Shortest-Path Constrained Reinforcement Learning for Sparse Reward Tasks

Sungryull Sohn [*1,2] **Sungtae Lee** [*3] **Jongwook Choi** [1] **Harm van Seijen** [4] **Mehdi Fatemi** [4] **Honglak Lee** [2,1]

Abstract

We propose the k-Shortest-Path (k-SP) constraint: a novel constraint on the agent's trajectory that improves the sample efficiency in sparse-reward MDPs. We show that any optimal policy necessarily satisfies the k-SP constraint. Notably, the k-SP constraint prevents the policy from exploring state-action pairs along the non-k-SP trajectories (*e.g.*, going back and forth). However, in practice, excluding state-action pairs may hinder the convergence of RL algorithms. To overcome this, we propose a novel cost function that penalizes the policy violating SP constraint, instead of completely excluding it. Our numerical experiment in a tabular RL setting demonstrates that the SP constraint can significantly reduce the trajectory space of policy. As a result, our constraint enables more sample efficient learning by suppressing redundant exploration and exploitation. Our experiments on *MiniGrid*, *DeepMind Lab*, *Atari*, and *Fetch* show that the proposed method significantly improves proximal policy optimization (PPO) and outperforms existing novelty-seeking exploration methods including count-based exploration even in continuous control tasks, indicating that it improves the sample efficiency by preventing the agent from taking redundant actions.

1. Introduction

Recently, deep reinforcement learning (RL) has achieved a large number of breakthroughs in many domains including video games (Mnih et al., 2015; Vinyals et al., 2019), and board games (Silver et al., 2017). Nonetheless, a central challenge in reinforcement learning (RL) is the sample efficiency (Kakade et al., 2003); it has been shown that the RL algorithm requires a large number of samples for successful learning in MDP with large state and action space. Moreover, the success of the RL algorithm heavily hinges on the quality of collected samples; the RL algorithm tends to fail if the collected trajectory does not contain enough evaluative feedback (*e.g.*, sparse or delayed reward).

To circumvent this challenge, planning-based methods utilize the environment's model to improve or create a policy instead of interacting with the environment. Recently, combining the planning method with an efficient path search algorithm, such as Monte-Carlo tree search (MCTS) (Norvig, 2002; Coulom, 2006), has demonstrated successful results (Guo et al., 2016; Vodopivec et al., 2017; Silver et al., 2017). However, such tree search methods would require an accurate model of MDP and the complexity of planning may grow intractably large for a complex domain. Model-based RL methods attempt to learn a model instead of assuming that model is given, but learning an accurate model also requires a large number of samples, which is often even harder to achieve than solving the given task. Model-free RL methods can be learned solely from the environment reward, without the need of a (learned) model. However, both value-based and policy-based methods suffer from poor sample efficiency, especially in sparse-reward tasks. To tackle the sparse reward problem, researchers have proposed to learn an intrinsic bonus function that measures the novelty of the state that agent visits (Schmidhuber, 1991; Oudeyer & Kaplan, 2009; Pathak et al., 2017; Savinov et al., 2018b; Choi et al., 2018; Burda et al., 2018b). However, when such an intrinsic bonus is added to the reward, it often requires a careful balancing between environment reward and bonus and scheduling of the bonus scale to guarantee the convergence to an optimal solution.

To tackle the aforementioned challenge of sample efficiency in sparse reward tasks, we introduce a constrained-RL framework that improves the sample efficiency of any model-free RL algorithm in sparse-reward tasks, under the mild assumptions on MDP (see Appendix J). Of note, though our framework will be formulated for policy-based methods, our final form of the cost function (Eq. (15) in Section 4) applies to both policy-based and value-based methods. We propose a novel *k-shortest-path* (k-SP) constraint (Definition 7) that improves the sample efficiency of policy learning (See Figure 1). The k-SP constraint is applied to a trajectory rolled out by a policy; all of its sub-path of length k is required to be a shortest-path under the *π-distance* metric which we

[*]Equal contribution [1]University of Michigan [2]LG AI Research [3]Yonsei University [4]Microsoft Research. Correspondence to: Sungryull Sohn <srsohn@umich.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

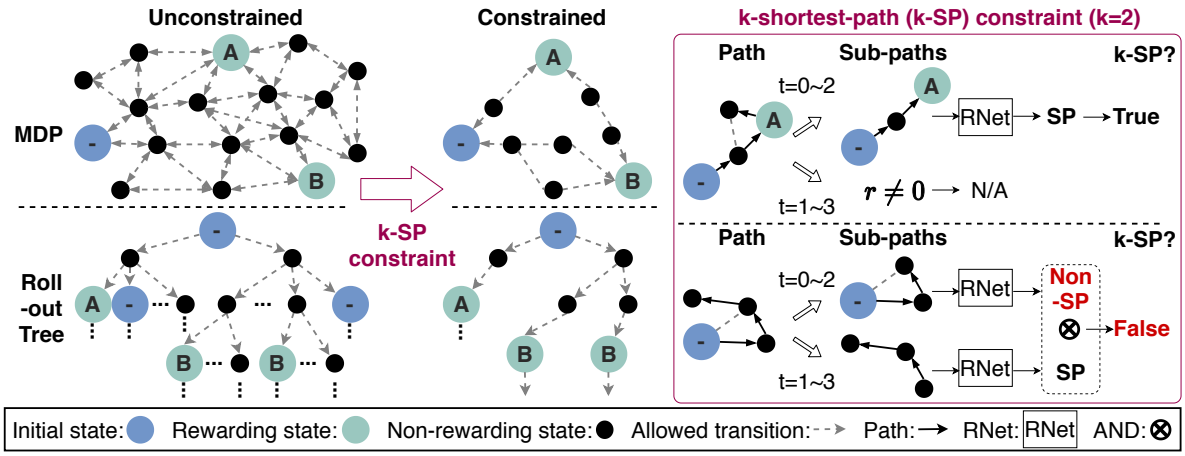


Figure 1. The k-SP constraint improves the sample efficiency of RL methods in sparse-reward tasks by pruning out suboptimal trajectories from the trajectory space. Intuitively, the k-SP constraint means that when a policy rolls out into trajectories, all of the sub-paths of length k is the shortest path (under a distance metric defined in terms of policy, discount factor, and transition probability; see Section 3.2 for the formal definition). **(Left)** MDP and a rollout tree are given. **(Middle)** The paths that satisfy the k-SP constraint. The number of admissible trajectories is drastically reduced. **(Right)** A path rolled out by a policy satisfies the k-SP constraint if all sub-paths of length k are shortest paths and have not received a non-zero reward. We use a reachability network to test if a given (sub-)path is the shortest path (See Section 4 for details).

define in Section 3.1. We prove that applying our constraint preserves the optimality for any MDP (Theorem 3), except the stochastic and multi-goal MDP which requires additional assumptions. We relax the hard constraint into a soft cost formulation (Tessler et al., 2019), and use a *reachability network* (Savinov et al., 2018b) (RNet) to efficiently learn the cost function in an off-policy manner.

