

Search Inspired Exploration for Reinforcement Learning

by

Georgios Sotirchos

In partial fulfillment of the requirements for the degree of

Master of Science

at Delft University of Technology,
to be defended publicly on 14 Nov 2025.

Faculty: Mechanical Engineering
Department: Cognitive Robotics (CoR)
Programme: MSc Robotics

Mentors/Supervisors: Zlatan Ajanović (RWTH Aachen University)
Jens Kober (TUD)

Graduation committee: Zlatan Ajanović (RWTH Aachen University)
Jens Kober (TUD)
Robert Babuška (TUD)

This thesis is confidential and cannot be made public until 31 Jan 2026.

An electronic version of this thesis is available at <http://repository.tudelft.nl>.



Keywords: *search, exploration, reinforcement learning*

ABSTRACT

Exploration in environments with sparse rewards remains a fundamental challenge for reinforcement learning (RL). Existing approaches such as curriculum learning and Go-Explore often rely on hand-crafted heuristics, while curiosity-driven methods risk converging to suboptimal policies. We propose Search-Inspired Exploration in Reinforcement Learning (SIERL), a novel method that actively guides exploration by setting sub-goals based on the agent’s learning progress. At the beginning of each episode, SIERL chooses a sub-goal from the *frontier* (the boundary of the agent’s known state space) before the agent continues exploring toward the main task objective. The key contribution of our method is the sub-goal selection mechanism, which provides state-action pairs that are neither overly familiar nor completely novel. It assures that the frontier is expanded systematically and that the agent is capable of reaching any state within it. Inspired by search, sub-goals are prioritized from the frontier based on estimates of cost-to-come and cost-to-go, effectively steering exploration towards the most informative regions. In experiments on challenging sparse-reward environments, SIERL outperforms dominant baselines in both achieving the main task goal and generalizing to reach arbitrary states in the environment.

1 INTRODUCTION

Reinforcement learning (RL) holds the promise of enabling agents to master complex tasks by interacting with their environments. Yet applying RL in realistic domains remains challenging due to the combination of high-dimensional state–action spaces and sparse reward signals. In many environments, meaningful feedback is obtained only after completing long sequences of actions, making standard RL algorithms highly data-inefficient.

A central obstacle is the exploration–exploitation dilemma: agents must discover novel behaviors while simultaneously leveraging what they already know to make progress. Existing methods often overlook the problem of how an agent can *actively* direct its exploration to collect the most informative experiences (Amin et al.). Addressing this challenge is crucial for developing RL agents that learn more stably and scale to environments with delayed or infrequent rewards. We argue that progress requires shifting from agents that passively process environment feedback to those that deliberately seek out information in a principled way.

Several approaches have been proposed to address the challenge of exploration in sparse-reward reinforcement learning. Curriculum Learning (CL) introduces tasks of increasing difficulty to gradually shape agent behavior, but it relies on carefully hand-crafted difficulty metrics and is prone to negative transfer if the curriculum is poorly designed (Fang et al., Liu et al.). Intrinsic motivation methods reward novelty or curiosity, encouraging the agent to seek unexplored regions of the state space. However, these methods are often a form of reward shaping, which can bias the learning process and lead to suboptimal policies. They are also susceptible to the “noisy-TV problem,” where agents are distracted by stochastic but irrelevant features (Burda et al., Ladosz et al.). Go-Explore (Ecoffet et al.) explicitly remembers and returns to promising states, but depends heavily on domain-specific heuristics and requires careful selection of interesting states.

Goal-Conditioned Reinforcement Learning (GCRL), particularly when combined with Hindsight Experience Replay (HER), offers another principled framework for overcoming these limitations by explicitly training agents to reach arbitrary states. We build on this paradigm to automatically generate sub-goals that extend progressively farther from the start state. In doing so, our method effectively constructs a curriculum without requiring manually designed tasks or environments of varying difficulty.

In this work, we introduce *Search-Inspired Exploration* (SIERL), a novel approach that guides exploration by setting sub-goals informed by the agent’s learning progress. Our main contributions are: 1. We propose a principled sub-goal selection mechanism that systematically expands exploration by defining a frontier of experience and prioritizing sub-goals using cost-to-come and cost-to-go

estimates. 2. We design a novel *Hallway* environment that enables fine-grained control over exploration difficulty by varying the length of action sequences required to succeed. 3. We show that SIERL leads to more efficient exploration in sparse-reward settings. 4. We present an empirical study that disentangles the contributions of individual components and identifies which mechanisms most effectively improve exploration for goal-conditioned agents.

The remainder of this paper is organized as follows: Section 2 reviews related literature; Section 3 introduces the necessary preliminaries; Section 4 details our algorithm; Section 5 presents experimental results; and Sections 6 and 7 discuss conclusions and future directions.

2 RELATED WORK

A wide range of exploration methods have been proposed for reinforcement learning (RL). These methods can be broadly categorized along several axes: whether they rely on extrinsic rewards or intrinsic exploration bonuses, employ memory or are memory-free, learn autonomously or from demonstrations, act randomly or deliberately (goal-based), or adopt an optimism-driven strategy (Amin et al., Ladosz et al.).

