Accelerate Multi-Agent Reinforcement Learning in Zero-Sum Games with Subgame Curriculum Learning

Jiayu Chen[1*], Zelai Xu[1*], Yunfei Li[1], Chao Yu[1], Jiaming Song[2], Huazhong Yang[1], Fei Fang[3], Yu Wang[1†], Yi Wu[1,4†]

[1]Tsinghua University
[2]Luma AI
[3]Carnegie Mellon University
[4]Shanghai Qi Zhi Institute
{jia768167535, zelai.eecs, jxwuyi}@gmail.com

Abstract

Learning Nash equilibrium (NE) in complex zero-sum games with multi-agent reinforcement learning (MARL) can be extremely computationally expensive. Curriculum learning is an effective way to accelerate learning, but an under-explored dimension for generating a curriculum is the difficulty-to-learn of the *subgames* – games induced by starting from a specific state. In this work, we present a novel subgame curriculum learning framework for zero-sum games. It adopts an adaptive initial state distribution by resetting agents to some previously visited states where they can quickly learn to improve performance. Building upon this framework, we derive a subgame selection metric that approximates the squared distance to NE values and further adopt a particle-based state sampler for subgame generation. Integrating these techniques leads to our new algorithm, *Subgame Automatic Curriculum Learning* (SACL), which is a realization of the subgame curriculum learning framework. SACL can be combined with any MARL algorithm such as MAPPO. Experiments in the particle-world environment and Google Research Football environment show SACL produces much stronger policies than baselines. In the challenging hide-and-seek quadrant environment, SACL produces all four emergent stages and uses only half the samples of MAPPO with self-play. The project website is at https://sites.google.com/view/sacl-rl.

Introduction

Applying reinforcement learning (RL) to zero-sum games has led to enormous success, with trained agents defeating professional humans in Go (Silver et al. 2016), StarCraft II (Vinyals et al. 2019), and Dota 2 (Berner et al. 2019). To find an approximate Nash equilibrium (NE) in complex games, these works often require a tremendous amount of training resources including hundreds of GPUs and weeks or even months of time. The unaffordable cost prevents RL from more real-world applications beyond these flagship projects supported by big companies and makes it important to develop algorithms that can learn close-to-equilibrium strategies in a substantially more efficient manner.

One way to accelerate training is curriculum learning – training agents in tasks from easy to hard. Many existing works in solving zero-sum games with MARL generate a curriculum by choosing whom to play with. They often use self-play to provide a natural policy curriculum as the agents are trained against increasingly stronger opponents (Bansal et al. 2018; Baker et al. 2020). The self-play framework can be further extended to population-based training (PBT) by maintaining a policy pool and iteratively training new best responses to mixtures of previous policies (McMahan, Gordon, and Blum 2003; Lanctot et al. 2017). Such a policy-level curriculum generation paradigm is very different from the paradigm commonly used in goal-conditioned RL (Matiisen et al. 2019; Portelas et al. 2020). Most curriculum learning methods for goal-conditioned problems directly reset the goal or initial states for each training episode to ensure the current task is of suitable difficulty for the learning agent. In contrast, the policy-level curriculum in zero-sum games only provides increasingly stronger opponents, and the agents are still trained by playing the full game starting from a fixed initial state distribution, which is often very challenging.

In this paper, we propose a general subgame curriculum learning framework to further accelerate MARL training for zero-sum games. It leverages ideas from goal-conditioned RL. Complementary to policy-level curriculum methods like self-play and PBT, our framework generates subgames (i.e., games induced by starting from a specific state) with growing difficulty for agents to learn and eventually solve the full game. We provide justifications for our proposal by analyzing a simple iterated Rock-Paper-Scissors game. We show that in this game, vanilla MARL requires exponentially many samples to learn the NE. However, by using a buffer to store the visited states and choosing an adaptive order of state-induced subgames to learn, the NE can be learned with linear samples.

A key challenge in our framework is to choose which subgame to train on. This is non-trivial in zero-sum games since there does not exist a clear progression metric like the success rate in goal-conditioned problems. While the squared difference between the current state value and the NE value can measure the progress of learning, it is impossible to calculate this value during training as the NE is generally un-

[*]These authors contributed equally.
[†]Corresponding Authors.

known. We derive an alternative metric that approximates the squared difference with a bias term and a variance term. The bias term measures how fast the state value changes and the variance term measures how uncertain the current value is. We use the combination of the two terms as the sampling weights for states and prioritize subgames with fast change and high uncertainty. Instantiating our framework with the state selection metric and a non-parametric subgame sampler, we develop an automatic curriculum learning algorithm for zero-sum games, i.e., *Subgame Automatic Curriculum Learning* (SACL). SACL can adopt any MARL algorithm as its backbone and preserve the overall convergence property. In our implementation, we choose the MAPPO algorithm (Yu et al. 2021) for the best empirical performances.

We first evaluate SACL in the Multi-Agent Particle Environment and Google Research Football, where SACL learns stronger policies with lower exploitability than existing MARL algorithms for zero-sum games given the same amount of environment interactions. We then stress-test the efficiency of SACL in the challenging hide-and-seek environment. SACL leads to the emergence of all four phases of different strategies and uses 50% fewer samples than MAPPO with self-play.

Preliminary

Markov Game

A Markov game (Littman 1994) is defined by a tuple $\mathcal{MG} = (\mathcal{N}, \mathcal{S}, \boldsymbol{\mathcal{A}}, P, \boldsymbol{R}, \gamma, \rho)$, where $\mathcal{N} = \{1, \cdots, N\}$ is the set of agents, $\mathcal{S}$ is the state space, $\boldsymbol{\mathcal{A}} = \Pi_{i=1}^{N} \mathcal{A}_i$ is the joint action space with $\mathcal{A}_i$ being the action space of agent i, $P : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \Delta(\mathcal{S})$ is the transition probability function, $\boldsymbol{R} = (R_1, \cdots, R_N) : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}^n$ is the joint reward function with R_i being the reward function for agent i, γ is the discount factor, and ρ is the distribution of initial states. Given the state s and the joint action $\boldsymbol{a} = (a_1, \cdots, a_N)$, the game moves to the next state s' with probability $P(s'|s, \boldsymbol{a})$ and agent i receives a reward $R_i(s, \boldsymbol{a})$.