We summarize our contributions as the following: **(1)** We propose a novel constraint that can improve the sample efficiency of any model-free RL method in sparse reward tasks. **(2)** We present several theoretical results including the proof that our proposed constraint preserves the optimal policy of given MDP. **(3)** We present a numerical result in tabular RL setting to precisely evaluate the effectiveness of the proposed method. **(4)** We propose a practical way to implement our proposed constraint and demonstrate that it provides a significant improvement on four complex deep RL domains. **(5)** We demonstrate that our method significantly improves the sample efficiency of PPO, and outperforms existing novelty-seeking methods on four complex domains in sparse reward settings.

2. Preliminaries

Markov Decision Process (MDP). We model a task as an MDP tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rho, \gamma)$, where $\mathcal{S}$ is a state set, $\mathcal{A}$ is an action set, $\mathcal{P}$ is a transition probability, $\mathcal{R}$ is a reward function, ρ is an initial state distribution, and $\gamma \in [0, 1)$ is a discount factor. For each state s, the value of a policy π is denoted by $V^\pi(s) = \mathbb{E}^\pi[\sum_t \gamma^t r_t \mid s_0 = s]$. Then, the goal is to find the optimal policy π^* that maximizes the expected return:

$$\pi^* = \arg\max_\pi \mathbb{E}^\pi_{s\sim\rho}\Big[\sum_t \gamma^t r_t \mid s_0 = s\Big] \quad (1)$$

$$= \arg\max_\pi \mathbb{E}_{s\sim\rho}\left[V^\pi(s)\right]. \quad (2)$$

Constrained MDP. A constrained Markov Decision Process (CMDP) is an MDP with extra constraints that restrict the domain of allowed policies (Altman, 1999). Specifically, CMDP introduces a constraint function $C(\pi)$ that maps a policy to a scalar, and a threshold $\alpha \in \mathbb{R}$. The objective of CMDP is to maximize the expected return $R(\tau) = \sum_t \gamma^t r_t$ of a trajectory $\tau = \{s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots\}$ subject to a constraint: $\pi^* = \arg\max_\pi \mathbb{E}_{\tau\sim\pi}\left[R(\tau)\right]$, s.t. $C(\pi) \leq \alpha$. A popular choice of constraint is based on the transition cost function (Tessler et al., 2019) $c(s, a, r, s') \in \mathbb{R}$ which assigns a scalar-valued cost to each transition. Then the constraint function for a policy π is defined as the discounted sum of the cost under the policy: $C(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_t \gamma^t c(s_t, a_t, r_{t+1}, s_{t+1})\right]$. In this work, we propose a *shortest-path* constraint, that provably preserves the optimal policy of the original unconstrained MDP, while reducing the trajectory space. We will use a cost function-based formulation to implement our constraint (see Section 3 and 4).

3. Formulation: k-shortest-path Constraint

We define the k-shortest-path (k-SP) constraint to remove redundant transitions (*e.g.*, unnecessarily going back and forth), leading to faster policy learning. We show two important properties of our constraint: (1) the optimal policy is preserved, and (2) the policy search space is reduced.

In this work, we limit our focus to MDPs satisfying

$R(s) + \gamma V^*(s) > 0$ for all initial states s *s.t* $\rho(s) > 0$ and all rewarding states that optimal policy visits with non-zero probability $s \in \{s | r(s) \neq 0, p_{\pi^*}(s) > 0\}$ where $p_\pi(s)$ is a probability of visiting state s with policy π. We exploit this mild assumption to prove that our constraint preserves optimality. Intuitively, we exclude the case when the optimal strategy for the agent is at best choosing a "lesser of evils" (*i.e.*, largest but negative value) which often still means a failure. We note that this is often caused by unnatural reward function design; in principle, we can avoid this by simply offsetting the reward function by a constant $-|\min_{s \in \{s | p_{\pi^*}(s)\}} V^*(s)|$ for every transition, assuming the policy is *proper*[1]. Goal-conditioned RL (Nachum et al., 2018), most of the well-known domains such as *Atari* (Bellemare et al., 2013), *DeepMind Lab* (Beattie et al., 2016), *MiniGrid* (Chevalier-Boisvert et al., 2018), etc., satisfy this assumption. Also, for general settings with stochastic MDP and multi-goals, we require additional assumptions to prove the optimality guarantee (See Appendix J for details).

3.1. Shortest-path Policy and Shortest-path Constraint

Let τ be a *path* defined by a sequence of states: $\tau = \{s_0, \dots, s_{\ell(\tau)}\}$, where $\ell(\tau)$ is the *length* of a path τ (*i.e.*, $\ell(\tau) = |\tau| - 1$). We denote the set of all paths from s to s' by $\mathcal{T}_{s,s'}$. A path τ^* from s to s' is called a *shortest path* from s to s' if $\ell(\tau^*)$ is minimum, *i.e.*, $\ell(\tau^*) = \min_{\tau \in \mathcal{T}_{s,s'}} \ell(\tau)$.

Now we will define similar concepts (length, shortest path, *etc.*) *with respect to* a policy. Intuitively, a policy that rolls out shortest paths (up to some stochasticity) to a goal state or between any state pairs should be a counterpart. We consider a set of all admissible paths from s to s' under a policy π:

Definition 1 (Path set)**.** $\mathcal{T}^\pi_{s,s'} = \{\tau \mid s_0 = s, s_{\ell(\tau)} = s', p_\pi(\tau) > 0, s_t \neq s' \text{ for } \forall t < \ell(\tau)\}$. *That is, $\mathcal{T}^\pi_{s,s'}$ is a set of all paths that policy π may roll out from s and terminate once visiting s'.*

If the MDP is a single-goal task, *i.e.*, there exists a unique (rewarding) goal state $s_g \in \mathcal{S}$ such that s_g is a terminal state, and $R(s) > 0$ if and only if $s = s_g$, any shortest path from an initial state to the goal state is the optimal path with the highest return $R(\tau)$, and a policy that always rolls out a shortest path from an initial state to the goal state is therefore optimal (see Lemma 4).[2] This is because all states except for s_g are non-rewarding states, but in general MDPs this is not necessarily true. However, this motivates us to limit the domain of the shortest path to *non-rewarding states*. We define *non-rewarding paths* from s to s' as follows:

Definition 2 (Non-rewarding path set)**.** $\mathcal{T}^\pi_{s,s',\text{nr}} = \{\tau \mid \tau \in \mathcal{T}^\pi_{s,s'}, r_t = 0 \text{ for } \forall t < \ell(\tau)\}$.

In words, $\mathcal{T}^\pi_{s,s',\text{nr}}$ is a set of all non-rewarding paths from s to s' rolled out by policy π (*i.e.*, $\tau \in \mathcal{T}^\pi_{s,s'}$) without any associated reward except the last step (*i.e.*, $r_t = 0$ for $\forall t < \ell(\tau)$). Now we are ready to define a notion of length with respect to a policy and shortest path policy:

Definition 3 (π-distance from s to s')**.** $D^\pi_{\text{nr}}(s, s') = \log_\gamma \left(\mathbb{E}_{\tau \sim \pi:\, \tau \in \mathcal{T}^\pi_{s,s',\text{nr}}} \left[\gamma^{\ell(\tau)} \right] \right)$

Definition 4 (Shortest path distance from s to s')**.** $D_{\text{nr}}(s, s') = \min_\pi D^\pi_{\text{nr}}(s, s')$.

