{\top} f(\phi) + \text{trace}(\nabla_{\phi} f(\phi)))] \quad (13)$$

$$= \mathbb{E}_{\phi \sim q(\phi)} [V(T(\phi), \pi) \cdot \text{trace}(\nabla_{\phi} \log p(\phi) f(\phi)^{\top} + \nabla_{\phi} f(\phi))] \quad (14)$$

where the final equality represents an inner product with a trace.  $\square$

**Theorem 1.** *Let  $T(\phi) = \phi + \epsilon f(\phi)$  where  $f$  is an element of some vector-valued RKHS of a positive definite kernel  $k(\phi, \phi') : \Phi \times \Phi \rightarrow \mathbb{R}$ , and  $q_{[T]}$  the density of  $\psi = T(\phi)$  when  $\phi \sim q$ , then*

$$\nabla_f \mathcal{L}_2(q_{[T]}, \pi)|_{f=0} = f^*(\phi), \quad (4)$$

where  $f^*(\cdot) = \mathbb{E}_{\phi' \in \mathcal{Q}} [V(\phi', \pi) (k(\phi', \cdot) \nabla_{\phi'} \log p(\phi') + \nabla_{\phi'} k(\phi', \cdot))]$ .

*Proof.* Let  $\mathcal{H}^d = \mathcal{H} \times \dots \times \mathcal{H}$  be a vector-valued RKHS, and  $F[f]$  be a functional of  $f$ . The gradient  $\nabla_f F[f]$  of  $F[\cdot]$  is a function in  $\mathcal{H}^d$  that satisfies

$$F[f + \epsilon g] = F[f] + \epsilon \langle \nabla_f F[f], g \rangle_{\mathcal{H}^d} + O(\epsilon^2). \quad (15)$$

Define  $F[f] = \mathbb{E}_{\phi \sim q(\phi)} [V(\phi + f(\phi), \pi) (\log p_{[(\phi + f(\phi))^{-1}]}(\phi) - \log q(\phi))]$ , we have

$$F[f + \epsilon g] = \mathbb{E}_{\phi \sim q(\phi)} [V(\phi + f(\phi) + \epsilon g(\phi), \pi) \cdot (\log p_{[(\phi + f(\phi) + \epsilon g(\phi))^{-1}]}(\phi) - \log q(\phi))] \quad (16)$$

$$= \mathbb{E}_{\phi \sim q(\phi)} [V(\phi + f(\phi), \pi) \cdot (\log p(\phi + f(\phi) + \epsilon g(\phi)) - \log q(\phi) + \log \det(I + \nabla_{\phi} f(\phi) + \epsilon \nabla_{\phi} g(\phi)))] \quad (17)$$

where  $V(\phi + f(\phi), \pi) = V(\phi + f(\phi) + \epsilon g(\phi), \pi)$  for  $f \approx 0, \epsilon \approx 0$  from Assumption 1.

Then, we have that:

$$\mathbb{E}_q[V(\phi + f(\phi), \pi) \cdot (\log p(\phi + f(\phi) + \epsilon g(\phi)) - \log p(\phi + f(\phi)))] \quad (18)$$

$$= \epsilon \cdot \mathbb{E}_q[V(\phi + f(\phi), \pi) \cdot \nabla_\phi \log p(\phi + f(\phi)) \cdot g(\phi)] + O(\epsilon^2) \quad (19)$$

$$= \epsilon \cdot \mathbb{E}_q[V(\phi + f(\phi), \pi) \cdot \nabla_\phi \log p(\phi + f(\phi)) \cdot \langle k(\phi, \cdot), g \rangle_{\mathcal{H}^d}] + O(\epsilon^2) \quad (20)$$

$$= \epsilon \cdot \langle \mathbb{E}_q[V(\phi + f(\phi), \pi) \cdot \nabla_\phi \log p(\phi + f(\phi)) \cdot k(\phi, \cdot)], g \rangle_{\mathcal{H}^d} + O(\epsilon^2), \quad (21)$$