Novelty and optimism-based methods. Novelty-bonus and optimism-based approaches encourage exploration by augmenting rewards with bonuses for visiting new or uncertain states. These methods are particularly useful in sparse-reward environments, where intrinsic signals provide more consistent feedback than delayed extrinsic rewards. In the bandit setting, the well-known Upper Confidence Bounds (UCB) algorithm balances exploration and exploitation by favoring actions with high value uncertainty (Auer et al.). In reinforcement learning, count-based techniques extend this principle by quantifying novelty through visitation counts over states or state-action pairs (Strehl & Littman). Practical implementations rely on approximations such as hashing (Tang et al.), pseudo-counts (Ostrovski et al.), or elliptical episodic bonuses (Henaff et al.), all of which assign higher exploration bonuses to rarely visited regions of the state space. Pseudo-count methods in particular have demonstrated strong performance on hard-exploration benchmarks, notably achieving state-of-the-art results on Montezuma’s Revenge (Bellemare et al.). To avoid the limitations of explicit counting, Random Network Distillation (RND) (Burda et al.) introduces a scalable alternative: a predictor network is trained to match the outputs of a fixed, randomly initialized target network, and the prediction error serves as an intrinsic reward. Novel states typically yield higher prediction errors, thus guiding exploration toward regions where the agent’s predictive model is least accurate. Broader novelty-driven methods extend beyond counts: optimistic initialization assumes unseen state-action pairs yield high returns, biasing agents toward exploration under the “Optimism in the Face of Uncertainty” principle (Treven et al.).

Goal-based exploration methods. Goal-based methods frame exploration as a deliberate process rather than relying on random or purely novelty-driven signals. By defining or generating explicit goals within the environment, these methods encourage the agent to learn policies that reach strategically important or unexplored states. This structured exploration typically involves three components: a mechanism for goal generation (e.g., sub-goals), a policy for goal discovery, and an overall strategy that coordinates exploration around these targets. Notable examples include Go-Explore (Ecoffet et al.), which achieves strong performance by explicitly remembering and returning to promising states before exploring further. Other approaches incorporate planning techniques, either within model-based RL frameworks (Hayamizu et al.) or by substituting policy search components with kinodynamic planners to better direct exploration (Hollenstein et al.). Planning Exploratory Goals (PEG) (Hu et al.) leverages learned world models to sample exploratory “goal commands” predicted to unlock novel states, from which the agent then explores.

Frontier- and confidence-driven exploration. Several recent methods refine goal-based exploration by explicitly reasoning about the frontier of reachable states or by incorporating measures of confidence. Latent Exploration Along the Frontier (LEAF) (Bharadhwaj et al.) learns a dynamics-aware latent manifold of states, deterministically navigates to its frontier, and then stochastically explores beyond it to reach new goals. Temporal Distance-aware Representations (TLDR) (Bae et al.) exploit temporal distance as a proxy for exploration potential, selecting faraway goals to encourage coverage and training policies to minimize or maximize temporal distance as needed. In reset-free settings, Reset-Free RL with Intelligently Switching Controller (RISC) dynamically al-

ternates between forward and backward exploration goals based on confidence in achieving them, effectively balancing task-oriented progress with revisiting initial states to diversify experience.

Summary. Together, these exploration strategies illustrate a steady evolution in RL: from simple count-based and novelty-driven approaches to increasingly structured methods that incorporate goal-setting, planning, and confidence-aware strategies. Novelty-based techniques provide intrinsic motivation to reduce uncertainty and expand coverage, but as a form of reward shaping, they can bias behavior and lead to suboptimal policies, in addition to being vulnerable to distractions such as the “noisy-TV problem.” Goal-based approaches make exploration more intentional by defining explicit targets such as distant states, frontier boundaries, or strategically planned points. However, they often rely on brittle heuristics, handcrafted difficulty metrics, or domain-specific knowledge that limits generality. These drawbacks highlight an open gap: how to design exploration methods that are both systematic and robust, capable of scaling beyond hand-tuned heuristics while ensuring that chosen exploratory targets remain novel but still reachable.

3 BACKGROUND AND PROBLEM SETUP

Exploration in reinforcement learning is especially challenging in environments with sparse rewards, where agents must solve long sequences of actions before receiving feedback. To formalize this setting, we focus on sequential decision-making problems with explicit goals, expressed through Goal Markov Decision Processes (GMDPs). This framework highlights the difficulty of discovering goals when reward signals are rare and emphasizes the role of the exploration–exploitation dilemma in guiding agent behavior.

3.1 THE HARD-EXPLORATION PROBLEM

Hard exploration problems are a direct consequence of sparse rewards, often exacerbated by large state and/or action spaces. When rewards are sparse, learning can be extremely slow because the agent wanders aimlessly for long periods without any signal to guide its behavior (Ladosz et al.). If the path to a reward is long and specific, random exploration strategies (like ε -greedy, with a small ε) are unlikely to find it in a reasonable amount of time. The agent might get stuck in local optima of familiar, non-rewarding behavior, or it might never encounter the critical states that lead to high rewards. Therefore, in these problems, we need more sophisticated exploration strategies that can intelligently seek out beneficial experiences for the agent to learn from.

3.2 GOAL MARKOV DECISION PROCESSES

We considered sequential decision-making problems that are formalized as Markov Decision Processes (MDPs). An MDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, P_a, r, \gamma \rangle$, where \mathcal{S} is a finite set of possible states, \mathcal{A} is a finite set of actions available to the agent, $P_a(s' | s, a)$ is a function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ that returns the transition probability defining the likelihood of transitioning to state s' after taking action a in state s ; $r(s, a, s')$ is the reward function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, \infty)$ specifying the immediate reward received after a transition, and $\gamma \in [0, 1]$ is a discount factor that balances the importance of immediate versus future rewards. The primary objective of an agent in an MDP is to learn a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected cumulative discounted reward ($\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_{t+1}]$), often referred to as the value function.