For infinite-horizon Markov games, a subgame $\mathcal{MG}(s)$ is defined as the Markov game induced by starting from state s, i.e., $\rho(s) = 1$. Selecting subgames is therefore equivalent to setting the Markov game's initial states. The subgames of finite-horizon Markov games are defined similarly and have an additional variable to denote the current step t.

We focus on two-player zero-sum Markov games, i.e., $N = 2$ and $R_1(s, \boldsymbol{a}) + R_2(s, \boldsymbol{a}) = 0$ for all state-action pairs. We use the subscript i to denote variables of player i and the subscript $-i$ to denote variables of the player other than i. Each player uses a policy $\pi_i : \mathcal{S} \to \mathcal{A}_i$ to produce actions and maximize its own accumulated reward. Given the joint policy $\boldsymbol{\pi} = (\pi_1, \pi_2)$, each player's value function of state s and Q-function of state-action pair $(s, \boldsymbol{a})$ are

$$V_i^{\boldsymbol{\pi}}(s) = \mathbb{E}\Big[\sum_t \gamma^t R_i(s^t, \boldsymbol{a}^t) \Big| s^0 = s\Big], \tag{1}$$

$$Q_i^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = \mathbb{E}\Big[\sum_t \gamma^t R_i(s^t, \boldsymbol{a}^t) \Big| s^0 = s, \boldsymbol{a}^0 = \boldsymbol{a}\Big]. \tag{2}$$

The solution concept of two-player zero-sum Markov games is Nash equilibrium (NE), a joint policy where no player can get a higher value by changing the policy alone.

Definition 1 (NE). *A joint policy $\boldsymbol{\pi}^* = (\pi_1^*, \pi_2^*)$ is a Nash equilibrium of a Markov game if for all initial states s^0 with $\rho(s^0) > 0$, the following condition holds*

$$\pi_i^* = \arg\max_{\pi_i} V_i^{(\pi_i, \pi_{-i}^*)}(s^0),\ \forall i \in \{1, 2\}. \tag{3}$$

We use $V_i^*(\cdot)$ to denote the NE value function of player i and $Q_i^*(\cdot, \cdot)$ to denote the NE Q-function of player i, and the following equations hold by definition and the minimax nature of zero-sum games.

$$V_i^*(s) = \max_{\pi_i} \min_{\pi_{-i}} \mathbb{E}_{\boldsymbol{a} \sim \boldsymbol{\pi}(\cdot|s)} \left[Q_i^*(s, \boldsymbol{a})\right], \tag{4}$$

$$Q_i^*(s, \boldsymbol{a}) = R_i(s, \boldsymbol{a}) + \gamma \cdot \mathbb{E}_{s' \sim P(\cdot|s, \boldsymbol{a})} \left[V_i^*(s')\right]. \tag{5}$$

MARL Algorithms in Zero-Sum Games

MARL methods have been applied to zero-sum games tracing back to the TD-Gammon project (Tesauro 1995). A large body of work (Zinkevich et al. 2007; Brown et al. 2019; Steinberger, Lerer, and Brown 2020; Gruslys et al. 2020) is based on regret minimization, and a well-known result is that the average of policies produced by self-play of regret-minimizing algorithms converges to the NE policy of zero-sum games (Freund and Schapire 1996). Another notable line of work (Littman 1994; Heinrich, Lanctot, and Silver 2015; Lanctot et al. 2017; Perolat et al. 2022) combines RL algorithms with game-theoretic approaches. These works typically use self-play or population-based training to collect samples and then apply RL methods like Q-learning (Watkins and Dayan 1992) and PPO (Schulman et al. 2017) to learn the NE value functions and policies, and have recently achieved great success (Silver et al. 2016; Jaderberg et al. 2018; Vinyals et al. 2019; Berner et al. 2019).

For the analysis in the next section, we introduce a classic MARL algorithm named minimax-Q learning (Littman 1994) that extends Q-learning to zero-sum games. Initializing functions $Q_i(\cdot, \cdot)$ with zero values, minimax-Q uses an exploration policy induced by the current Q-functions to collect a batch of samples $\{(s^t, \boldsymbol{a}^t, r_i^t, s^{t+1})\}_{t=0}^{T}$ and uses these samples to update the Q-functions by

$$\begin{aligned} Q_i(s^t, \boldsymbol{a}^t) \leftarrow (1 - \alpha) \cdot Q_i(s^t, \boldsymbol{a}^t) + \\ \alpha \cdot \big(r_i^t + \gamma \cdot \max_{\pi_i} \min_{\pi_{-i}} \mathbb{E}_{\boldsymbol{a} \sim \boldsymbol{\pi}(\cdot|s)} \left[Q_i(s^{t+1}, \boldsymbol{a})\right]\big), \end{aligned} \tag{6}$$

where α is the learning rate. This sample-and-update process continues until the Q-functions converge. Under the assumptions that the state-action sets are discrete and finite and are visited an infinite number of times, it is proved that the stochastic updates by Eq. (6) lead to the NE Q-functions (Szepesvári and Littman 1999).

A Motivating Example

In this section, we show by a simple illustrative example that vanilla MARL methods like minimax-Q require exponentially many samples to derive the NE. However, if we can dynamically set the initial state distribution and induce an

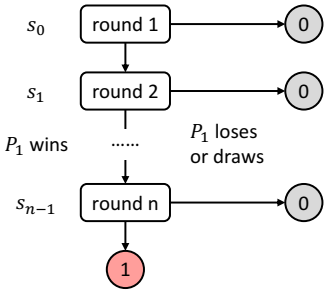


Figure 1: Illustration of the $RPS(n)$ game.