where the first equality uses the definition of functional gradient and second equality uses the representation theorem for RKHS. Similarly, we also have that:

$$\mathbb{E}_q[V(\phi + f(\phi), \pi) \cdot (\log \det(I + \nabla_\phi f(\phi) + \epsilon \nabla_\phi g(\phi)) - \log \det(I + \nabla_\phi f(\phi)))] \quad (22)$$

$$= \epsilon \langle \mathbb{E}_q[V(\phi + f(\phi), \pi) \cdot \text{trace}((I + \nabla_\phi f(\phi))^{-1} \cdot \nabla_\phi k(\phi, \cdot))], g \rangle + O(\epsilon^2). \quad (23)$$

Therefore, by combining Equation (21) and Equation (23), we have that

$$F[f + \epsilon g] = F[f] + \epsilon \langle \nabla_f F[f], g \rangle_{\mathcal{H}^d} + O(\epsilon^2), \quad (24)$$

where

$$\nabla_f F[f] = \mathbb{E}_{\phi \in \mathcal{Q}}[V(\phi, \pi) \cdot (k(\phi, \cdot) \nabla_\phi \log p(\phi + f(\phi)) + \text{trace}((I + \nabla_\phi f(\phi))^{-1} \cdot \nabla_\phi k(\phi, \cdot))],$$

and taking  $f = 0$  completes the proof.  $\square$

## B Additional Results

**Pure Task Expansion Results on MPE:** VACL contains entity progression in the result of Fig. 4. To specifically study the performance of task expansion, we exclude entity progression module from VACL and compare with baselines in *Simple-Spread* with  $n = 4$  and *Push-Ball* with  $n = 2$ . For a fair comparison, we also provide additional experiments to combine GoalGAN and AMIGo with the initial knowledge of easy tasks. As show in Fig. 8, VACL without entity progression also outperforms all the baselines in the two environments.

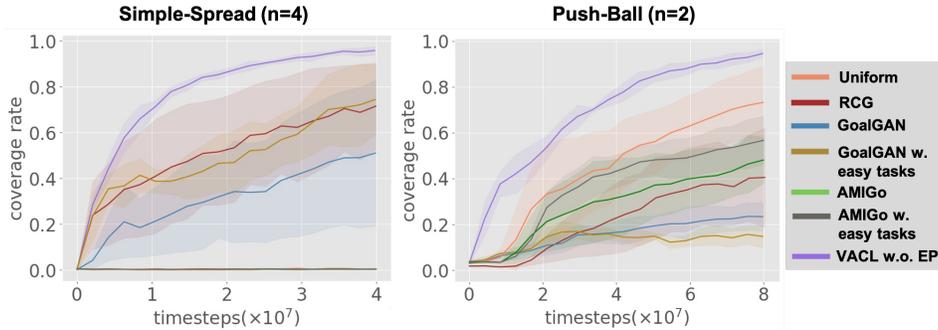


Figure 8: Comparison of baselines and VACL without entity progression on MPE (i.e., task expansion only).

**Additional Results on SVGD-Principled Update:** We additionally conduct experiments to compare VACL with the gradient-free version (*Unif. noise*) in the original *Simple-Spread* with  $n = 4$  and *Push-Ball* with  $n = 2$ . As shown in Fig. 9, VACL is comparable with the variant using gradient-free exploration in *Simple-Spread* and the gap becomes significant in *Push-Ball*, which implies a gradient-based update rule can explore more novel tasks than the simplified gradient-free method in harder scenarios, which is consistent with what we find in the main paper.

