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

Anonymous Author(s)
Affiliation
Address
email

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.

1 Introduction

Building intelligent agents in complex multi-agent games remains a long-standing challenge in artificial intelligence [21]. 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 [27] 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, 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 MARL setting.

We implement our variational training paradigm through a computationally efficient algorithm, *Variational Automatic Curriculum Learning (VACL)*. VACL consists two components, *task expansion*

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

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 MARL and 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) [15] to expand the task distribution throughout 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 4 physical multi-agent cooperative games, i.e., 2 in the multi-agent particle-world environment (MPE) [17] and 2 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 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), $M(n, \phi) =< n, \phi, \mathcal{S}, \mathcal{A}, \mathcal{O}, O, P, R, s^0_\phi, g_\phi, \gamma >$, parameterized by a discrete variable n (e.g., number of agents[1]) and a continuous parameter $\phi \in \Phi$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the shared action space for each agent and $\mathcal{O}$ is the observation space. $o_i = O(s; i)$ denotes the observation for agent i under state S. s^0_ϕ denotes the initial state and g_ϕ 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 $A = (a_1, \ldots, a_n)$. γ is the discounted factor.

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

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

where $V(n, \phi, \pi_\theta)$ denotes the value function for π_θ 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 π_θ.

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 π_θ 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 ϕ in this subsection. Herein, our objective is to maximize the expectation $\mathcal{L} = \mathbb{E}_{\phi\sim p(\phi)}\left[V(\phi, \pi)\right]$, where $p(\phi)$ is a uniform distribution over feasible values of ϕ. The subscript θ of π 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 $q(\phi)$. The following lower bound for the original objective $\mathcal{L}$ can be derived:

$$\mathcal{L} = \mathbb{E}_{\phi\sim p}[V(\phi, \pi)] = \mathbb{E}_{\phi\sim q}\left[\frac{p(\phi)}{q(\phi)}V(\phi, \pi)\right] = \mathbb{E}_{\phi\sim q}\left[V(\phi, \pi) + \left(\frac{p(\phi)}{q(\phi)} - 1\right)V(\phi, \pi)\right] \tag{2}$$

$$\geq \underbrace{\mathbb{E}_{\phi\sim q(\phi)}\left[V(\phi, \pi)\right]}_{\mathcal{L}_1\text{: policy update}} + \underbrace{\mathbb{E}_{\phi\sim q(\phi)}\left[V(\phi, \pi)\log\frac{p(\phi)}{q(\phi)}\right]}_{\mathcal{L}_2\text{: curriculum update}} \tag{3}$$

[1] 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.

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

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 π. 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 π (policy update) and (2) maximizing $\mathcal{L}_2$ with $q(\phi)$ (curriculum update). We remark that we only optimize π 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 $\pi^\star$ such that $V(\phi, \pi^\star) = 1$ for all ϕ. 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), $\mathcal{L}_2(\pi, q) < \mathcal{L}_2(\pi^\star, p) = 0$.*

Proof. First, we have that $\mathcal{L}_1(\pi^\star, p) = 1$ (definition of $\pi^\star$) and $\mathcal{L}_2(\pi^\star, p) = 0$. 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. □

In other words, if our policy π 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 π, 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. To update the particles, we consider an incremental transform $T(\phi) = \phi + \epsilon f(\phi)$, where $f(\phi)$ is a smooth function and ϵ 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 [25], we introduce an assumption over the generalization ability of the task space.

Assumption 1. *An optimal policy π that is trained over a specific task ϕ has some generalization ability to another goal ϕ' close to ϕ. 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 ϕ.*

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 $k(\phi, \phi') : \Phi \times \Phi \to \mathbb{R}$, and $q_{[T]}$ the density of $\psi = T(\phi)$ when $\phi \sim q$, then*

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

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 is in App. A. While the above statement is very similar to that used for Stein variational gradient descent [15], the update for our case also directly depends on the values $V(\phi', \pi)$, and thus indirectly depends on the current policy. Intuitively, if $V(\phi', \pi)$ is large, then ϕ' will contribute more to the curriculum update, whereas if $V(\phi', \pi) = 0$, then ϕ' 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 Φ, and the task update becomes

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

For example, we take the RBF kernel $k(\phi, \phi') = \exp(-\frac{1}{h}\|\phi - \phi'\|_2^2)$, and then Eq. (5) will drive ϕ 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.*, ϕ' has not been solved at all, then there is no repelling force from ϕ' to ϕ. 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 ϕ can be applied here. Specifically, assuming n denotes the number of agents, we represent n with a categorical distribution $p(n; z) = \mathrm{Cat}(z_1, z_2, \ldots, z_N)$ parametrized by a continuous vector $z = [z_1, \ldots, z_N]$ assuming the maximum possible of agents[2] 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 generalizes across different number of agents, so 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.

3.1 Task Expansion as Approximated Stein Variantional Inference

We present *Task Expansion*, a computation-efficient curriculum update method, which assumes a continuous task representation ϕ and optimizes the variational lower bound $\mathcal{L}_2$ w.r.t. $q(\phi)$.