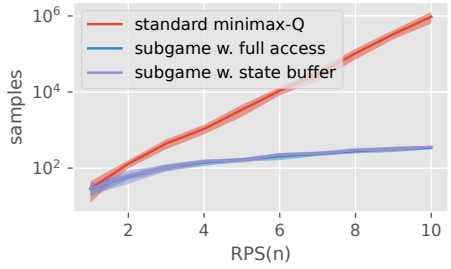


Figure 2: Number of samples used to learn the NE Q-values of $RPS(n)$ games.

appropriate order of subgames to learn, the sample complexity can be substantially reduced from exponential to linear. Such an observation motivates our proposed algorithm described in later sections.

Iterated Rock-Paper-Scissors Game

We introduce an iterated variant of the Rock-Paper-Scissor (RPS) game, denoted as $RPS(n)$. As shown in Fig. 1, P_1 and P_2 play the RPS game for up to n rounds. If P_1 wins all rounds, it gets a reward of 1 and P_2 gets a reward of -1. If P_1 loses or draws in any round, the game ends immediately without playing the remaining rounds and both players get zero rewards. Note that the $RPS(n)$ game is different from playing the RPS game repeatedly for n times because players can play less than n rounds and they only receive a non-zero reward if P_1 wins in all rounds. We use s_k to denote the state where players have already played k RPS games and are at the $k+1$ round. It is easy to verify that the NE policy for both players is to play Rock, Paper, or Scissors with equal probability at each state. Under this joint NE policy, P_1 can win one RPS game with $1/3$ probability, and the probability for P_1 to win all n rounds and get a non-zero reward is $1/3^n$.

Consider using standard minimax-Q learning to solve the $RPS(n)$ game. With Q-functions initialized to zero, we execute the exploration policy to collect samples and perform the update in Eq. (6). Note all state-actions pairs are required to be visited to guarantee convergence to the NE. Therefore, in this sparse-reward game, random exploration will clearly take $\mathcal{O}(3^n)$ steps to get a non-zero reward. Moreover, even if the exploration policy is perfectly set to the NE policy, the probability for P_1 to get the non-zero reward by winning all RPS games is still $\mathcal{O}(1/3^n)$, requiring at least $\mathcal{O}(3^n)$ samples to learn the NE Q-values of the $RPS(n)$ game.

Algorithm 1: Subgame curriculum learning

Input: state sampler $\text{oracle}(\cdot)$.
Initialize policy $\boldsymbol{\pi}$;
repeat
  Sample $s^0 \sim \text{oracle}(\mathcal{S})$;
  Rollout $\boldsymbol{\pi}$ in $\mathcal{MG}(s^0)$;
  Train $\boldsymbol{\pi}$ via MARL;
until $\boldsymbol{\pi}$ *converges*;
Output: final policy $\boldsymbol{\pi}$.

From Exponential to Linear Complexity

An important observation is that the states in later rounds become exponentially rare in the samples generated by starting from the fixed initial state. If we can directly reset the game to these states and design a smart order of minimax-Q updates on the subgames induced by these states, the NE learning can be accelerated significantly. Note that $RPS(n)$ can be regarded as the composition of n individual $RPS(1)$ games, a suitable order of learning would be from the easiest subgame $RPS(1)$ starting from state s_{n-1} to the full game $RPS(n)$ starting from state s_0. Assuming we have full access to the state space, we first reset the game to s_{n-1} and use minimax-Q to solve subgame $RPS(1)$ with $\mathcal{O}(1)$ samples. Given that the NE Q-values of $RPS(k)$ are learned, the next subgame $RPS(k+1)$ is equivalent to an $RPS(1)$ game where the winning reward is the value of state s_{n-k}. By sequentially applying minimax-Q to solve all n subgames from $RPS(1)$ to $RPS(n)$, the number of samples required to learn the NE Q-values is reduced substantially from $\mathcal{O}(3^n)$ to $\mathcal{O}(n)$.

In practice, we usually do not have access to the entire state space and cannot directly start from the last subgame $RPS(1)$. Instead, we can use a buffer to store all visited states and gradually span the state space. By resetting games to the newly visited states, the number of samples required to cover the full state space is still $\mathcal{O}(n)$, and we can then apply minimax-Q from $RPS(1)$ to $RPS(n)$. Therefore, the total number of samples is still $\mathcal{O}(n)$. We validate our analysis by running experiments on $RPS(n)$ games for $n = 1, \cdots, 10$ and the results averaged over ten seeds are shown in Fig. 2. It can be seen that the sample complexity reduces from exponential to linear by running minimax-Q over a smart order of subgames, and the result of using a state buffer in practice is comparable to the result with full access. The detailed analysis can be found in Appendix A.1[1].

Method

The motivating example suggests that NE learning can be largely accelerated by running MARL algorithms in a smart order over states. Inspired by this insight, we present a general framework to accelerate NE learning in zero-sum games by training over a curriculum of subgames. We further propose two practical techniques to instantiate the framework and present the overall algorithm.

[1]Appendix can be found at http://arxiv.org/abs/2310.04796.

Algorithm 2: Subgame Automatic Curriculum Learning (SACL)

Input: state buffers $\mathcal{M}$ with capacity K, probability p to sample initial state from the state buffer.
Randomly initialize policy π_i and value function V_i for player $i = 1, 2$;
repeat
  $V_i' \leftarrow V_i,\ i = 1, 2$;
  `// Select subgame and train policy.`
  for *each parallel environment* **do**
    Sample $s^0 \sim \text{sampler}(\mathcal{M})$ with probability p, else $s^0 \sim \rho(\cdot)$;
    Rollout in $\mathcal{MG}(s^0)$ and collect samples;
  Train $\{\pi_i, V_i\}_{i=1}^2$ via MARL;
  `// Compute weight by Eq. (10) and update state buffer.`
  $\tilde{w}^t \leftarrow \alpha \cdot \mathbb{E}[\tilde{V}_i(s^t) - \tilde{V}_i'(s^t)]^2 + \text{Var}(\{\tilde{V}_i(s^t)\}_{i=1}^2)$, $t = 0, \cdots, T$;
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{(s^t, \tilde{w}^t)\}_{t=0}^T$;
  if $\|\mathcal{M}\| > K$ **then**
    $\mathcal{M} \leftarrow \text{FPS}(\mathcal{M}, K)$;
until (π_1, π_2) *converges*;
Output: final policy (π_1, π_2).