We define π-distance to be the log-mean-exponential of the length $\ell(\tau)$ of non-rewarding paths $\tau \in \mathcal{T}^\pi_{s,s',\text{nr}}$. To be thorough, pi-distance is not a "distance" but a quasi-metric since by definition, pi-distance is asymmetric. When there exists no admissible path from s to s' under policy π, the path length is defined to be ∞: $D^\pi_{\text{nr}}(s, s') = \infty$ if $\mathcal{T}^\pi_{s,s',\text{nr}} = \emptyset$. We note that when both MDP and policy are deterministic, $D^\pi(s, s')$ recovers the natural definition of path length, $D^\pi_{\text{nr}}(s, s') = \ell(\tau)$.

We call a policy a *shortest-path policy* from s to s' if it rolls out a path with the smallest π-distance:

Definition 5 (Shortest path policy from s to s')**.** $\pi \in \Pi^{SP}_{s \to s'} = \{\pi \in \Pi \mid D^\pi_{\text{nr}}(s, s') = D_{\text{nr}}(s, s')\}$.

Finally, we will define the shortest-path (SP) constraint. Let $\mathcal{S}^{\text{IR}} = \{s \mid R(s) > 0 \text{ or } \rho(s) > 0\}$ be the union of all initial and rewarding states, and $\Phi^\pi = \{(s, s') \mid s, s' \in \mathcal{S}^{\text{IR}}, \rho(s) > 0, \mathcal{T}^\pi_{s,s',\text{nr}} \neq \emptyset\}$ be the subset of $\mathcal{S}^{\text{IR}}$ such that agent may roll out. Then, the SP constraint is applied to the non-rewarding sub-paths between states in Φ^π: $\mathcal{T}^\pi_{\Phi,\text{nr}} = \bigcup_{(s,s') \in \Phi^\pi} \mathcal{T}^\pi_{s,s',\text{nr}}$. We note that these definitions are used in the proofs (Appendix J). Now, we define the shortest-path constraint as follows:

Definition 6 (Shortest-path constraint)**.** *A policy π satisfies the shortest-path (SP) constraint if $\pi \in \Pi^{SP}$, where $\Pi^{SP} = \{\pi \mid$ For all $s, s' \in \mathcal{T}^\pi_{\Phi,\text{nr}}$, it holds $\pi \in \Pi^{SP}_{s \to s'}\}$.*

Intuitively, the SP constraint forces a policy to transition between initial and rewarding states via shortest paths. The SP constraint would be particularly effective in sparse-reward settings, where the distance between rewarding states is large.

Given these definitions, we can show that an optimal policy indeed satisfies the SP constraint in a general MDP setting. In other words, the shortest path constraint should not change optimality:

Theorem 1. *For any MDP, an optimal policy π^* satisfies the shortest-path constraint: $\pi^* \in \Pi^{SP}$.*

Proof. See Appendix J for the proof. □

[1] It is an instance of potential-based reward shaping which has optimality guarantee (Ng et al., 1999).

[2] We refer the readers to Appendix I for more detailed discussion and proofs for single-goal MDPs.

3.2. Relaxation: k-shortest-path Constraint

Implementing the shortest-path constraint is, however, intractable since it requires a distance predictor $D_{\text{nr}}(s, s')$. Note that the distance predictor addresses the optimization problem, which might be as difficult as solving the given task. To circumvent this challenge, we consider its more tractable version, namely a *k-shortest path* constraint, which reduces the shortest-path problem $D_{\text{nr}}(s, s')$ to a binary decision problem — is the state s' reachable from s within k steps? — also known as k-reachability (Savinov et al., 2018b). The k-shortest path constraint is defined as follows:

Definition 7 (k-shortest-path constraint)**.** *A policy π satisfies the k-shortest-path constraint if $\pi \in \Pi^{SP}_k$, where*

$$\Pi^{SP}_k = \{\pi \mid \textit{For all } s, s' \in \mathcal{T}^{\pi}_{\Phi,\text{nr}}, D^{\pi}_{\text{nr}}(s, s') \leq k, \textit{ it holds } \pi \in \Pi^{SP}_{s \to s'}\}. \tag{3}$$

Note that the SP constraint (Definition 6) is relaxed by adding a condition $D^{\pi}_{\text{nr}}(s, s') \leq k$. In other words, the k-SP constraint is imposed only for s, s'-path whose length is not greater than k. From Eq. (3), we can prove an important property and then Theorem 3 (optimality):

Lemma 2. *For an MDP $\mathcal{M}$, $\Pi^{SP}_m \subset \Pi^{SP}_k$ if $k < m$.*

Proof. It is true since $\{(s, s') \mid D^{\pi}_{\text{nr}}(s, s') \leq k\} \subset \{(s, s') \mid D^{\pi}_{\text{nr}}(s, s') \leq m\}$ for $k < m$. □

Theorem 3. *For an MDP $\mathcal{M}$ and any $k \in \mathbb{R}$, an optimal policy π^* is a k-shortest-path policy.*

Proof. Theorem 1 tells $\pi^* \in \Pi^{\text{SP}}$. Eq. (3) tells $\Pi^{\text{SP}} = \Pi^{\text{SP}}_{\infty}$ and Lemma 2 tells $\Pi^{\text{SP}}_{\infty} \subset \Pi^{\text{SP}}_k$. Collectively, we have $\pi^* \in \Pi^{\text{SP}} = \Pi^{\text{SP}}_{\infty} \subset \Pi^{\text{SP}}_k$. □

In conclusion, Theorem 3 states that the k-SP constraint does not change the optimality of policy, and Lemma 2 states a larger k results in a larger reduction in policy search space. Thus, it motivates us to apply the k-SP constraint in policy search to more efficiently find an optimal policy. For the numerical experiment on measuring the reduction in the policy roll-outs space, please refer to Section 6.6.