A significant group of problems, particularly relevant in planning and many reinforcement learning applications, are goal-oriented tasks. Such tasks can be formalized using Goal-MDPs or *Shortest-path MDPs* (Bertsekas). In this formulation, there can be one or more designated goal states in the environment, and the agent’s primary task at each point in time is to reach the current goal. $s_G \subseteq \mathcal{S}$ is the set of all possible (absorbing) goal states. The reward structure in Goal MDPs is often adjusted to reflect this objective; a common setup involves a positive or zero reward upon reaching a goal state and zero and small negative rewards (costs) for all other transitions. Thus, instead of the reward function $r(s, a, s')$, a cost function $c(a, s)$, a map $\mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, is used that specifies a cost for each action. Goal states $s_G \in \mathcal{S}$ can be absorbing, meaning $P_a(s_G | a, s_G) = 1$ for all $a \in \mathcal{A}$, and cost free, meaning $c(a, s_G) = 0$ for all $a \in \mathcal{A}$. This transforms the problem into one of finding an optimal path or policy for each goal, with the objective of achieving a desired terminal goal or

Algorithm 1: SIERL Algorithm (abridged - details in Appendix A)

```
Input: agent,  $s_G$ 
frontier  $\leftarrow \{\}$  // Initialize frontier
while training do
     $s_{SG} = \text{get\_subgoal}(frontier)$  // Get sub-goal
    while not timeout do
         $s' \leftarrow \text{Execute}(\pi(s, s_G))$ 
        frontier.update( $s, a$ ) // Insert (or not) in frontier
        if should_switch( $s', s_G$ ) then
             $s_{SG} \leftarrow s_G$  // Sub-goal reached or early switching
```

condition. It has been shown that (partially observable) MDPs can be transformed into equivalent (partially observable) Goal MDPs (Bertsekas)

3.3 EXPLORATION VS EXPLOITATION DILEMMA

In Goal-MDPs with sparse rewards, the exploration-exploitation dilemma is particularly acute. Exploitation leverages past knowledge but offers little benefit early on, when goal rewards remain undiscovered. Exploration requires trying new actions and states without immediate payoff, often at high cost, but is essential for locating rare reward signals. The central challenge is to balance extensive exploration with eventual convergence on an optimal policy: without sufficient exploration, goals may never be found, but without exploitation, progress toward them cannot be consolidated.

4 METHOD

Our method, SIERL, introduces a principled way to perform deliberate exploration in reinforcement learning through goal-conditioned sub-goal setting. The key premise is that state-action regions become progressively less informative as they are explored more extensively: once the agent has learned accurate value estimates locally, further exploration in the same region yields diminishing returns. Instead, the agent should expand exploration toward novel but reachable states at the edge of its current knowledge, thereby extending the frontier of explored regions.

To achieve this, we employ a two-phase exploration process. In the first phase, the agent follows a goal-conditioned policy to reach selected frontier sub-goals, systematically expanding the boundary of explored states. In the second phase, the agent uses the experience gained in Phase 1 to explore efficiently toward the main task goal. This strategy combines systematic expansion with goal-directed exploration, ensuring both stable learning of an optimal policy for the task goal and improved generalization to alternative goals.

A short pseudo-code description of SIERL is provided in Algorithm 1, with full implementation details in Appendix A.

4.1 TWO-PHASE EXPLORATION STRATEGY

Formally, we assume a goal-conditioned policy $\pi(a | s, g)$ that selects actions conditioned on the current state $s \in \mathcal{S}$ and a goal $g \in \mathcal{G}$. At the start of each episode, our method alternates between two phases: frontier-reaching exploration and main-goal exploration.

Phase 1: Frontier Reaching and Expansion. In the first phase, the agent is assigned a frontier sub-goal $s_{SG} \in \mathcal{F}$, where \mathcal{F} denotes the frontier set extracted from the replay buffer \mathcal{RB} (see Section 4.2). The agent then executes the goal-conditioned policy $\pi(a | s, s_{SG})$ to deliberately reach s_{SG} . By incrementally selecting such frontier sub-goals, the agent systematically expands the explored region of the state space in a curriculum-like fashion, while simultaneously improving its estimates of local dynamics and value functions.

Phase 2: Main-Goal Exploration. After reaching the frontier sub-goal s_{SG} , the agent transitions to the second phase and executes $\pi(a | s, s_G)$, where s_G denotes the main task goal. Starting ex-

ploration from s_{SG} makes reaching s_G more efficient, as the agent benefits from previously acquired experience near the boundary of known states.

Phase Switching Strategy. The transition between phases is governed by a hybrid deterministic–stochastic mechanism: 1. **Predefined horizons:** Each phase $i \in \{1, 2\}$ is assigned a maximum number of steps H_i , ensuring balanced allocation of exploration. 2. **Probabilistic early termination:** If during Phase 1 the agent encounters a novel state s with a visitation count of $N_{\mathcal{RB}}(s) \leq N_{\text{thr}}$ ($N_{\text{thr}} = 1$ in our experiments), it may switch immediately to Phase 2 with probability $p_{\text{switch}} \in (0, 1)$, even if H_1 has not yet been exhausted.