Subgame Curriculum Learning

The key issue of the standard sample-and-update framework is that the rollout trajectories always start from the fixed initial state distribution ρ, so visiting states that are most critical for efficient learning can consume a large number of samples. To accelerate training, we can directly reset the environment to those critical states. Suppose we have an oracle state sampler oracle$(\cdot)$ that can initiate suitable states for the current policy to learn, i.e., generate appropriate induced subgames, we can derive a general-purpose framework in Alg. 1, which we call subgame curriculum learning. Note that this framework is compatible with any MARL algorithm for zero-sum Markov games.

A desirable feature of subgame curriculum learning is that it does not change the convergence property of the backbone MARL algorithm, as discussed below.

Proposition 1. *If all initial states s^0 with $\rho(s^0) > 0$ are sampled infinitely often, and the backbone MARL algorithm is guaranteed to converge to an NE in zero-sum Markov games, then subgame curriculum learning also produces an NE of the original Markov game.*

The proof can be found in Appendix A.2. Note that such a requirement is easy to satisfy. For example, given any state sampler oracle$(\cdot)$, we can construct a valid mixed sampler by sampling from oracle$(\cdot)$ for probability $0 < p < 1$ and sampling from ρ for probability $1 - p$.

Remark. With a given state sampler, the only requirement of our subgame curriculum learning framework is that the environment can be reset to a desired state to generate the induced game. This is a standard assumption in the curriculum learning literature (Florensa et al. 2018; Matiisen et al. 2019; Portelas et al. 2020) and is feasible in many RL environments. For environments that do not support this feature, we can simply reimplement the reset function to make them compatible with our framework.

Subgame Sampling Metric

A key question is how to instantiate the oracle sampler, i.e., *which subgame should we train on for faster convergence*? Intuitively, for a particular state s, if its value has converged to the NE value, that is, $V_i(s) = V_i^*(s)$, we should no longer train on the subgame induced by it. By contrast, if the gap between its current value and the NE value is substantial, we should probably train more on the induced subgame. Thus, a simple way is to use the squared difference of the current value and the NE value as the weight for a state and sample states with probabilities proportional to the weights. Concretely, the state weight can be written as

$$w(s) = \frac{1}{2}\sum_{i=1}^{2}(V_i^*(s) - V_i(s))^2 \tag{7}$$

$$= \mathbb{E}_i\left[(V_1^*(s) - \tilde{V}_i(s))^2\right] \tag{8}$$

$$= \mathbb{E}_i\left[V_1^*(s) - \tilde{V}_i(s)\right]^2 + \text{Var}_i\left[V_1^*(s) - \tilde{V}_i(s)\right], \tag{9}$$

where $\tilde{V}_1(s) = V_1(s)$ and $\tilde{V}_2(s) = -V_2(s)$. The second equality holds because the game is zero-sum and $V_2^*(s) = -V_1^*(s)$. With random initialization and different training samples, $\{\tilde{V}_i\}_{i=1}^2$ can be regarded as an ensemble of two value functions, and the weight $w(s)$ becomes the expectation over the ensemble. The last equality further expands the expectation to a bias term and a variance term, and we sample state with probability $P(s) = w(s)/\sum_{s'} w(s')$. For the motivating example of $RPS(n)$ game, the NE value decreases exponentially from the last state s_{n-1} to the initial state s_0. With value functions initialized close to zero, the prioritized subgames throughout training will move gradually from the last round to the first round, which is approximately the optimal order.

However, Eq. (9) is very hard to compute in practice because the NE value is generally unknown. Inspired by Eq. (9), we propose the following alternative state weight

$$\tilde{w}(s) = \alpha \cdot \mathbb{E}_i\left[\tilde{V}_i^{\pi_i^{(t)}}(s) - \tilde{V}_i^{\pi_i^{(t-1)}}(s)\right]^2 + \text{Var}_i\left[\tilde{V}_i(s)\right], \tag{10}$$

which takes a hyperparameter α and uses the difference between two consecutive value function checkpoints instead of the difference between the NE value and the current value in Eq. (9). The first term in Eq. (10) measures how fast the value functions change over time. If this term is large, the value functions are changing constantly and still far from the NE value; if this term is marginal, the value functions are probably close to the converged NE value. The second term in Eq. (10) measures the uncertainty of the current learned values and is the same as the variance term in Eq. (9) because $V_1^*(s)$ is a constant. If $\alpha = 1$, Eq. (10) approximates Eq. (9) as t increases. It is also possible to train an ensemble of value functions for each player to further improve the empirical performance. More analysis can be found in Appendix A.3.