4. Shortest-Path Reinforcement Learning (SPRL)

In Section 3, we defined our k-SP constraint and proved that it preserves the optimality. In this section, we derive how the k-SP constraint on policy can be estimated by the k-SP cost that can be computed from the agent's on-policy trajectory. The proposed intrinsic cost term can be subtracted from the extrinsic reward and optimized together via any model-free RL method.

k-shortest-path Cost. The objective of RL with the k-SP constraint Π^{SP}_k can be written as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}^{\pi}\left[R(\tau)\right], \quad \text{s.t. } \pi \in \Pi^{\text{SP}}_k, \tag{4}$$

where $\Pi^{\text{SP}}_k = \{\pi \mid \forall (s, s' \in \mathcal{T}^{\pi}_{\Phi,\text{nr}}), D^{\pi}_{\text{nr}}(s, s') \leq k, \text{it holds } \pi \in \Pi^{\text{SP}}_{s \to s'}\}$ (Definition 7). We want to formulate the constraint $\pi \in \Pi^{\text{SP}}_k$ in the form of constrained MDP (Section 2), *i.e.*, as $C(\pi) \leq \alpha$. We begin by re-writing the k-SP constraint into a cost-based form:

$$\Pi^{\text{SP}}_k = \{\pi \mid C^{\text{SP}}_k(\pi) = 0\}, \text{ where} \tag{5}$$

$$C^{\text{SP}}_k(\pi) = \sum_{(s,s' \in \mathcal{T}^{\pi}_{\Phi,\text{nr}}): D^{\pi}_{\text{nr}}(s,s') \leq k} \mathbb{I}\left[D_{\text{nr}}(s, s') < D^{\pi}_{\text{nr}}(s, s')\right]. \tag{6}$$

Note that $\mathbb{I}\left[D_{\text{nr}}(s, s') < D^{\pi}_{\text{nr}}(s, s')\right] = 0 \leftrightarrow D_{\text{nr}}(s, s') = D^{\pi}_{\text{nr}}(s, s')$ since $D_{\text{nr}}(s, s') \leq D^{\pi}_{\text{nr}}(s, s')$ from Definition 4. Similar to Tessler et al. (2019), we apply the constraint to the on-policy trajectory $\tau = (s_0, s_1, \ldots)$ with discounting by replacing (s, s') with (s_t, s_{t+l}) where $[t, t+l]$ represents each segment of τ with length l:

$$C^{\text{SP}}_k(\pi) \simeq \mathbb{E}_{\tau \sim \pi}\left[C^{\text{SP}}_k(\tau)\right], \tag{7}$$

$$C^{\text{SP}}_k(\tau) = \sum_{(t,l): t \geq 0, l \leq k} \gamma^t \cdot \left(\prod_{j=t}^{t+l-1} \mathbb{I}\left[r_j = 0\right]\right) \cdot \mathbb{I}\left[D_{\text{nr}}(s_t, s_{t+l}) < D^{\pi}_{\text{nr}}(s_t, s_{t+l})\right] \tag{8}$$

$$\leq \sum_{(t,l): t \geq 0, l \leq k} \gamma^t \cdot \left(\prod_{j=t}^{t+l-1} \mathbb{I}\left[r_j = 0\right]\right) \cdot \mathbb{I}\left[D_{\text{nr}}(s_t, s_{t+l}) < k\right] \tag{9}$$

$$\triangleq \widehat{C}^{\text{SP}}_k(\pi), \tag{10}$$

where the Inequality. (9) holds because $D^{\pi}_{\text{nr}}(s_t, s_{t+l}) = \log_{\gamma}\left(\mathbb{E}_{\tau \in \mathcal{T}^{\pi}_{s_t, s_{t+l}, \text{nr}}}\left[\gamma^{|\tau|}\right]\right) < k$ from Jensen's inequality. Note that it is sufficient to consider only the cases $l = k$ (because for $l < k$, given $D_{\text{nr}}(s_t, s_{t+k}) < k$, we have $D(s_t, s_{t+l}) \leq l < k$). Then, we simplify $\widehat{C}^{\text{SP}}_k(\tau)$ as

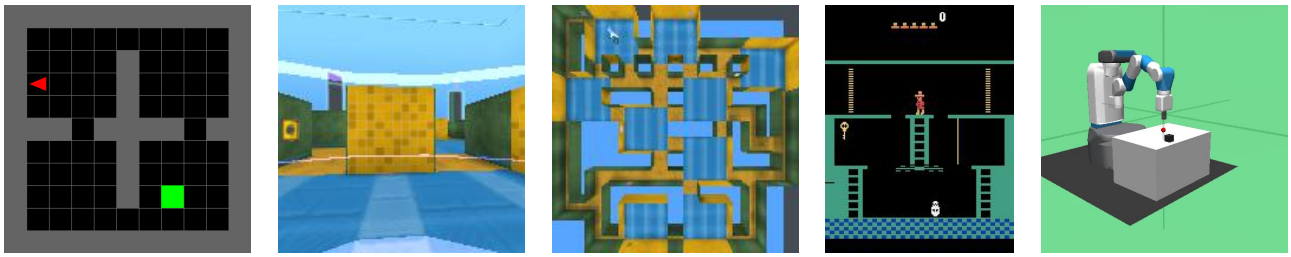
$$\widehat{C}^{\text{SP}}_k(\tau) = \sum_t \gamma^t \mathbb{I}\left[D_{\text{nr}}(s_t, s_{t+k}) < k\right] \prod_{j=t}^{t+k-1} \mathbb{I}\left[r_j = 0\right] \tag{11}$$

$$= \sum_t \gamma^t \mathbb{I}[t \geq k] \mathbb{I}\left[D_{\text{nr}}(s_{t-k}, s_t) < k\right] \prod_{j=t-k}^{t-1} \mathbb{I}\left[r_j = 0\right]. \tag{12}$$

Finally, the per-time step cost c_t is given as:

$$c_t = \mathbb{I}[t \geq k] \cdot \mathbb{I}\left[D_{\text{nr}}(s_{t-k}, s_t) < k\right] \cdot \prod_{j=t-k}^{t-1} \mathbb{I}\left[r_j = 0\right], \tag{13}$$

where $\widehat{C}^{\text{SP}}_k(\pi) = \mathbb{E}_{\tau \sim \pi}\left[\sum_t \gamma^t c_t\right]$. Note that $\widehat{C}^{\text{SP}}_k(\pi)$ is an upper bound of $C^{\text{SP}}_k(\pi)$, which will be minimized by the bound to make as little violation of the shortest-path constraint as possible. Intuitively speaking, c_t penalizes the



(a) (b) (c) (d) (e)

Figure 2. An example observation of (a) *FourRooms-11×11*, (b) *GoalLarge* in *DeepMind Lab*, (c) the maze layout (not available to the agent) of *GoalLarge*, (d) *Montezuma's Revenge* in *Atari*, and (e) *FetchPush-v1* in *Fetch*.

agent from taking a non-k-shortest path at each step, so minimizing such penalties will make the policy satisfy the k-shortest-path constraint. In Eq. (13), c_t depends on the previous k steps; hence, the resulting CMDP becomes a $(k+1)$-th order MDP. In practice, however, we empirically found that feeding only the current time-step observation to the policy performs better than stacking the previous k-steps of observations (See Appendix A.3 for details). Thus, we did not stack the observation in all the experiments. We use the Lagrange multiplier method to convert the objective (4) into an equivalent unconstrained problem as follows:

$$\min_{\lambda>0}\max_{\theta} L(\lambda,\theta) = \min_{\lambda>0}\max_{\theta} \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_t \gamma^t \left(r_t - \lambda c_t\right)\right], \tag{14}$$

where L is the Lagrangian, θ is the parameter of policy π, and $\lambda > 0$ is the Lagrangian multiplier. While in ordinary Lagrange multiplier method λ is unique since Theorem 3 shows that the shortest-path constraint preserves the optimality, we can control how much weight we will give to the constraint, i.e., we are free to set any $\lambda > 0$. Thus, we simply consider λ as a tunable *positive* hyperparameter, and simplify the min-max problem (14) to an RL objective with costs c_t being added:

$$\max_{\theta} \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_t \gamma^t \left(r_t - \lambda c_t\right)\right]. \tag{15}$$