4.2 FRONTIER EXTRACTION

A critical aspect of our method is the identification of the frontier \mathcal{F} from which the sub-goal is selected for the first phase. Those sub-goals are represented as state-action pairs (s, a) , instead of plain states. We initially filter the agent’s past experiences from the replay buffer to select the best candidates. State-actions considered less novel or “very well known” are filtered out at this stage. In practice, we first rank the visited state-actions based on a familiarity score F and exclude the *familiar* ones with a score above a threshold F_{π}^{thr} . The motivation is to maintain the focus of the exploration away from the increasingly more visited states, whose transitions will be occupying an increasingly larger part of the experience replay buffer. Formally, this filter can be expressed as:

$$\mathcal{F} = \{(s, a) \in \mathcal{RB} : F_{\pi}(s) < F_{\pi}^{\text{thr}}\}$$

The potential sub-goals are obtained from the same state-actions being inserted in the replay buffer \mathcal{RB} , which are filtered to maintain a continuously updated frontier, in the same manner an *Open list* and a *Closed list* is used in search. The frontier is populated with all state-actions that have been visited at least once and have a familiarity score below a threshold F_{π}^{thr} , as well as those actions on the newly states that have not yet been tried. More specifically, when a new state s is visited for the first time, we insert all possible state-action pairs (s, a_i) for all available actions $a_i \in \mathcal{A}$ into the frontier.

For each frontier state-action pair, the additional relevant information recorded is its visitation counts, $N(s, a)$, as well as its *familiarity score*, $F(s)$.

Definition 1 (State Familiarity). *Let \mathcal{RB} denote the replay buffer containing all past experiences of an agent, and let $N_{\mathcal{RB}}(s, a)$ be the visitation count of a state–action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$ within \mathcal{RB} . The *familiarity* of s with respect to \mathcal{RB} is defined as*

$$F_{\mathcal{RB}}(s, a) = \frac{1}{1 + N_{\mathcal{RB}}(s, a)^{-1}}.$$

Such definition ensures that $F_{\mathcal{RB}}(s, a) \rightarrow 1$ as (s, a) becomes frequent in \mathcal{RB} , and $F_{\mathcal{RB}}(s, a) \rightarrow 0$ when (s, a) is rare. Besides state familiarity, we also define trajectory familiarity.

Definition 2 (Trajectory Familiarity). *For a trajectory $\tau = \langle s_1, s_2, \dots, s_k \rangle$ resulting from running goal-conditioned policy π for a goal s_k , the familiarity of the terminal state s_k is defined recursively as*

$$F_{\pi}(s_k) = \prod_{i=1}^k \frac{1}{1 + N_{\mathcal{RB}}(s_i)^{-1}}.$$

Assuming that we learn consistently and that policy π conditioned on state s_{k-1} results in trajectory $\langle s_1, s_2, \dots, s_{k-1} \rangle$, we can calculate trajectory familiarity for state s_k using the current state’s visitation counts and trajectory novelty the previous one: $F_{\pi}(s_k) = \frac{1}{1 + N(s_k)^{-1}} F_{\pi}(s_{k-1})$.

Motivation for such a definition is that when reaching a sub-goal, if the current policy succeeds in reaching it through familiar states, that should indicate that the agent has mastered reaching that state and can focus on further states. Using products in the calculation ensures the balance of the influence of trajectory length and the effect of familiarity of individual states. This strategy, particularly when combined with the probabilistic early switching mechanism, ensures that while the frontier is gradually populated with states near the expanding boundary of the familiar region, the agent concurrently gains experiences in states that are adjacent and relevant to each chosen sub-goal. This promotes a more consistent and thorough exploration.

4.3 SUB-GOAL SELECTION

The remaining state-action pairs that form the frontier \mathcal{F} after the filtering are then ranked and prioritized. This prioritization is determined by minimizing a combination of the following cost factors:

Novelty Cost c_n This cost penalizes more novel states, thereby favoring those familiar states that are more visited while still not overly familiar (since they have passed the initial filtering stage). This is based on the idea that the agent should first focus on mastering sub-goals it already practices before continuing further. Additionally, states visited extremely infrequently might be outliers or part of highly stochastic regions not yet suitable for directed exploration.

Cost-to-Come c_c (from the initial state to the sub-goal): This is estimated directly using the learned Q-values, representing the expected cumulative reward (or cost, in our negative reward setting) to reach the potential sub-goal from the episode's starting state, calculated as $\max_{a \in \mathcal{A}} Q(s_1, s_{SG})$.

Cost-to-Go c_g (from sub-goal to main goal): This is the estimated cost from the potential sub-goal to the ultimate task goal, again derived from the learned Q-values as $\max_{a \in \mathcal{A}} Q(s_{SG}, s_G)$.

Thus, the prioritization of the filtered goals can be formalized as the sum of each one's cost-to-come c_c and cost-to-go c_g , weighted by \mathbf{w} , multiplied by the novelty cost, which is also weighted with a weight-exponent w_n :

$$\text{softmin}_{s,a} (c_n(s, a)^{w_n} \mathbf{w}^\top \mathbf{c}(s)).$$

Where:

$$\begin{aligned} c_n(s, a) &= \sigma(z(-N(s, a))), \\ \mathbf{w}^\top &= [w_c \quad w_g], \\ \mathbf{c}(s) &= [c_c(s) \quad c_g(s)]^\top, \\ z(x) &= \frac{x - \mathbb{E}[X]}{\text{Var}(X)}. \end{aligned}$$

Thus, the state with the optimal combined score is selected as the next sub-goal for the agent in Phase 1.