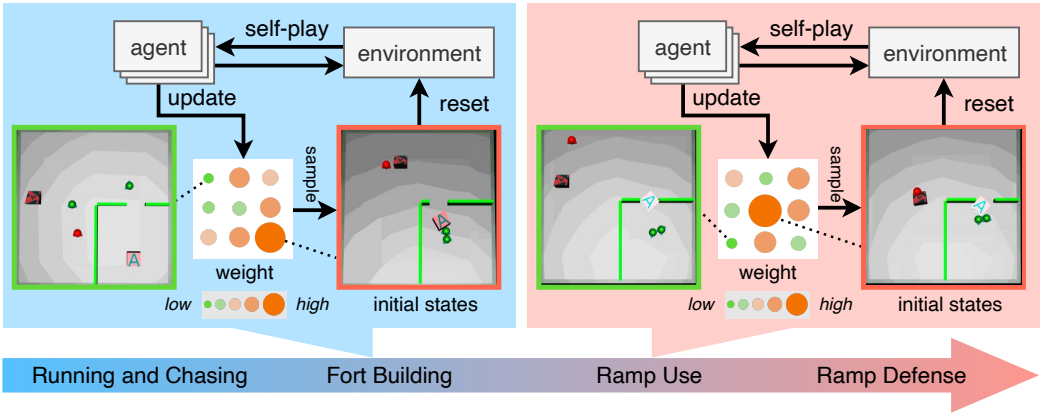


Figure 3: Illustration of SACL in the hide-and-seek environment. In the Fort Building stage, the states with hiders near the box have high weights (red) and agents can easily learn to build a fort by practicing on these subgames, while the states with randomly spawned hiders have low weights (green) and contribute less to learning.

Since Eq. (10) does not require the unknown NE value to compute, it can be used in practice as the weight for state sampling and can be implemented for most MARL algorithms. By selecting states with fast value change and high uncertainty, our framework prioritizes subgames where agents' performance can quickly improve through learning.

Particle-based Subgame Sampler

With the sample weight at hand, we can generate subgames by sampling initial states from the state space. But it is impractical to sample from the entire space which is usually unavailable and can be exponentially large for complex games. Typical solutions include training a generative adversarial network (GAN) (Dendorfer, Osep, and Leal-Taixé 2020) or using a parametric Gaussian mixture model (GMM) (Portelas et al. 2020) to generate states for automatic curriculum learning. However, parametric models require a large number of samples to fit accurately and cannot adapt instantly to the ever-changing weight in our case. Moreover, the distribution of weights is highly multi-modal, which is hard to capture for many generative models.

We instead adopt a particle-based approach and maintain a large state buffer $\mathcal{M}$ using all visited states throughout training to approximate the state space. Since the size of the buffer is limited while the state space can be infinitely large, it is important to keep representative samples that are sufficiently far from each other to ensure good coverage of the state space. When the number of states exceeds the buffer's capacity K, we use farthest point sampling (FPS) (Qi et al. 2017) which iteratively selects the farthest point from the current set of points. In our implementation, we first normalize each dimension of the states and the distance between two states is simply the Euclidean distance. More details can be found in Appendix B.1.

Overall Algorithm

Combining the subgame sampling metric and the particle-based sampler, we present a realization of the subgame curriculum learning framework, i.e., the *Subgame Automatic Curriculum Learning* (SACL) algorithm, which is summarized in Alg. 2. When each episode resets, we use the particle-based sampler to generate suitable initial states s_0 for the current policy to learn. To satisfy the requirements in Proposition 1, we also reset the game according to the initial state distribution $\rho(\cdot)$ with 0.3 probability. After collecting a number of samples, we train the policies and value functions using MARL. The weights for the newly collected states are computed according to Eq. (10) and used to update the state buffer $\mathcal{M}$. If the capacity of the state buffer is exceeded, we use FPS to select representative states-weight pairs and delete the others. An overview of SACL in the hide-and-seek game is illustrated in Fig. 3.

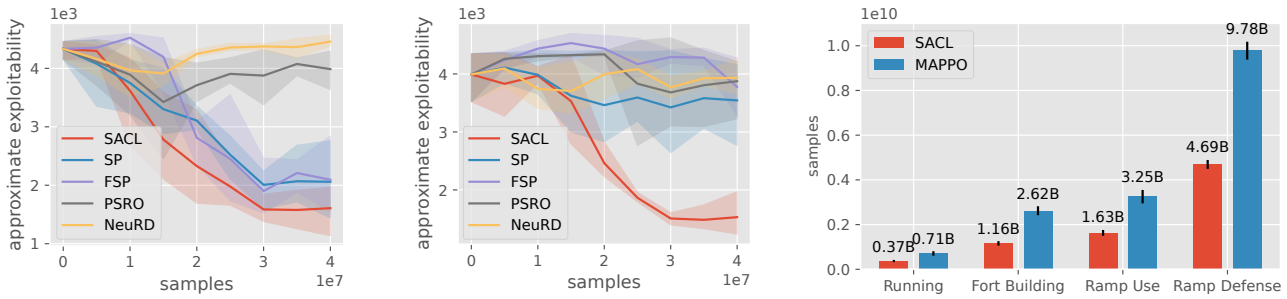
Experiment

We evaluate SACL in three different zero-sum environments: Multi-Agent Particle Environment (MPE) (Lowe et al. 2017), Google Research Football (GRF) (Kurach et al. 2020), and the hide-and-seek (HnS) environment (Baker et al. 2020). We use a state-of-the-art MARL algorithm MAPPO (Yu et al. 2021) as the backbone in all experiments. We evaluate the performance of policies by approximate exploitability. The definition and method to compute approximate exploitability can be found in Appendix B.3.

Main Results

We first compare the performance of SACL in three environments against the following baselines for solving zero-sum games: self-play (SP), two popular variants including Fictitious Self-Play (FSP) (Heinrich, Lanctot, and Silver 2015) and Neural replicator dynamics (NeuRD) (Hennes et al. 2020), and a population-based training method policy-space response oracles (PSRO) (Lanctot et al. 2017). More implementation details can be found in Appendix B.2.

Multi-Agent Particle Environment. We consider the *predator-prey* scenario in MPE, where three slower cooperating predators chase one faster prey in a square space with two obstacles. In the default setting, all agents are spawned uniformly in the square. We also consider a harder setting where the predators are spawned in the top-right corner and the prey is spawned in the bottom-left corner. All algorithms



(a) MPE: exploitability. (b) MPE hard: exploitability. (c) HnS: number of samples.

Figure 4: Main experiment results in (a) MPE, (b) MPE hard, and (c) Hide-and-seek.