Practical Implementation of the Cost Function. We implement the binary distance discriminator $\mathbb{I}(D_{\text{nr}}(s_{t-k}, s_t) < k)$ in Eq. (13) using the *k-reachability network* (Savinov et al., 2018b). The k-reachability network $\text{Rnet}_k(s, s')$ is trained to output 1 if the state s' is reachable from the state s with less than or equal to k consecutive actions, and 0 otherwise. Formally, we take the functional form: $\text{Rnet}_k(s, s') \simeq \mathbb{I}\left(D_{\text{nr}}(s, s') < k+1\right)$. We then estimate the cost term c_t using $(k-1)$-reachability network as follows:

$$c_t = \mathbb{I}\left[D_{\text{nr}}(s_{t-k}, s_t) < k\right]\mathbb{I}\left[t \geq k\right] \prod_{j=t-k}^{t-1} \mathbb{I}\left[r_j = 0\right] \tag{16}$$

$$= \text{Rnet}_{k-1}(s_{t-k}, s_t)\mathbb{I}\left[t \geq k\right] \prod_{j=t-k}^{t-1} \mathbb{I}\left[r_j = 0\right]. \tag{17}$$

Intuitively speaking, if the agent takes a k-shortest path, then the distance between s_{t-k} and s_t is k, hence $c_t = 0$. If it is not a k-shortest path, $c_t > 0$ since the distance between s_{t-k} and s_t will be less than k. In practice, due to the error in the reachability network, we add a small tolerance $\Delta t \in \mathbb{N}$ to ignore outliers. It leads to an empirical version of the cost as follows:

$$c_t \simeq \text{Rnet}_{k-1}(s_{t-k-\Delta t}, s_t) \cdot \prod_{j=t-k-\Delta t}^{t-1} \mathbb{I}\left[r_j = 0\right] \cdot \mathbb{I}(t \geq k + \Delta t). \tag{18}$$

In our experiment, we found that a small tolerance $\Delta t \simeq k/5$ works well in general. Similar to Savinov et al. (2018b), we used the following contrastive loss for training the reachability network:

$$\mathcal{L}_{\text{Rnet}} = -\log\left(\text{Rnet}_{k-1}(s_{\text{anc}}, s_+)\right) \tag{19}$$

$$-\log\left(1 - \text{Rnet}_{k-1}(s_{\text{anc}}, s_-)\right), \tag{20}$$

where s_{anc}, s_+, s_- are the anchor, positive, and negative samples, respectively (See Appendix E.3 for the detail of training).

5. Related Work

Shortest-path Problem and Planning. Many early works (Bellman, 1958; Ford Jr, 1956; Bertsekas & Tsitsiklis, 1991; 1995) have discussed (stochastic) shortest path problems in the context of MDP. They viewed the shortest path problem as a planning problem and proposed a dynamic programming-based algorithm similar to the value iteration (Sutton & Barto, 2018) to solve it. Our main idea is inspired by (but not based on) this viewpoint. Specifically, our method does not directly solve the shortest path problem via planning; hence, our method does not require a forward model for planning. Our method only exploits the optimality guarantee of the shortest-path under the π-distance to prune out sub-optimal policies (*i.e.*, non-shortest paths).

Distance Metric in Goal-conditioned RL. In goal-conditioned RL, there has been a recent surge of interest in learning a distance metric in state (or goal) space to construct a high-level MDP graph and perform planning to find a shortest-path to the goal state. Huang et al. (2019); Laskin et al. (2020) used the universal value func-

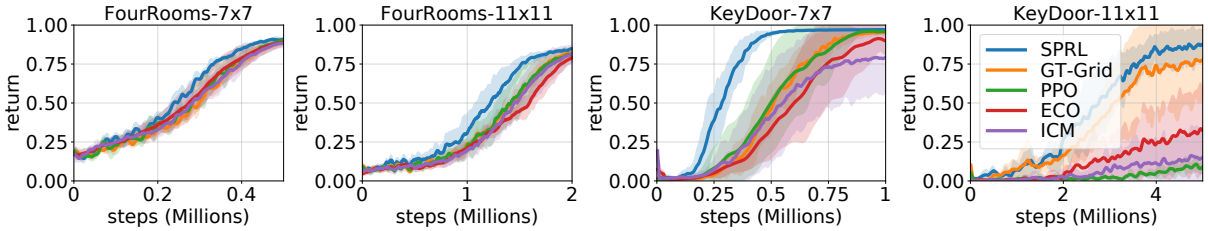


Figure 3. Progress of average episode reward on *MiniGrid* tasks. We report the mean (solid curve) and standard error (shadowed area) of the performance over six random seeds.

tion (UVF) (Schaul et al., 2015) with a constant step penalty as a distance function. Zhang et al. (2018); Laskin et al. (2020) used the success rate of transition between nodes as distance and searched for the longest path to find the plan with the highest success rate. SPTM (Savinov et al., 2018a) defined a binary distance based on the reachability network (RNet) to connect nearby nodes in the graph. However, the proposed distance metrics and methods can be used only for the goal-conditioned task and lack the theoretical guarantee in general MDP, while our theory and framework are applicable to general MDP (see Section 3.1).

Reachability Network. The reachability network (RNet) was first proposed by Savinov et al. (2018b) as a way to measure the novelty of a state for *exploration*. Intuitively, if the current state is not reachable from previous states in episodic memory, it is considered to be novel. SPTM (Savinov et al., 2018a) used RNet to predict the local connectivity (*i.e.*, binary distance) between observations in memory for *graph-based planning* in a navigation task. Zhang et al. (2020) used the k-adjacency network, which is analogous to the RNet, to improve the subgoal generation of hierarchical reinforcement learning (HRL) by constraining the goal space into adjacent states from the current state. On the other hand, we use RNet for *constraining the policy* (*i.e.*, removing the sub-optimal policies from policy space). Thus, in ours and the other three compared works, RNet is being employed for fundamentally different purposes.

Sparse Reward Problem. One of the most famous approaches to tackle the sparse reward problem in RL is intrinsic motivation (Bellemare et al., 2016; Pathak et al., 2017; Savinov et al., 2018b; Burda et al., 2018a). By adding intrinsic reward, they aim to transform the original sparse reward problem into a dense reward problem. Bellemare et al. (2016) is one of the pioneer works that formulated the intrinsic reward based on the pseudo count of state visitation to measure the state novelty. Pathak et al. (2017); Burda et al. (2018a) designed an intrinsic reward using prediction error. Savinov et al. (2018b) defined the intrinsic reward based on whether the current state is "reachable" from the previously visited states, where the reachability between a pair of states is predicted by a neural network that is trained via temporal contrastive learning. Florensa et al. (2017) created a curriculum based on the starting positions in training to tackle the sparse reward problem. The positions near the goal are considered easy and the positions far from the goal are considered hard. Riedmiller et al. (2018) formed the sparse reward problem into multiple low-level tasks. After designing an auxiliary reward function and learning a policy for every task, they learned a high-level policy that decides the sequence of low-level policies. Ecoffet et al. (2019) proposed to learn a policy that can go back to previously visited states, such that the agent can perform a directed exploration around the promising state. Our **SPRL** is not concerned with measuring the state novelty but aims to shrink the policy search space to improve the sample efficiency of RL algorithms. The exploration is promoted as a byproduct of the reduced policy search space.