5 EXPERIMENTS

For our experiments we aimed to set up situations which require deliberate exploration and a more thorough coverage of the state space to be solved. We strove to answer the following: (a) Does SIERL enable consistently succeeding in environments where goal discovery is non-trivial? (b) In which cases and in which aspects SIERL is more promising than its competitors? (c) Which components enable SIERL to perform well?

5.1 SETUP

Our experiments are designed to evaluate the performance of SIERL in scenarios that demand deliberate exploration and a comprehensive understanding of the state space. We use discrete state and action environments, where goal discovery is non-trivial due to sparse rewards and deceptive rewards from “trap” obstacles. We adjusted the rewards such that a signal of -1 is given for each step, and a reward of 0 upon reaching the goal, effectively turning the task into a shortest path problem. During evaluation, we run 10 main-goal reaching episodes as well as 10 random-goal reaching episodes for each method, reporting the mean success rate and standard error, in order to capture the methods' capacity to generalize while learning with a specific goal. All methods are run with 5 seeds to account for variance. The environments used are a subset of the MiniGrid framework (Chevalier-Boisvert et al.) and the experiments were set up using RLHive (Patil et al.).

These MiniGrid environments are minimalistic 2D grid worlds set up with a discrete action space representing moving left, right, up, or down. The state space is fully observable, with the agent's goal being to reach a specific static goal state. The agent's observation is a grid containing information

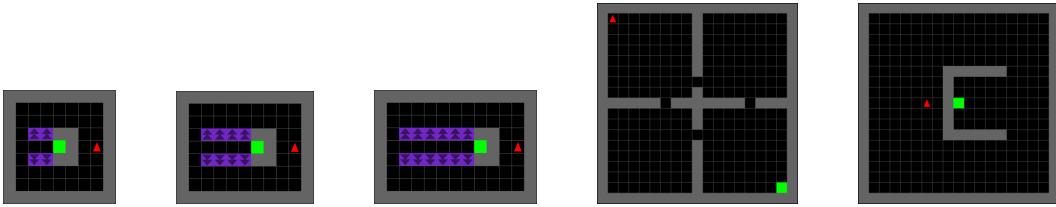


Figure 1: The MiniGrid Hallway room variants. From left to right: 2-, 4-, and 6- steps-long hallways, FourRooms, and BugTrap

about its location, the walls/obstacles, and the goal location. We specifically used several custom-made variants of a *Hallway* room, the *FourRooms* room, and a typical *BugTrap* room.

Hallway variants: These are challenging environments, containing a hallway flanked with “slippery” unidirectional tiles along the sides, as shown in Figure 1. The goal lies at the end of each corridor and the agent is required to perform a precise (albeit repetitive) sequence of actions to reach its end.

BugTrap: In this room, the agent has to navigate around a concave enclosure to reach the goal on the other side. Being more open requires the agent to progressively explore a larger region of state-action space until reaching the goal.

FourRooms: The agent is required to navigate from one corner of a square space comprised of 4 rooms to the diagonally opposite corner, through doors between the rooms. Reaching arbitrary locations in this more segregated space is a harder task than in the other cases.

5.2 RESULTS

Main goal success rates: The evaluation performance in the success rate for reaching the main goal is shown in the upper part of Figure 2. In all three Hallway variants SIERL performed on par with the most competitive baselines, such as Novelty bonuses, while outperforming HER, and Q-Learning. Specifically, on the small enough 2- and 4-step long Hallways, Random-goals Q-learning performs similarly as well; however, its performance is hampered on the larger 6-step long variant, following closely behind that of Novelty bonuses, whose performance is also impacted, albeit to a smaller degree. Nonetheless, SIERL is always able to discover and learn the main goal for all seeds.

In FourRooms, SIERL performs comparably well to HER but less so compared to Novelty bonuses. It is notable that succeeding in such an environment requires systematic coverage of the state-action space, which is accomplished via intrinsic rewards but not by relying solely on random exploration. This indicates SIERL is able to learn on the less accessible parts of the state space and, contrary to Novelty bonuses, it accomplishes that without tampering with the reward signal, but rather by guiding the agent’s exploration and thus adjusting the experience distribution to improve learning.

Random goal success rates: The success rate during evaluation for reaching uniformly sampled random goals is shown in the lower part of Figure 2. In all cases, the only methods capable of solving for arbitrarily set goals are SIERL and Random-goals Q-learning. In the smallest 2-step Hallway variant SIERL outperforms Random-goals, while this is less pronounced in the harder variants, where the latter continues improving at a slower pace. Notably, on the two larger environments of BugTrap and FourRooms, SIERL clearly outperforms every baseline. Arguably, the capacity of the Q-value network’s architecture imposes a limit on the simultaneous learning of both a wide range of goals and a specific main goal. Both SIERL, and a random-goal focused method leverage this capacity better than the other methods, while each trading off main-goal and random-goal focus in different degrees.