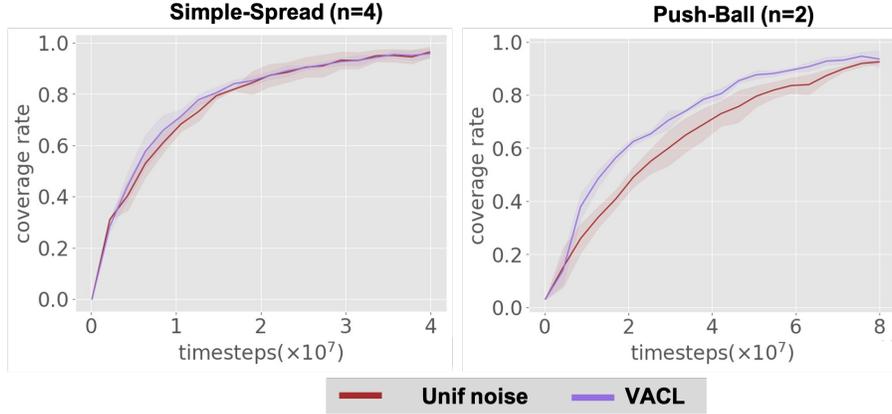


Figure 9: Comparison of VACL and the gradient-free exploration method in *Simple-Spread* and *Push-Ball*

**VACL for Heterogeneous Agents:** We add another experiment domain with heterogeneous agents in *Speaker-Listener* (Fig. 10(a)), which is one of the basic tasks in the MADDPG [19] paper. This task consists of two cooperative agents, a speaker and a listener, and three landmarks with different colors. The speaker and listener obtain +1 reward when the listener covers the correct landmark. However, while the listener can observe the relative position and color of the landmarks, it does not know which landmark it should navigate to. Conversely, the speaker’s observation consists of the correct landmark color, producing a communication output that the listener can observe. Thus, the speaker must learn to output the landmark color based on the motions of the listener. The results in Fig. 10(b) show that our algorithm can also work on the problems with heterogeneous agents.

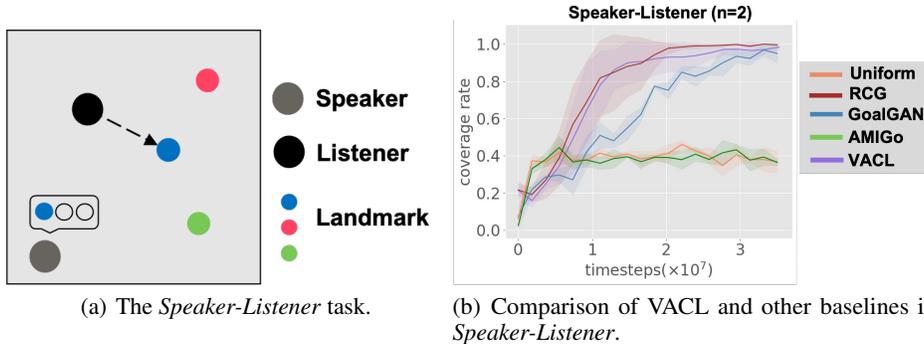


Figure 10: We compare VACL with other baselines in *Speaker-Listener*. The results show that our algorithm can also handle the problems with heterogeneous agents

## C Environment Details

### C.1 Environment Configurations

In *Simple-Spread*, agents get +4 reward when all the landmarks are occupied and get a reward of -1 when any agents collide with each other. The agents are only penalized by collision once at every timestep. In addition, agents and landmarks are randomly generated in a square area with a side length of 6 in the evaluation process. This is the same task as the Cooperative Navigation game in the MADDPG paper [19]. We generate entities in a larger map to increase the difficulty. A larger scene means it is more difficult for agents to get positive reward signals in the sparse-rewards setting. We construct the *Hard-Spread* scenario by adding walls to separate the room into three parts. Agents

can observe the position of landmarks, but the walls are invisible. The *Hard-Spread* mentioned in Fig. 5(b) is a  $10 \times 2$  rectangle.