Scenario	SACL	SP	FSP	PSRO	NeuRD
pass and shoot	**0.35 (0.13)**	0.48 (0.31)	0.83 (0.10)	0.80 (0.09)	0.79 (0.15)
run pass and shoot	**0.60 (0.04)**	0.68 (0.09)	0.78 (0.08)	0.83 (0.04)	0.95 (0.04)
3 vs 1 with keeper	**0.45 (0.06)**	0.83 (0.03)	0.63 (0.25)	0.85 (0.05)	0.81 (0.16)

Table 1: Approximate exploitability of learned policies in different GRF scenarios.

are trained for 40M environment samples and the curves of approximate exploitability w.r.t. sample over three seeds are shown in Fig. 4(a) and 4(b). SACL converges faster and achieves lower exploitability than all baselines in both settings, and its advantage is more obvious in the hard scenario. This is because the initial state distribution in corners makes the full game challenging to solve, while SACL generates an adaptive state distribution and learns on increasingly harder subgames to accelerate NE learning. More results and discussions can be found in Appendix C.1.

Google Research Football. We evaluate SACL in three GRF academy scenarios, namely *pass and shoot*, *run pass and shoot*, and *3 vs 1 with keeper*. In all scenarios, the left team's agents cooperate to score a goal and the right team's agents try to defend them. The first scenario is trained for 300M environment samples and the last two scenarios are trained for 400M samples. Table 1 lists the approximate exploitabilities of different methods' policies over three seeds, and SACL achieves the lowest exploitability. Additional results and discussions can be found in Appendix C.2.

Hide-and-seek environment. HnS is a challenging zero-sum game with known NE policies, which makes it possible for us to directly evaluate the number of samples used for NE convergence. We consider the *quadrant* scenario where there is a room with a door in the lower right corner. Two hiders, one box, and one ramp are spawned uniformly in the environment, and one seeker is spawned uniformly outside the room. Both the box and the ramp can be moved and locked by agents. The hiders aim to avoid the lines of sight from the seeker while the seeker aims to find the hiders.

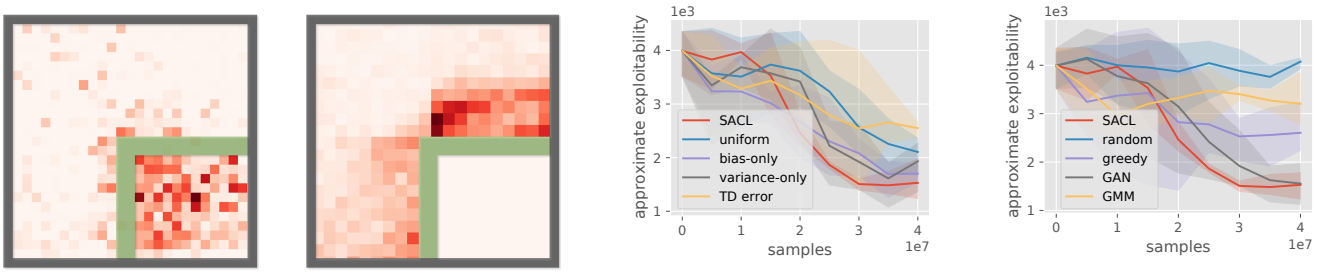
There is a total of four stages of emergent stages in HnS, i.e., Running and Chasing, Fort Building, Ramp Use, and Ramp Defense. As shown in Fig. 4(c), SACL with MAPPO backbone produces all four stages and converges to the NE policy with only 50% the samples of MAPPO with self-play. We also visualize the initial state distribution to show how SACL selects appropriate subgames for agents to learn. Fig. 5(a) depicts the distribution of hiders' position in the Fort Building stage. The probabilities of states with hiders inside the room are much higher than states with hiders outside, making it easier for hiders to learn to build a fort with the box. Similarly, the distribution of the seeker's position in the Ramp Use stage is shown in Fig. 5(b), and the most sampled subgames start from states where the seeker is close to the walls and is likely to use the ramp.

Ablation Study

We perform ablation studies to examine the effectiveness of the proposed sampling metric and particle-based sampler. All experiments are done in the hard *predator-prey* scenario of MPE and the results are averaged over three seeds. More ablation studies on state buffer size, subgame sample probability, and others can be found in Appendix C.1.

Subgame sampling metric. The sampling metric used in SACL follows Eq. (10) which consists of a bias term and a variance term. We compare it with five other metrics including a uniform metric, a bias-only metric, a variance-only metric and a temporal difference (TD) error metric. The last metric uses the TD error $|\delta_t| = |r^t + \gamma V(s^{t+1}) - V(s^t)|$ as the weight, which can be regarded as an estimation of value uncertainty. The results are shown in Fig. 5(c) and the sampling metric used by SACL outperforms both the bias-only metric and variance-only metric.

State generator. We substitute the particle-based sampler with other state generators including using GAN from the work (Dendorfer, Osep, and Leal-Taixé 2020) and using GMM from the work (Portelas et al. 2020). We also replace the FPS buffer update method with a uniform one that randomly keeps states and a greedy one that keeps states with the highest weights. Results in Fig. 5(c) show that our particle-based sampler with FPS update leads to the fastest convergence and lowest exploitability.



(a) Fort Building. (b) Ramp Use. (c) Ablation on metric. (d) Ablation on generator.

Figure 5: Visualization of the state distributions in HnS (a-b) and ablation studies (c-d).

Related Work

A large number of works achieve faster convergence in zero-sum games by playing against an increasingly stronger policy. The most popular methods are self-play and its variants (Heinrich and Silver 2016; Bai, Jin, and Yu 2020; Jin et al. 2021; Perolat et al. 2022). Self-play creates a natural curriculum and leads to emergent complex skills and behaviors (Bansal et al. 2018; Baker et al. 2020). Population-based training like double oracle (McMahan, Gordon, and Blum 2003) and policy-space response oracles (PSRO) (Lanctot et al. 2017) extend self-play by training a pool of policies. Some follow-up works further accelerate training by constructing a smart mixing strategy over the policy pool according to the policy landscape (Balduzzi et al. 2019; Perez-Nieves et al. 2021; Liu et al. 2021; Feng et al. 2021). (McAleer et al. 2021) extends PSRO to extensive-form games by building policy mixtures at all states rather than only the initial states, but it still directly solves the full game starting from some fixed states.