This behavior can possibly be attributed to having a more diverse goal distribution in the experiences’ transitions. It is also notable that SIERL is able to reach, learn on, and set sub-goals from a larger portion of the state space than HER, without relying on augmenting its experiences. The more systematic training with goals in a gradually expanding subset of the state space might prove bene-

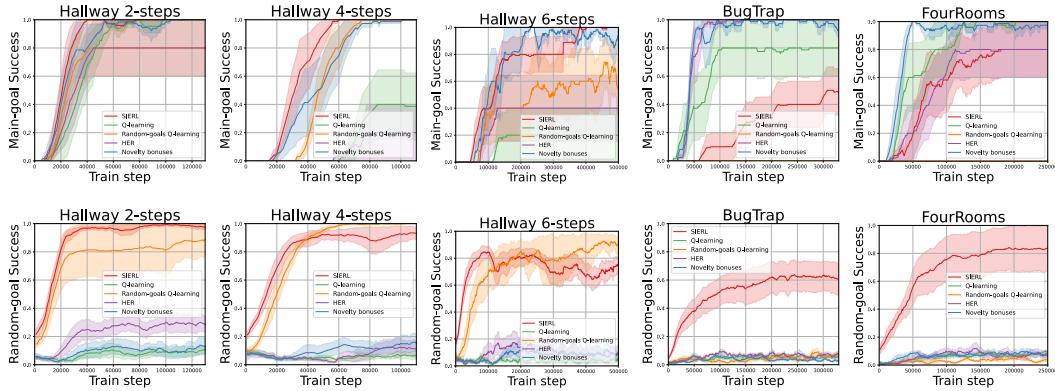


Figure 2: Main-goal (top row) and random-goal (bottom row) performance for the Hallway variant in columns. SIERL achieves a remarkable performance for both criteria at the same time, matched by no other method.

ficial to such generalization, provided those sub-goals are feasible, while at the same time managing to consistently learn to reach a main goal.

5.3 ABLATION STUDIES

To identify the crucial components enabling SIERL’s success, we performed a series of ablation studies on the most challenging environment variant, to observe performance differences in several aspects. The main-goal, random-goal, and sub-goal (during training) performance of all variants was examined. The core ideas of SIERL are: guiding the exploration by gradually expanding the state space’s well-learned region, while pursuing sub-goals towards the most promising direction of expansion of the region’s frontier.

The first aspect we ablated was the early switching mechanism of the first phase of exploration. This way, the agent’s experience gathering when pursuing sub-goals will extend without constraints further past the frontier of the familiar region, which contains the prospective sub-goals. Subsequently, focused on the contribution of the frontier extraction from experience filtering using the *familiarity* measure. By removing the extraction, the state-actions of which will be prioritized (the frontier) consist now of the complete set of experiences the agent has gathered, including all frequently tried state-actions. Lastly, we ablated the prioritization strategy of SIERL. In this case, the filtered states are not subjected to any scoring, and the sub-goal is picked at random with uniform probability. The aim is to evaluate the effectiveness of the prioritization strategy.

The ablation experiments’ results are shown in Figure 3. While random-goal performance seems unaffected for all variants, barring one, all of them exhibit a negative impact on either the ramp-up time or stability in reaching the main goal. Specifically, removing frontier prioritization for selecting sub-goals results in notably worse stability on learning for the main goal. Likewise, ablating early switching impacts main-goal discovery and learning. The seemingly better case of ablating the frontier filtering shows better random-goal performance, which is expected as the agent is consistently provided with a wider range of goal-conditioned experiences; however, it struggles to consistently learn on the main-goal. These observations further reinforce our understanding that SIERL demonstrates the capacity to stably balance learning on both types of goals with the same sample-efficiency as other competitive single-goal focused methods.

6 CONCLUSION

In this work, we presented SIERL, a method for Search Inspired Exploration in RL. This method is based on the principle that by gradually expanding the frontier of the explored region of the state space using sub-goal setting, the agent is able to efficiently cover the state space while learning a robust goal-conditioned behavior. SIERL exhibits competitive performance in reaching the main

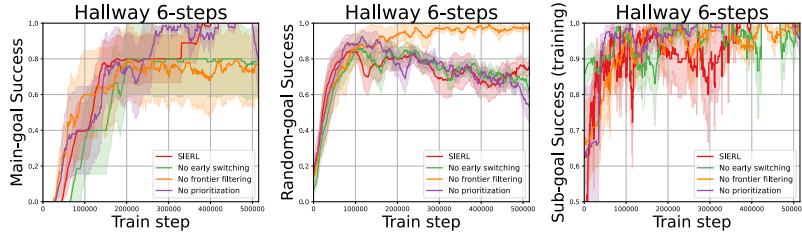


Figure 3: Success rate for reaching: the main goal (left), random goals (center), and “lateral” sub-goals (right). Most notably, removing early switching hampers SIERL’s ramp-up time and stability, while removing frontier prioritization reduces main-goal stability.

goal, while simultaneously learns to reach any other state within range of its familiar region; a property all of the baselines lack.

We demonstrated that this method is particularly suitable for hard exploration environments where getting from start to goal requires strictly executing a sequence of actions. Through ablation studies, we have shown that keeping the exploration’s first phase within the familiar region (with early switching) and by selecting its sub-goals by prioritizing states in a frontier, which is extracted by filtering the agent’s experiences, are all crucial components for SIERL’s success.

7 LIMITATIONS

In its present implementation SIERL is limited to discrete-state action spaces as it relies on visitation counts to define the notions of novelty. A measure that can provide a more generic notion of novelty on any location in the state-action space, usable in continuous state-action spaces as well, can enable SIERL to be used on a wider range of problems. This could be done by adopting one of the approximate methods for pseudo-counts. We believe that regardless of the way in which the visitation counting and novelty is replaced, the *familiarity* notion is preserved. The current implementation is also limited by the capacity of the replay buffer, depending on the state-action space size, dimensionality, and discretization scheme.