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 $\mathcal{Q}$.

3.1.2 Value Quantization

Another issue for optimizing $\mathcal{L}_2$ is that whenever π 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, $\mathcal{Q}_{\text{sol}} = \{\phi | V(\phi, \pi) > \sigma_{\max}\}$, an active subspace of moderate values, $\mathcal{Q}_{\text{act}} = \{\phi | \sigma_{\min} \leq V(\phi, \pi) \leq \sigma_{\max}\}$, and the remaining unsolved subspace. Here $\sigma_{\min}$ and $\sigma_{\max}$ 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

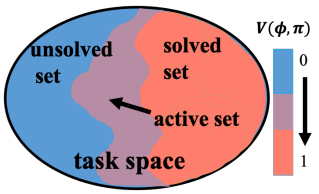


Figure 1: Task space partition.

[2] 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.

estimation becomes no longer necessary. In addition, w.r.t. the curriculum learning principle, when optimizing π under $\mathcal{L}_1$, we can also sample more training tasks from $\mathcal{Q}_{\text{act}}$ for more effective training.

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., $V \geq \sigma_{\min}$) 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 $\mathcal{G}$. Due to the difficulty in value estimation, we again follow the value quantization framework and simplify Eq.(5) as follows:

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

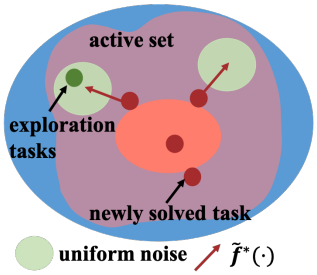


Figure 2: We explore novel tasks in the boundary region between $\mathcal{Q}_{\text{sol}}$ and $\mathcal{Q}_{\text{act}}$

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.

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. 6 illustrates a maze scenario where agents need to navigate towards the landmarks and avoid the black obstacles. A valid task parameterization ϕ includes the positions of agents and landmarks. The feasible subspace of ϕ 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 sampling-based enhancement, which promotes task-space exploration and can be viewed as a zero-th order approximation of the projected gradient.

Rejection-Sampling Exploration: We explore novel tasks ϕ_{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.

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.

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 directly apply 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 ϕ, 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.

Algorithm 1: The VACL Algorithm

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

3.4 Connection to Existing Works

We focus on goal-conditioned multi-agent cooperative problems and assume a paramterized task space. This is a common setting in the recent Automatic Curriculum Learning (ACL) literature [4, 23]. Some works also consider a fixed set of tasks [18], which are typically referred to as multi-task learning [31]. 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 learn a generative model over the task space with density concentrated on tasks with desired difficulties, such as VAE [24], GAN [9, 7] and Gaussian mixture models [22, 12, 20]. 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, 19] 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 to curriculum updates, and achieves superior performance in practice. Besides, [28] 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; [30] propose to meta-learn $f^*(\phi)$ to more effectively expand the curriculum distribution, which is complementary to our work.

[16, 29] 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 as 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) [17], and *Ramp-Use* and *Lock-and-Return* in the hide-and-seek

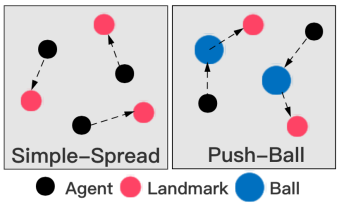


Figure 3: MPE tasks.

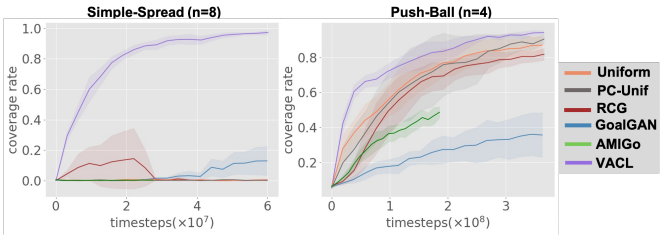


Figure 4: Comparison of VACL and baselines on MPE.

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 are 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 [16].

4.1.1 Main Result

We compare *VACL* with 5 baselines, including (1) multi-agent PPO 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[3] again consumes a large number of samples to make training progress.

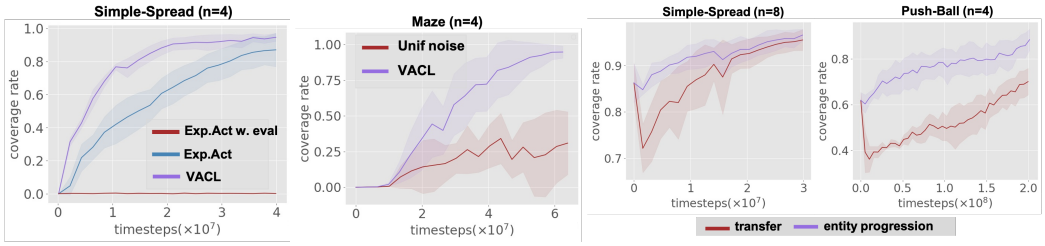
Table 1: The best coverage rate ever reported on *Simple-Spread*.

n	EPC	ATOC	VACL
24	56.8%	/	**97.6%**
50	/	92%	**98.5%**
100	/	89%	**98%**

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[4]. 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) [16], 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.