More Related Works. Please refer to Appendix K for further discussions about other related works.

6. Experiments

6.1. Settings

Environments. We evaluate our **SPRL** on four challenging domains: *MiniGrid* (Chevalier-Boisvert et al., 2018), *DeepMind Lab* (Beattie et al., 2016), *Atari* (Bellemare et al., 2013), and *Fetch* (Plappert et al., 2018). *MiniGrid* is a 2D grid world environment with challenging features such as pictorial observation, random initialization of the agent and the goal, complex state and action space where coordinates, directions, and other object statuses (*e.g.*, key-door) are considered. We conducted experiments on four standard tasks: *FourRooms-7×7*, *FourRooms-11×11*, *KeyDoors-7×7*, and *KeyDoors-11×11*. *DeepMind Lab* is a 3D environment with a first-person view. Along with the nature of partially-observed MDP, at each episode, the agent's initial and the goal location are reset randomly with a change of texture, maze structure, and colors. We conducted experiments on three standard tasks: *GoalSmall*, *GoalLarge*[3] , and *ObjectMany*. For *Atari*, among 52 games we chose two sparse-reward tasks (*Montezuma's Revenge*, *Freeway*), one dense-reward task (*Ms.Pacman*), and three non-navigational tasks (*Gravitar*, *Seaquest*, *HERO*) where the agent receives reward by hitting the enemy by firing a bullet or removing the

[3] *GoalLarge* task corresponds to the *Sparse* task in Savinov et al. (2018b), and our Figure 4 reproduces the result reported.

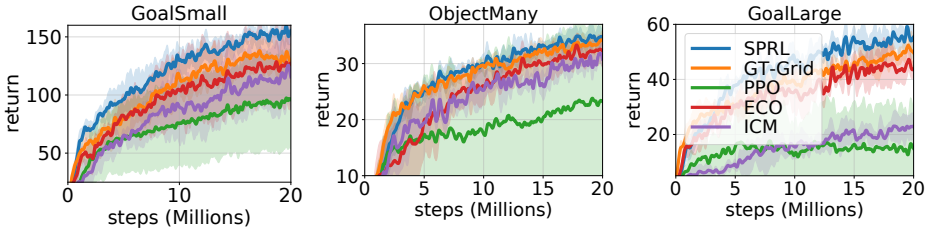


Figure 4. Progress of average episode reward on *DeepMind Lab* tasks. We report the mean (solid curve) and standard error (shadowed area) of the performance over four random seeds.

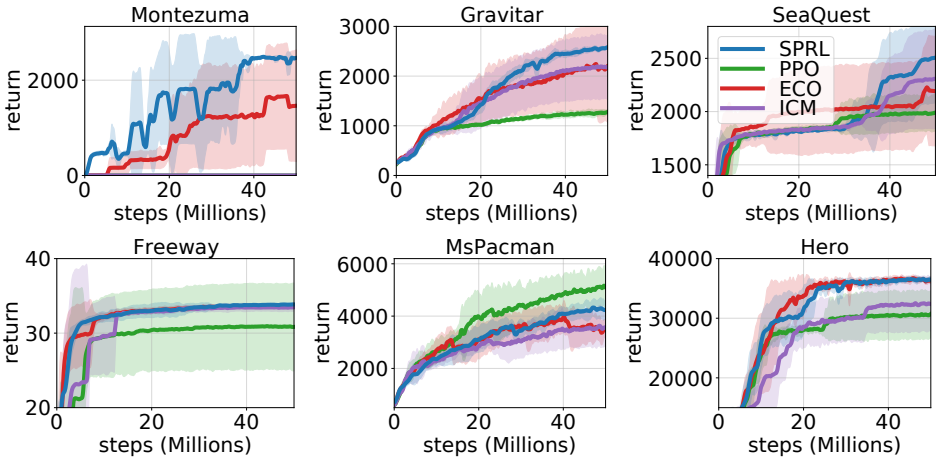


Figure 5. Progress of average episode reward on *Atari* tasks. We report the mean (solid curve) and standard error (shadowed area) of the performance over four random seeds. The performances of PPO and ICM in *Montezuma's Revenge* are both zero, hence invisible in the figure.

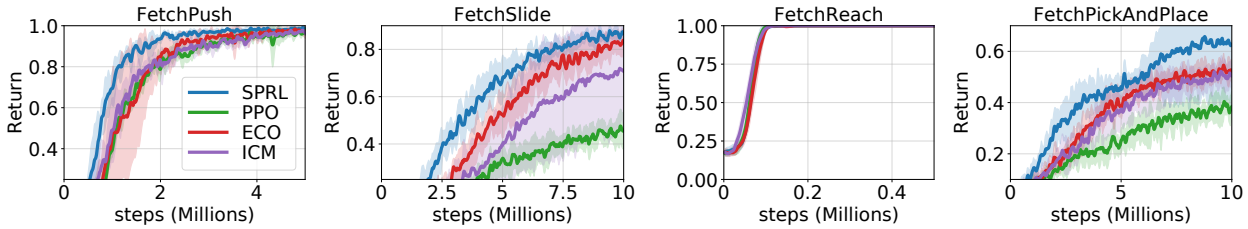


Figure 6. Progress of average episode reward on *Fetch* tasks. We report the mean (solid curve) and standard error (shadowed area) of the performance over four random seeds.

obstacle by installing a bomb. *Fetch* is a continuous control environment with a two-fingered gripper. Initial locations of the agent and the goal change every episode. We made two changes in the environment to make it a sparse-reward task. The agent receives +1 reward if the agent reaches the goal and 0 rewards otherwise. Also, the episode terminates when the agent reaches the goal such that the agent can receive a non-zero reward at most once in an episode. We conducted experiments on all four tasks: *FetchPush-v1*, *FetchSlide-v1*, *FetchReach-v1*, *FetchPickAndPlace-v1*. We refer the readers to Figure 2 for examples of observations. See Appendix D, Appendix E, Appendix F, and Appendix G for more details of *MiniGrid*, *DeepMind Lab*, *Atari*, and *Fetch* respectively.