In *Push-Ball*, there are  $n$  agents,  $n$  balls and,  $n$  landmarks. Agents will get a shared reward  $2/n$  per timestep when any ball occupies one landmark. If all of the landmarks are occupied, agents will get an extra +1 reward. The collision penalty is the same as *Simple-Spread*. Entities are randomly generated in a  $4 \times 4$  square area in the evaluation process. Push-Ball is our first proposed sparse-rewards task based on the physical kernel of the particle world in the MADDPG [19] paper.

For the tasks in the particle-world environment, we evaluate the performances of our algorithm and baselines with the average coverage of landmarks in the last five evaluation steps within every episode.

In *Lock-and-Return*, if all the boxes are locked, agents get +0.2 reward. Moreover, agents can get a success bonus of +1 if they return to their birthplaces after all the boxes are locked. We test our algorithm and baselines on a floor size of 12, with sides twice as the standard hide-and-seek environment. The environment is fully-observable in our setting for simplicity. In this task, we adopt the mean return rate of agents for comparison in the last five evaluation steps.

In *Ramp-Use*, we hope the seeker to learn how to use ramps to enter the enclosed room and catch the hider (Fig. 6). When the hider is spotted, the seeker gets a reward of +1. Otherwise, he gets -1. The environment is fully-observable, and the hider is fixed in the room. We evaluate the performance with the success rate of finding the hider in the last five steps.

## C.2 Definition of $\phi$ and GetEasy(n)

$\phi$  is a vector that contains positions of agents and landmarks in *Simple-Spread*, and positions of agents, balls, and landmarks in *Push-Ball*. In *Lock-and-Return*, it contains positions of agents and boxes (without birthplaces). In *Ramp-Use*, we concatenate positions of the hider, boxes, and the ramp to get  $\phi$ . As for easy cases that generated by **GetEasy**(n), we consider those cases where agents are near landmarks in *Simple-Spread*, and agents, balls, and landmarks are close to each other in *Push-Ball* as easy tasks. In *Lock-and-Return*, easy cases have agents near birthplaces, and boxes randomly placed near them. In *Ramp-Use*, easy cases have the ramp right next to the wall, and agents located next to the ramp.

## D Training Details

### D.1 The Multi-Agent Particle-World Environment

In the two particle-world environments, *Simple-Spread* and *Push-Ball*, we use the same network architecture with a self-attention mechanism (Fig. 11) as EPC [18]. The value network is divided into two parts, which accepts observations of all the agents as input and output the V-value  $V$ . The first part (right in Fig. 11), named *observation encoder*, is used to encode the observation of agent  $j$  into  $f_i(o_j)$ . For each agent  $j$ , *observation encoder* takes observation  $o_j$  as input and then applies an entity encoder for each entity type to obtain embedding vectors of all the entities within this type. We attend the entity embedding of agent  $j$  with all the entities of this type to obtain an attended embedding vector. Then we concatenate all the entity embedding vectors. The observation of agent  $j$ ,  $o_{j,j}$ , is directly forwarded to a fully connected layer. Finally, both the concatenated vectors and the embedded  $o_{j,j}$  are forwarded to a fully connected layer to generate the output,  $f_i(o_j)$ . The output size of entity encoder (green and purple in Fig. 11) and attention layer (orange in Fig. 11) is 64. The second part (left in Fig. 11) accepts the encoding vector  $f_i$  of all the agents. We attend all the agent embedding  $f_i(o_j)$  to the global attention embedding denoted as  $g_i$  (the orange box in the left part in Fig. 11). The  $i$ th agent’s own observation is forwarded to a 1-layer fully connected network to get  $m_i$ . The final layer is a 2-layer fully connected network that takes the concatenation of  $m_i$  and the global attention embedding  $g_i$  and outputs the final V value. The policy network has a similar structure as the *observation encoder*  $f_i(o_i)$ , which uses an attention module over the entities of each type in the observation  $o_i$ . We do not share parameters between the policy and value network.