In addition to policy-level curriculum learning methods, other works to accelerate training in zero-sum games usually adopt heuristics and domain knowledge like the number of agents (Long et al. 2020; Wang et al. 2020b) or environment specifications (Berner et al. 2019; Serrino et al. 2019; Tang et al. 2021). By contrast, our method automatically generates a curriculum over subgames without domain knowledge and only requires the environments can be reset to desired states. Subgame-solving technique (Brown and Sandholm 2017) is also used in online strategy refinement to improve the blueprint strategy of a simplified abstract game. Another closely related work to our method is (Chen et al. 2021b) which combines backward induction with policy learning, but this method requires knowledge of the game topology and can only be applied to finite-horizon Markov games.

Besides zero-sum games, curriculum learning is also studied in cooperative settings. The problem is often formalized as goal-conditioned RL where the agents need to reach a specific goal in each episode. Curriculum learning methods design or train a smart sampler to generate proper task configurations or goals that are most suitable for training advances w.r.t. some progression metric (Chen et al. 2016; Florensa et al. 2017, 2018; Racaniere et al. 2019; Matiisen et al. 2019; Portelas et al. 2020; Dendorfer, Osep, and Leal-Taixé 2020). Such a metric typically relies on an explicit signal, such as the goal-reaching reward, success rates, or the expected value of the testing tasks. However, in the setting of zero-sum games, these explicit progression metrics become no longer valid since the value associated with a Nash equilibrium can be arbitrary. A possible implicit metric is value disagreement (Zhang, Abbeel, and Pinto 2020) used in goal-reaching tasks, which can be regarded as the variance term in our metric. By adding a bias term, our metric approximates the squared distance to NE values and gives better results in ablation studies.

Our work adopts a non-parametric subgame sampler which is fast to learn and naturally multi-modal, instead of training an expensive deep generative model like GAN (Florensa et al. 2018). Such an idea has been recently popularized in the literature. Some representative samplers are Gaussian mixture model (Warde-Farley et al. 2019), Stein variational inference (Chen et al. 2021a), Gaussian process (Mehta et al. 2020), or simply evolutionary computation (Wang et al. 2019, 2020a). Technically, our method is related to prioritized experience replay (Schaul et al. 2015; Florensa et al. 2017; Li et al. 2022) with the difference that we maintain a buffer (Warde-Farley et al. 2019) to approximate the uniform distribution over the state space. Our method is also related to episodic memory replay (Blundell et al. 2016; Pritzel et al. 2017) which stores past experience and chooses actions based on previous success when similar states are encountered. By contrast, our method proactively resets the environment to intermediate states and collects experience in the sampled subgame.

Conclusion

We present SACL, a general algorithm for accelerating MARL training in zero-sum Markov games based on the subgame curriculum learning framework. We propose to use the approximate squared distance to NE values as the sampling metric and use a particle-based sampler for subgame generation. Instead of starting from the fixed initial states, RL agents trained with SACL can practice more on subgames that are most suitable for the current policy to learn, thus boosting training efficiency. We report appealing experiment results that SACL efficiently discovers all emergent strategies in the challenging hide-and-seek environment and uses only half the samples of MAPPO with self-play. We hope SACL can be helpful to speed up prototype development and help make MARL training on complex zero-sum games more affordable to the community.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (No.62325405, U19B2019, M-0248), Tsinghua University Initiative Scientific Research Program, Tsinghua-Meituan Joint Institute for Digital Life, Beijing National Research Center for Information Science, Technology (BNRist) and Beijing Innovation Center for Future Chips.

References

Bai, Y.; Jin, C.; and Yu, T. 2020. Near-optimal reinforcement learning with self-play. *Advances in neural information processing systems*, 33: 2159–2170.

Baker, B.; Kanitscheider, I.; Markov, T.; Wu, Y.; Powell, G.; McGrew, B.; and Mordatch, I. 2020. Emergent Tool Use From Multi-Agent Autocurricula. In *International Conference on Learning Representations*.

Balduzzi, D.; Garnelo, M.; Bachrach, Y.; Czarnecki, W.; Perolat, J.; Jaderberg, M.; and Graepel, T. 2019. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning*, 434–443. PMLR.

Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; and Mordatch, I. 2018. Emergent Complexity via Multi-Agent Competition. In *International Conference on Learning Representations*.

Berner, C.; et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Blundell, C.; et al. 2016. Model-free episodic control. *arXiv preprint arXiv:1606.04460*.

Brown, N.; Lerer, A.; Gross, S.; and Sandholm, T. 2019. Deep counterfactual regret minimization. In *International conference on machine learning*, 793–802. PMLR.

Brown, N.; and Sandholm, T. 2017. Safe and nested subgame solving for imperfect-information games. *Advances in neural information processing systems*, 30.

Chen, J.; Zhang, Y.; Xu, Y.; Ma, H.; Yang, H.; Song, J.; Wang, Y.; and Wu, Y. 2021a. Variational Automatic Curriculum Learning for Sparse-Reward Cooperative Multi-Agent Problems. *Advances in Neural Information Processing Systems*, 34: 9681–9693.

Chen, W.; Zhou, Z.; Wu, Y.; and Fang, F. 2021b. Temporal Induced Self-Play for Stochastic Bayesian Games. *arXiv preprint arXiv:2108.09444*.

Chen, X.; et al. 2016. Variational Lossy Autoencoder. *arXiv preprint arXiv:1611.02731*.

Dendorfer, P.; Osep, A.; and Leal-Taixé, L. 2020. Goal-gan: Multimodal trajectory prediction based on goal position estimation. In *Proceedings of the Asian Conference on Computer Vision*.