Baselines. We compared our methods with four baselines: **PPO** (Schulman et al., 2017), *episodic curiosity* (**ECO**) (Savinov et al., 2018b), *intrinsic curiosity module* (**ICM**) (Pathak et al., 2017), and **GT-Grid** (Savinov et al., 2018b). The **PPO** is used as a baseline RL algorithm for all other agents. The **ECO** agent is rewarded when it visits a state that is not reachable from the states in episodic memory within a certain number of actions; thus the novelty is only measured within an episode. Following Savinov et al. (2018b), we trained RNet in an off-policy manner from the agent's experience and used it for our **SPRL** and **ECO** on *MiniGrid* (Section 6.2), *DeepMind Lab* (Section 6.3), *Atari* (Section 6.4) and *Fetch* (Section 6.5). For the accuracy of the learned RNet on each task, please refer to Appendix B. The **GT-Grid** agent has access to the agent's (x, y) coordinates. It uniformly divides the world into 2D grid cells, and the agent is rewarded for visiting a novel grid cell. The **ICM** agent learns a forward and inverse dynamics model and uses the prediction error of the forward model to measure the

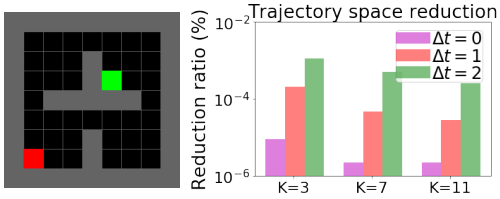


Figure 7. (Left) 7×7 Tabular four-rooms domain with initial agent location (red) and the goal location (green). (Right) The trajectory space reduction ratio (%) before and after constraining the trajectory space for various k and Δt with k-SP constraint. Even a small k can greatly reduce the trajectory space with a reasonable tolerance Δt.

novelty. We used the publicly available codebase (Savinov et al., 2018b) to obtain the baseline results. We used the same hyperparameter for all the tasks for a given domain — the details are described in the Appendix. We used the standard domain and tasks for reproducibility.

6.2. Results on *MiniGrid*

Figure 3 shows the performance of all the methods on the *MiniGrid* domain. **SPRL** consistently outperforms all the baseline methods over all tasks. We observe that exploration-based methods (*i.e.*, **ECO**, **ICM**, and **GT-Grid**) perform similarly to the **PPO** in the tasks with small state space (*e.g.*, *FourRooms-7×7* and *KeyDoors-7×7*). However, **SPRL** demonstrates a significant performance gain since it improves the exploitation by avoiding sub-optimality caused by taking a non-shortest-path.

6.3. Results on *DeepMind Lab*

Figure 4 shows the performance of all the methods on *DeepMind Lab* tasks. Overall, **SPRL** achieves superior results compared to other methods. By the task design, the difficulty of exploration increases in the order of *GoalSmall*, *ObjectMany*, and *GoalLarge* tasks, and we observe a coherent trend in the result. For harder exploration tasks, the exploration-based methods (**GT-Grid**, **ICM** and **ECO**) achieve a larger improvement over **PPO**: *e.g.*, 20%, 50%, and 100% improvement in *GoalSmall*, *ObjectMany*, and *GoalLarge*, respectively. As shown in Lemma 2, **SPRL** is expected to have larger improvement for larger trajectory space and sparser reward settings. We can verify this from the result: **SPRL** has the largest improvement in *GoalLarge* task, where both the map is largest and the reward is most sparse. Interestingly, **SPRL** even outperforms **GT-Grid** which simulates the upper-bound performance of novelty-seeking exploration method. This is possible since **SPRL** improves the exploration by suppressing unnecessary explorations, which is different from novelty-seeking methods and also improves the exploitation by reducing the policy search space.

6.4. Results on *Atari*

One of the main challenges in *Atari* is the distribution shift in the state space within a task. Unlike *MiniGrid* and *DeepMind Lab*, many *Atari* tasks involve the transition between different rooms in each of which the agent observes a significantly different set of states. This induces instability in the RNet training; RNet often overfits to the initial room and performs poorly when the agent navigates to the different rooms. To mitigate this problem, we added the weight decay for the RNet training. For other technical details, please refer to Appendix F.3.

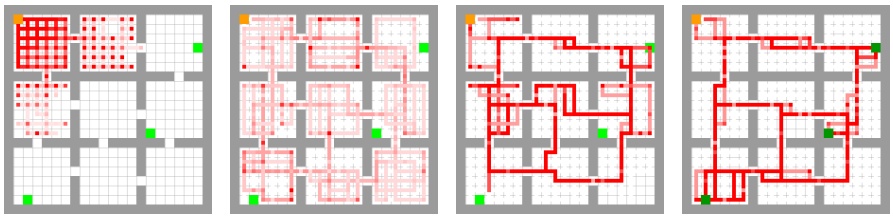
Figure 5 summarizes the performance of all the methods on *Atari* tasks. **SPRL** outperforms all the baseline methods on five out of six tasks except for *Ms.Pacman*, which is a dense reward task. We note that other exploration methods, **ICM** and **ECO**, also perform poorly on this task. For the sparse reward task, especially in *Montezuma's Revenge*, **SPRL** achieves the performance comparable to the SOTA exploration methods such as RND (Burda et al., 2018b) (1000 score at 50M steps with 32 parallel environments. **SPRL** used 12 parallel environments.) and SOTA exploitation methods such as SIL (Oh et al., 2018) (2500 score at 50M steps). Lastly, **SPRL** achieves the largest improvement to the **PPO** in non-navigational tasks (*Gravitar*, *Seaquest*, *HERO*). This verifies that our k-SP constraint is not limited to just the geometric path but can be applied to any general trajectory, or a sequence of state transitions, in MDP.

6.5. Results on *Fetch*

Figure 6 summarizes the performance of all the methods on *Fetch* tasks. **SPRL** outperforms all the baseline methods on every task except for *FetchReach-v1*, in which all the compared methods perform similarly well. We also found that the performance improvement of the exploration methods such as **SPRL**, **ECO**, and **ICM** over **PPO** is larger for the tasks with sparser reward; the reward is the densest in *FetchReach-v1* while the most sparse in *FetchSlide-v1*. As suggested by our theory, this result shows that **SPRL** is not restricted to the domains with discrete action space but performs well on the continuous control domain. We present the additional results in Appendix B showing that the reachability network, which is the key component of **SPRL**, can be efficiently trained and accurately compute the state proximity in the continuous control domain.

6.6. Quantitative Analysis on k-SP Constraint

In this section, we numerically evaluate the effect of our k-shortest path constraint in a tabular-RL setting. Specifically, we study the following questions: (1) Does the k-SP constraint with larger k results in more reduction in trajectory space? (*i.e.*, validation of Lemma 2) (2) How much reduction in trajectory space does k-SP constraint provide



(a) **Random** (b) **GT-UCB** (c) **SPRL** (d) **SPRL**+Reward

Figure 8. Transition count maps for baselines and **SPRL**: (a), (b), and (c) are in a *reward-free* while (d) is in a *reward-aware* setting. In reward-free settings (a-c), we show rewarding states in light green only for the visualization, but the agent does not receive rewards from the environment. The location of the agent's initial state (orange) and rewarding states (dark green) are fixed. The episode length is limited to 500 steps.

with different k and tolerance Δt?

We implemented a simple tabular 7x7 four-rooms domain where each state maps to a unique (x, y) location of the agent. The agent can take *up, down, left, right* primitive actions to move to the neighboring state, and the episode horizon is set to 14 steps. The goal of the agent is reaching the goal state, which gives +1 reward and terminates the episode. We computed the ground-truth distance between a pair of states to implement the k-shortest path constraint. We used the ground-truth distance function instead of the learned RNet to implement the exact SPRL agent.