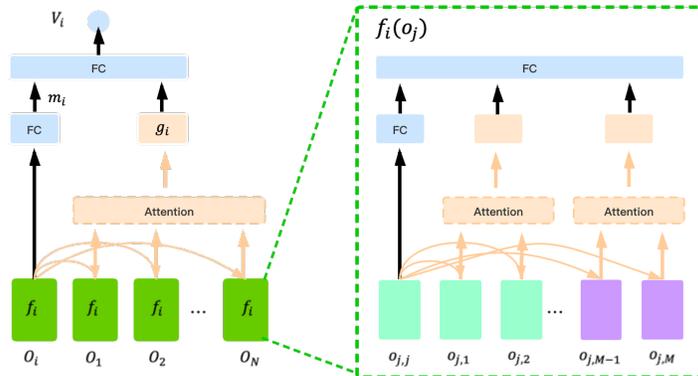


Figure 11: Our population-invariant architecture for the value function.

Agents in *Simple-Spread* and *Push-Ball* are trained by PPO. It is worth mentioning that mini-batch size should be changed when the entity size switches from  $n_{k-1}$  to  $n_k$  due to memory requirements. In *Simple-Spread*, we divide each batch into 2 mini-batch for  $n=4$  and 16 for  $n=8$ . In *Push-Ball*, the number is 2 for  $n=2$  and 8 for  $n=4$ . In addition, easy tasks are different for each training phase. We use the side length  $s$  to represent the tasks generated by **GetEasy**( $n_k$ ). Entities in the initial tasks are randomly generated in a  $s \times s$  square region. We choose  $s = 0.6$  for  $n=4$  and  $s = 2.0$  for  $n=8$  in *Simple-Spread* and *Hard-Spread*. And the side length in *Push-Ball* is  $s = 0.8$  for  $n=2$  and  $s = 1.6$  for  $n=4$ . Note that with value quantization, we categorize  $q(\phi)$  into an active set  $\mathcal{Q}_{\text{act}}$  and a solved set  $\mathcal{Q}_{\text{sol}}$  w.r.t. the task values. Each set uses 2000 particles. Moreover, we define a threshold of convergence. When the evaluation result of the current policy reaches 0.9, we start entity progression. More hyper-parameter settings are shown in Tab. 4.

Table 4: VACL hyper-parameters used in the particle-world environment.

Hyper-parameters	Value
Learning rate	5e-4
Discount rate ( $\gamma$ )	0.99
GAE parameter ( $\lambda$ )	0.95
Adam stepsize	1e-5
Value loss coefficient	1
Entropy coefficient	0.01
PPO clipping	0.2
Parallel threads	500
PPO epochs	15
Reward scale parameter	0.1
Horizon	70 ( <i>Simple-Spread</i> ), 120 ( <i>Push-Ball</i> )
Gradient step ( $\epsilon$ )	0.6( <i>Simple-Spread</i> ), 0.4 ( <i>Push-Ball</i> )
Uniform noise ( $\delta$ )	0.6( <i>Simple-Spread</i> ), 0.4 ( <i>Push-Ball</i> )
RBF kernel ( $h$ )	1
$B_{\text{exp}}$	150

## D.2 The Hide-and-Seek Environment

In the hide-and-seek environment, we use the same attention mechanism as the network in [29] for the policy and critic network. In addition, we train a recurrent policy and split data into chunks in HnS which is similar to Bowen Baker et al. [2]. The size of hidden states is 64. In *Lock-and-Return*, the initialization tasks in the active set are defined in the quadrant room where agents are near the boxes. In *Ramp-Use*, the ramp is placed against the wall, and the seeker is near the ramp for initialization. It is worth mentioning that we divide the floor into grids and define gradient step  $\epsilon$  and uniform noise  $\delta$  with the number of grids in Hns. The grids are  $60 \times 60$  in *Lock-and-Return* and  $30 \times 30$  in *Ramp-Use*. We set  $\epsilon$  and  $\delta$  to 1 in the two environments. More hyper-parameter settings are shown in Tab. 5.