Although they are intuitive, several hyper-parameters are pre-determined and environment-dependent, providing opportunities for exploring more environment-agnostic definitions and adaptations. Determining the familiarity threshold is dependent on the size of the state-action space and the distance between start and goal. A broader concept of familiarity would be linked more directly to the degree the agent has learned about parts of the state-action space, rather than assuming this to be so based on experience counting.

Similarly, there is room for improving the phase lengths and phase-switch timing. While also presently environment-dependent and fixed, these parameters can benefit from an implementation more reliant on the agent’s learning at each point during training. Ideally, selecting a new goal and determining the right time to do so should be done, aiming to balance pursuing novelty and providing “practicing” for a goal-conditioned agent.

LLM USAGE STATEMENT

During the preparation of this document, an LLM was used for grammar, punctuation, and wording improvements. The core ideas, research, and conclusions are the authors’ own.

REFERENCES

Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. URL <http://arxiv.org/abs/2109.00157>.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem*. 47(2):235–256. ISSN 08856125. doi: 10.1023/A:1013689704352. URL <http://link.springer.com/10.1023/A:1013689704352>.

Junik Bae, Kwanyoung Park, and Youngwoon Lee. TLDR: Unsupervised goal-conditioned RL via temporal distance-aware representations. URL <https://openreview.net/forum?id=deywgeWmL5>.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. 29.

Dimitri P. Bertsekas. Approximate dynamic programming.

Homanga Bharadhwaj, Animesh Garg, and Florian Shkurti. LEAF: Latent exploration along the frontier. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 677–684. doi: 10.1109/ICRA48506.2021.9560922. URL <https://ieeexplore.ieee.org/document/9560922> ISSN: 2577-087X.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations (ICLR 2019)*, pp. 1–17. URL <https://www.research.ed.ac.uk/en/publications/exploration-by-random-network-distillation>.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In *Advances in neural information processing systems 36, new orleans, LA, USA*.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. 590(7847):580–586. ISSN 1476-4687. doi: 10.1038/s41586-020-03157-9. URL <https://www.nature.com/articles/s41586-020-03157-9>. Publisher: Nature Publishing Group.

Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc. URL <https://proceedings.neurips.cc/paper/2019/hash/83715fd4755b33f9c3958e1a9ee221e1-Abstract.html>.

Yohei Hayamizu, Saeid Amiri, Kishan Chandan, Keiki Takadama, and Shiqi Zhang. Guiding robot exploration in reinforcement learning via automated planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 31, pp. 625–633. doi: 10.1609/icaps.v31i1.16011. URL <https://ojs.aaai.org/index.php/ICAPS/article/view/16011>. ISSN: 2334-0843.

Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. Exploration via elliptical episodic bonuses. 35:37631–37646. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/f4f79698d48bd1a6dec20583724182b-Abstract-Conference.html.

Jakob J. Hollenstein, Erwan Renaudo, Matteo Saveriano, and Justus Piater. Improving the exploration of deep reinforcement learning in continuous domains using planning for policy search. URL <http://arxiv.org/abs/2010.12974>.

Edward S. Hu, Richard Chang, Oleh Rybkin, and Dinesh Jayaraman. Planning goals for exploration. URL <https://openreview.net/forum?id=6qeBuZSo7Pr>.

Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. 85:1–22. ISSN 1566-2535. doi: 10.1016/j.inffus.2022.03.003. URL <https://www.sciencedirect.com/science/article/pii/S1566253522000288>.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. URL <http://arxiv.org/abs/2201.08299>.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and R. Munos. Count-based exploration with neural density models. URL <https://www.semanticscholar.org/paper/Count-Based-Exploration-with-Neural-Density-Models-Ostrovski-Bellemare/12f67fb182bc934fc95ce97acff553d83e2ca72e>.

Darshan Patil, Ali Rahimi-Kalahroudi, Hadi Nekoei, Sai Krishna Gottipati, Mohammad Reza Samami, Kshitij Gupta, Sriyash Poddar, Artem Zholus, Maryam Hashemzadeh, Xutong Zhao, and Sarath Chandar. RLHive. URL <https://github.com/chandar-lab/RLHive>

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for markov decision processes. 74(8):1309–1331. Publisher: Elsevier.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3a20f62a0af1aa152670bab3c602feed-Abstract.html.

Lenart Treven, Jonas Hübotter, Florian Dorfler, and Andreas Krause. Efficient exploration in continuous-time model-based reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 42119–42147. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/836012122f3de08aeeae67369b087964-Abstract-Conference.html.

A SIERL IMPLEMENTATION

SIERL, presented in Algorithm 2, implements the two-phase exploration strategy detailed previously. The core of the algorithm operates in a continuous loop. Each iteration begins with the goal-setting and switching logic, including initialization of the environment, yielding an initial state s_1 and the main task goal s_G . After a reset, the agent’s state is set to s_1 and its current sub-goal, `agent.goal`, is set to s_G , while the trajectory step counter t is initialized to zero at the start of each phase.

At the start of each iteration is the decision to switch sub-goals, determined based on the following conditions: First, if the agent’s current state s' matches the main goal s_G , then a reset is performed and a sub-goal is generated, thus starting Phase 1. The sub-goal is selected. Otherwise, if the current state s' matches the state component of the current `agent.goal`, then the Phase 1 sub-goal was just reached and it is time to move to Phase 2 by directing the exploration towards s_G . If that is not the case either, then if it is time for an early switch from Phase 1, or if the current trajectory length t has reached a predefined maximum M , then the transition from Phase 1 to 2 takes place likewise.