Figure 7 summarizes the reduction in the trajectory space size. We searched over all possible trajectories of length 14 using breadth-first-search (BFS). Then we counted the number of trajectories satisfying our k-SP constraint with varying parameters k and tolerance Δt and divided by the total number of trajectories (*i.e.*, $4^{14} = 268$M). The result shows that our k-SP constraint drastically reduces the trajectory space even in a simple 2D grid domain; with a very small $k = 3$ and no tolerance $\Delta t = 2$, we get only $24/268$M size of the original search space. As we increase k, we can see more reduction in the trajectory space, which is consistent with Lemma 2. Also, increasing the tolerance Δt slightly hurts the performance, but still achieves a large reduction (See Appendix A for more analysis on the effect of k and tolerance).

6.7. Qualitative Analysis on *MiniGrid*

We qualitatively studied what type of policy is learned with the k-SP constraint with the ground-truth RNet in *NineRooms* domain of *MiniGrid*. Figure 8 (a-c) shows the converged behavior of **SPRL** ($k = 15$), the ground-truth count-based exploration (Lai & Robbins, 1985) agent (**GT-UCB**), and uniformly random policy (**Random**) in a reward-free setting. We counted all the state transitions ($s_t \rightarrow s_{t+1}$) of each agent's roll-out and averaged over 4 random seeds. **Random** cannot explore further than the initial few rooms. **GT-UCB** seeks novel states and visits all the states uniformly. **SPRL** learns to take the longest possible shortest path, which results in a "straight" path across the rooms. Note that this only represents a partial behavior of **SPRL**, since our cost also considers the *existence of non-zero reward* (see Eq. (13)). Thus, in (d), we tested **SPRL** while providing only the *existence* of non-zero reward (but not the reward magnitude). **SPRL** learns to take the shortest path between rewarding and initial states that is consistent with the shortest-path definition in Definition 7.

7. Conclusion

We presented the k-shortest-path constraint, which can improve the sample efficiency of any model-free RL method by preventing the agent from taking sub-optimal transitions. We empirically showed that **SPRL** outperforms vanilla RL and strong novelty-seeking exploration baselines on four challenging domains. We believe that our framework develops a unique direction for improving the sample efficiency in reinforcement learning; hence, combining our work with other techniques for better sample efficiency will be an interesting future work that could benefit many practical tasks.

Acknowledgements This research is supported in part by NSF IIS #1453651 and Korea Foundation for Advanced Studies.

References

Abel, D., Hershkowitz, D., and Littman, M. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*, pp. 2915–2923, 2016.

Abel, D., Arumugam, D., Lehnert, L., and Littman, M. State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*, pp. 10–19, 2018.

Altman, E. *Constrained Markov decision processes*, volume 7. CRC Press, 1999.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. *arXiv preprint arXiv:1606.01868*, 2016.

Bellman, R. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

Bertsekas, D. P. and Tsitsiklis, J. N. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.

Bertsekas, D. P. and Tsitsiklis, J. N. Neuro-dynamic programming: an overview. In *Proceedings of 1995 34th IEEE Conference on Decision and Control*, volume 1, pp. 560–564. IEEE, 1995.

Bertsekas, D. P., Castanon, D. A., et al. Adaptive aggregation methods for infinite horizon dynamic programming. 1988.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018a.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018b.

Castro, P. S. Scalable methods for computing state similarity in deterministic markov decision processes. *arXiv preprint arXiv:1911.09291*, 2019.

Chevalier-Boisvert, M., Willems, L., and Pal, S. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

Choi, J., Guo, Y., Moczulski, M., Oh, J., Wu, N., Norouzi, M., and Lee, H. Contingency-aware exploration in reinforcement learning. *arXiv preprint arXiv:1811.01483*, 2018.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

Dean, T. L., Givan, R., and Leach, S. Model reduction techniques for computing approximately optimal solutions for markov decision processes. *arXiv preprint arXiv:1302.1533*, 2013.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. https://github.com/openai/baselines, 2017.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

Ferns, N., Panangaden, P., and Precup, D. Metrics for finite markov decision processes. In *UAI*, volume 4, pp. 162–169, 2004.

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

Ford Jr, L. R. Network flow theory. Technical report, Rand Corp Santa Monica Ca, 1956.

Givan, R., Dean, T., and Greig, M. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

Guo, X., Singh, S., Lewis, R., and Lee, H. Deep learning for reward design to improve monte carlo tree search in atari games. *arXiv preprint arXiv:1604.07095*, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1480–1490. JMLR. org, 2017.

Huang, Z., Liu, F., and Su, H. Mapping state space using landmarks for universal goal reaching. In *Advances in Neural Information Processing Systems*, pp. 1940–1950, 2019.

Kakade, S. M. et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

Khetarpal, K. and Precup, D. Attend before you act: Leveraging human visual attention for continual learning. *arXiv preprint arXiv:1807.09664*, 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lai, T. L. and Robbins, H. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1): 4–22, 1985.

Laskin, M., Emmons, S., Jain, A., Kurutach, T., Abbeel, P., and Pathak, D. Sparse graphical memory for robust planning. *arXiv preprint arXiv:2003.06417*, 2020.

Li, L., Walsh, T. J., and Littman, M. L. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3303–3313, 2018.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Norvig, P. R. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

Oh, J., Guo, Y., Singh, S., and Lee, H. Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887. PMLR, 2018.

Oudeyer, P.-Y. and Kaplan, F. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2778–2787. JMLR. org, 2017.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Riedmiller, M., Hafner, R., Lampe, T., Neunert, M., Degrave, J., Wiele, T., Mnih, V., Heess, N., and Springenberg, J. T. Learning by playing solving sparse reward tasks from scratch. In *International Conference on Machine Learning*, pp. 4344–4353. PMLR, 2018.

Savinov, N., Dosovitskiy, A., and Koltun, V. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018a.

Savinov, N., Raichuk, A., Marinier, R., Vincent, D., Pollefeys, M., Lillicrap, T., and Gelly, S. Episodic curiosity through reachability. *ICLR*, 2018b.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International conference on machine learning*, pp. 1312–1320, 2015.

Schmidhuber, J. Adaptive confidence and adaptive curiosity. In *Institut fur Informatik, Technische Universitat Munchen, Arcisstr. 21, 800 Munchen 2*. Citeseer, 1991.

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

Shang, W., Sohn, K., Almeida, D., and Lee, H. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *international conference on machine learning*, pp. 2217–2225, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Sutton, R. S. Between mdps and semi-mdps: Learning, planning, and representing knowledge at multiple temporal scales. 1998.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tessler, C., Mankowitz, D. J., and Mannor, S. Reward constrained policy optimization. *ICLR*, 2019.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3540–3549. JMLR. org, 2017.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Vodopivec, T., Samothrakis, S., and Ster, B. On monte carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, 60:881–936, 2017.

Zhang, A., Lerer, A., Sukhbaatar, S., Fergus, R., and Szlam, A. Composable planning with attributes. *ICML*, 2018.

Zhang, T., Guo, S., Tan, T., Hu, X., and Chen, F. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *arXiv preprint arXiv:2006.11485*, 2020.