Feng, X.; Slumbers, O.; Wan, Z.; Liu, B.; McAleer, S.; Wen, Y.; Wang, J.; and Yang, Y. 2021. Neural auto-curricula in two-player zero-sum games. *Advances in Neural Information Processing Systems*, 34: 3504–3517.

Florensa, C.; Held, D.; Wulfmeier, M.; Zhang, M.; and Abbeel, P. 2017. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*, 482–495. PMLR.

Florensa, C.; et al. 2018. Automatic goal generation for reinforcement learning agents. In *International conference on machine learning*, 1515–1528. PMLR.

Freund, Y.; and Schapire, R. E. 1996. Game theory, on-line prediction and boosting. In *Proceedings of the ninth annual conference on Computational learning theory*, 325–332.

Gruslys, A.; et al. 2020. The advantage regret-matching actor-critic. *arXiv preprint arXiv:2008.12234*.

Heinrich, J.; Lanctot, M.; and Silver, D. 2015. Fictitious self-play in extensive-form games. In *International conference on machine learning*, 805–813. PMLR.

Heinrich, J.; and Silver, D. 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*.

Hennes, D.; et al. 2020. Neural replicator dynamics: Multi-agent learning via hedging policy gradients. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 492–501.

Jaderberg, M.; et al. 2018. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281*.

Jin, C.; Liu, Q.; Wang, Y.; and Yu, T. 2021. V-Learning–A Simple, Efficient, Decentralized Algorithm for Multiagent RL. *arXiv preprint arXiv:2110.14555*.

Kurach, K.; et al. 2020. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 4501–4510.

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30.

Li, Y.; Kong, T.; Li, L.; and Wu, Y. 2022. Learning Design and Construction with Varying-Sized Materials via Prioritized Memory Resets. In *2022 International Conference on Robotics and Automation (ICRA)*, 7469–7476.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, 157–163.

Liu, X.; Jia, H.; Wen, Y.; Hu, Y.; Chen, Y.; Fan, C.; Hu, Z.; and Yang, Y. 2021. Towards Unifying Behavioral and Response Diversity for Open-ended Learning in Zero-sum Games. *Advances in Neural Information Processing Systems*, 34: 941–952.

Long, Q.; et al. 2020. Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning. In *International Conference on Learning Representations*.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.

Matiisen, T.; Oliver, A.; Cohen, T.; and Schulman, J. 2019. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*.

McAleer, S.; Lanier, J. B.; Wang, K. A.; Baldi, P.; and Fox, R. 2021. XDO: A double oracle algorithm for extensive-form games. *Advances in Neural Information Processing Systems*, 34: 23128–23139.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 536–543.

Mehta, B.; et al. 2020. Active domain randomization. In *Conference on Robot Learning*, 1162–1176. PMLR.

Perez-Nieves, N.; Yang, Y.; Slumbers, O.; Mguni, D. H.; Wen, Y.; and Wang, J. 2021. Modelling behavioural diversity for learning in open-ended games. In *International Conference on Machine Learning*, 8514–8524. PMLR.

Perolat, J.; et al. 2022. Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning. *arXiv preprint arXiv:2206.15378*.

Portelas, R.; Colas, C.; Hofmann, K.; and Oudeyer, P.-Y. 2020. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. In *Conference on Robot Learning*, 835–853. PMLR.

Pritzel, A.; et al. 2017. Neural episodic control. In *International conference on machine learning*, 2827–2836. PMLR.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in Neural Information Processing Systems*, 30.

Racaniere, S.; Lampinen, A. K.; Santoro, A.; Reichert, D. P.; Firoiu, V.; and Lillicrap, T. P. 2019. Automated curricula through setter-solver interactions. *arXiv preprint arXiv:1909.12892*.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Serrino, J.; Kleiman-Weiner, M.; Parkes, D. C.; and Tenenbaum, J. 2019. Finding friend and foe in multi-agent games. *Advances in Neural Information Processing Systems*, 32.

Silver, D.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484.

Steinberger, E.; Lerer, A.; and Brown, N. 2020. DREAM: Deep regret minimization with advantage baselines and model-free learning. *arXiv preprint arXiv:2006.10410*.

Szepesvári, C.; and Littman, M. L. 1999. A unified analysis of value-function-based reinforcement-learning algorithms. *Neural computation*, 11(8): 2017–2060.

Tang, Z.; et al. 2021. Discovering diverse multi-agent strategic behavior via reward randomization. *arXiv preprint arXiv:2103.04564*.

Tesauro, G. 1995. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3): 58–68.

Vinyals, O.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wang, R.; Lehman, J.; Clune, J.; and Stanley, K. O. 2019. Poet: open-ended coevolution of environments and their optimized solutions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 142–151.

Wang, R.; Lehman, J.; Rawal, A.; Zhi, J.; Li, Y.; Clune, J.; and Stanley, K. 2020a. Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International Conference on Machine Learning*, 9940–9951. PMLR.

Wang, W.; Yang, T.; Liu, Y.; Hao, J.; Hao, X.; Hu, Y.; Chen, Y.; Fan, C.; and Gao, Y. 2020b. From few to more: Large-scale dynamic multiagent curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 7293–7300.

Warde-Farley, D.; de Wiele, T. V.; Kulkarni, T.; Ionescu, C.; Hansen, S.; and Mnih, V. 2019. Unsupervised Control Through Non-Parametric Discriminative Rewards. In *International Conference on Learning Representations*.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8(3): 279–292.

Yu, C.; Velu, A.; Vinitsky, E.; Wang, Y.; Bayen, A.; and Wu, Y. 2021. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*.

Zhang, Y.; Abbeel, P.; and Pinto, L. 2020. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33: 7648–7659.

Zinkevich, M.; Johanson, M.; Bowling, M.; and Piccione, C. 2007. Regret minimization in games with incomplete information. *Advances in neural information processing systems*, 20.