[3] We do not finish the training of AMIGo in *Push-Ball* before the deadline, but the current results show that AMIGo is converging much slower than our VACL method.

[4] We contacted the authors but did not get a response.



(a) Choice of seed tasks (b) Exploration strategies (c) Effectiveness of entity progression

Figure 5: Ablation studies. (a) We evaluate difference choices of seed tasks in *Simple-Spread*; (b) We compare our SVGD-principled exploration strategy with random exploration from seed tasks in the *Hard-Spread* scenario (Fig. 6), which has a highly constrained task space; (c) We compare entity progression with direct transfer in standard *Simple-Spread* and *Push-Ball*.

4.1.2 Ablation Study

Choice of seed task: As implied by Eq. (5) and Eq. (6), VACL explores from the boundary between $\mathcal{Q}_{\text{act}}$ and $\mathcal{Q}_{\text{sol}}$ by taking newly solved tasks as seeds to generate exploration tasks. A naive alternative is to simply take active tasks from $\mathcal{Q}_{\text{act}}$ 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 $\mathcal{Q}_{\text{act}}$. 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 $\mathcal{Q}_{\text{act}}$. 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., $\phi_{\text{exp}} \leftarrow \phi_{\text{seed}} + \text{Unif}(-\delta, \delta)$. To illustrate the effectiveness of our SVGD-principled update rule, we consider a challenging *Hard-Spread* scenario in Fig. 6 with 4 agents and 4 landmarks. Obstacles are in black and landmarks can only be placed in the right side. We initialize $\mathcal{Q}_{\text{act}}$ with easy tasks where each landmark has an agent nearby and evaluate the performance of the learning policy on those hardest tasks in Fig. 6. 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(b). Our principled rule produces a stable learning curve with fast convergence while the gradient-free version fails to solve the task.

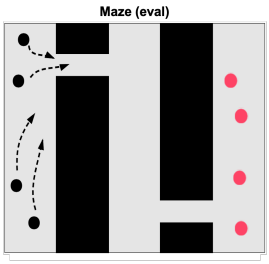


Figure 6: The *Hard-Spread* scenario.

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(c). 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: Results of VACL and baselines in HnS tasks.

		Uniform	RCG	GoalGAN	AMIGo	VACL
Ramp-Use	$n = 1$	$42.8\% \pm 35.4\%$	$31.5\% \pm 33.7\%$	$1.0\% \pm 0.8\%$	$47.2\% \pm 10.3\%$	$93.3\% \pm 5.4\%$
Lock-and-Return	$n = (2, 2)$	<1%	$5.0\% \pm 5.1\%$	<1%	$< 2\%$	$97.3\% \pm 0.1\%$
	$n = (4, 4)$	/	/	/	/	$97.0\% \pm 1.6\%$

4.2 The Hide-and-Seek Environment

We consider two tasks in the hide-and-seek environment (HnS) as shown in Fig. 7, 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 of 1 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.

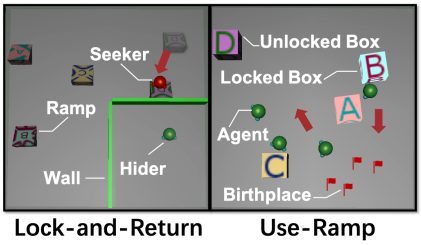


Figure 7: Tasks in the hide-and-seek environment (HnS).

The results of different methods are shown in Tab. 2. We train each algorithm on *Ramp-Use* for 80M 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.

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 transition 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., $z_{n_0} = 1 \to z_{(4,2)} = 1 \to z_N = 1$ (*agent-first*). The other is to first double the number of boxes and then increase the agent number, i.e., $z_{n_0} = 1 \to z_{(2,4)} = 1 \to z_N = 1$ (*box-first*). Although both two paths are feasible transitions, they lead to different practical performances. The results are shown in Fig. 8, 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.

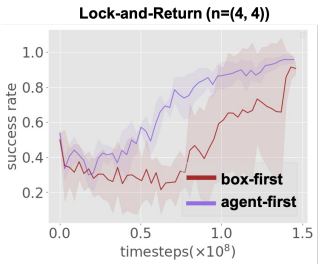


Figure 8: Results of different entity switching strategies.

5 Conclusion

In this paper, we propose a novel curriculum learning paradigm following the principle of Stein variational inference and developed a effective algorithm, Varitional 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 also study ACL for competitive games [26, 14, 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.

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] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*, 2019.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. *arXiv preprint arXiv:1906.04279*, 2019.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. In *Advances in Neural Information Processing Systems*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See conclusion.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See appendix.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]