Table 5: VACL hyper-parameters used in the hide-and-seek environment.

Hyper-parameters	Value
Learning rate	5e-4
Discount rate ( $\gamma$ )	0.99
GAE parameter ( $\lambda$ )	0.95
Adam stepsize ( $\epsilon$ )	1e-5
Value loss coefficient	1
Entropy coefficient	0.01
PPO clipping	0.2
Parallel threads	300
PPO epochs	15
Chunk length	40 ( <i>Lock-and-Return</i> ), 10 ( <i>Ramp-Use</i> )
Mini-batch size	1 ( <i>Lock-and-Return</i> ), 2 ( <i>Ramp-Use</i> )
Horizon	60
RBF kernel ( $h$ )	1
$B_{\text{exp}}$	200

### D.3 Additional Implementation Details

**Diversified Queue:** We use two fixed-size queues to maintain  $\mathcal{Q}_{\text{act}}$  and  $\mathcal{Q}_{\text{sol}}$  for computation and memory efficiency. Note that SVGD naturally suggests a diversified data queue, which is our implementation by default. Concretely, We define a distance measure for a task  $\phi$  and a task set  $\mathcal{Q}$  by the average top- $k$  minimum distance between  $\phi$  and each element in  $\mathcal{Q}$ , i.e.,

$$D(\phi, \mathcal{Q}) = \text{mean}(\min(k; \{\|\phi - \phi'\| : \phi' \in \mathcal{Q}\})),$$

where  $k = 5$  in this paper. Then when the size of  $\mathcal{Q}$  exceeds the bound  $L$ , we remove samples w.r.t. their intra-set distance measures:

$$\mathcal{Q} \leftarrow \mathcal{Q} \setminus \min(|\mathcal{Q}| - L; \{D(\phi_j, \mathcal{Q}) : \phi_j \in \mathcal{Q}\}),$$

We set the queue capacity  $L$  to 2000 for both  $\mathcal{Q}_{\text{act}}$  and  $\mathcal{Q}_{\text{sol}}$ . Note that this greedy strategy only works when the overflow size  $|\mathcal{Q}| - L$  is small, e.g., in our setting. When a large number of samples overflows, the distances need to be synced during the deletion process. Finally, we also tested the use of standard FIFO queue and didn't notice any significant performance drop even using a FIFO queue, which can potentially be another practical alternative for even better computational efficiency.

**The Ratio of Active and Solved Tasks:** We follow the convention of curriculum learning to sample more training tasks from  $\mathcal{Q}_{\text{act}}$  for effective training. The ratio of active and solved tasks that we use in the training batch is 95% to 5%. We also compare VACL with a baseline version with uniform sampling from  $\mathcal{Q}_{\text{act}}$  and  $\mathcal{Q}_{\text{sol}}$  (Uniform sampling) in *Simple-Spread* with  $n = 4$  and *Push-Ball* with  $n = 2$ . The results are shown in Fig. 12, where VACL performs slight better which is also consistent with the principle of curriculum learning.

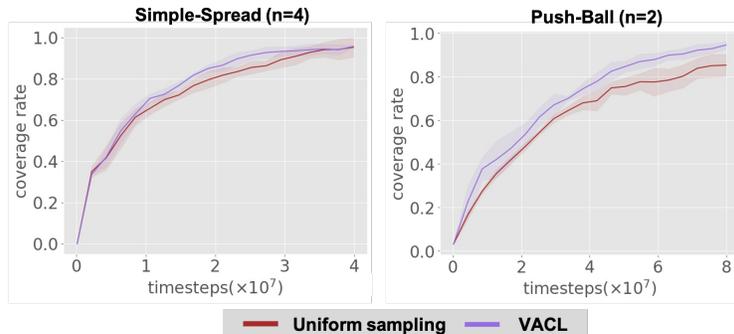


Figure 12: Comparison of VACL and the uniform sampling method in *Simple-Spread* and *Push-Ball*