Early switching from Phase 1 is performed by calling `early_switch()` (Algorithm 4) which samples a random variable to determine whether to switch based on a predefined switching probability as follows: if the agent encounters a state s' it has never visited before (i.e., `agent.visitations[s'] == 0`), a switch occurs with a probability $P_{\text{switch}}(\text{state_is_novel} = \text{true})$. Otherwise, if the agent’s state is not novel, the probability to switch to the next phase is $P_{\text{switch}}(\text{state_is_novel} = \text{false})$ (typically lower or zero).

The sub-goal for Phase 1 is obtained by calling `get_subgoal()` (Algorithm 3), which filters and prioritizes states from the replay buffer. This is achieved by first calling the `get_frontier()` method (Algorithm 5) to obtain a list of candidate frontier state-action pairs. By iterating through the agent’s replay buffer, the pairs (s, a) whose familiarity `agent.familiarity[s, a]` is higher than the maximum threshold F_{π}^{thr} , and those whose counts are above the minimum allowed percentile threshold $P_{10}(N)$. Subsequently, for each candidate frontier (s_f, a_f) pair from this list, a cost is calculated as was described in Subsection 4.2. The new sub-goal is then sampled, biased towards the state-action pair with the minimum calculated cost (e.g., by using a softmax distribution over the costs).

Subsequently, the agent selects an action a based on its current state s and the active `agent.goal` using its goal-conditioned policy (e.g., ε -greedy). Upon executing the action, the environment transitions to a new state s' and provides a reward r . The agent then updates its internal model, its Q-values or policy, using the experience $(s, a, r, s', \text{agent.goal})$, as well as its familiarity for the

last state-action with `update_familiarity()`, and the step counter t for the current phase is incremented. In this update step the batch of randomly sampled experiences can contain transitions with either the main goal or any other previous frontier sub-goal.

Finally, the agent's current state s is replaced by s' , and the loop continues. This interplay between pursuing generated sub-goals (Phase 1) and the main task goal (Phase 2), guided by the frontier extraction and prioritization logic, allows SIERL to systematically expand the familiar region, while moving towards the main goal.

Algorithm 2: SIERL Algorithm

Input: Agent `agent`; Environment `env`

```

while true do
  if  $s_G = \text{null} \vee s = s_G$  then
    |  $s, s_G \leftarrow \text{env.reset}(); t \leftarrow 0; f \leftarrow 0$                                 // episode reset
    | agent.goal = agent.get_subgoal( $s$ )
  else if  $s = \text{agent.goal}$  then
    | agent.goal  $\leftarrow s_G; t \leftarrow 0$                                 // sub-goal reached, switch
  else if (agent.early_switch( $s$ )  $\wedge \text{agent.goal} \neq s_G$ )  $\vee (t \geq M)$  then
    | agent.goal  $\leftarrow s_G; t \leftarrow 0$                                 // unwanted exploration or timeout
  else
    |  $a \leftarrow \text{agent.}\pi(s, \text{agent.goal})$ 
    |  $s', r \leftarrow \text{env.step}(s, a)$ 
    | agent.}\pi.\text{update}(s, a, r, s', \text{agent.goal})
    | agent.familiarity[ $s, a$ ]  $\leftarrow \text{update_familiarity}(s, a, f)$ 
    |  $f \leftarrow \text{agent.}\mathbf{familiarity}[s, a]$ 
    | foreach  $a' \in \mathcal{A}$  do                                // add all possible  $(s', a)$ 
      | | agent.frontier.insert( $s', a'$ )
    | end
    | if  $f < F_\pi^{\text{thr}}$  then                                // exclude too familiar  $(s, a)$ 
      | | agent.frontier  $\leftarrow \text{agent.frontier} \cup \{(s, a)\}$ 
    | end
    |  $t \leftarrow t + 1$ 
    |  $s \leftarrow s'$ 
  end
end

```

Algorithm 3: `get_subgoal()` method

Input: Current state: s

```

 $costs \leftarrow \{\}$ 
foreach  $(s_f, a_f) \in \text{agent.get_frontier}()$  do
  |  $c \leftarrow \sigma(-z(\text{agent.}\mathbf{visitations}[s_f, a_f]))^{w_n}$                                 // novelty cost
  | |  $\cdot \sigma(z(w_l \text{agent.}Q(s, s_f) +$                                 // cost-to-reach
  | | |  $w_c \text{agent.}Q(s_l, s_f) +$                                 // cost-to-come
  | | |  $w_g \text{agent.}Q(s_f, s_G))$                                 // cost-to-go
  | |  $costs \leftarrow costs \cup \{c\}$ 
end
 $subgoal \leftarrow \text{sample}(\text{softmin}(costs))$                                 // sample based on minimum cost
return subgoal

```

Algorithm 4: `early_switch()` method

Input: Current state: s
Switching probabilities: P_{switch}

```
state_is_novel  $\leftarrow$  agent.visitations[ $s$ ] == 0
early_switch  $\leftarrow$  random()  $<$   $P_{\text{switch}}(state\_is\_novel)$ 
return early_switch
```

Algorithm 5: `get_frontier()` method

Input: Familiarity threshold: F_{π}^{thr}

```
frontier  $\leftarrow$  agent.open_list
foreach  $(s, a) \in frontier$  do
    if agent.familiarity[ $s, a$ ]  $>$   $F_{\pi}^{\text{thr}}$  then
        | frontier  $\leftarrow$  frontier  $\setminus \{(s, a)\}$  // exclude too familiar  $(s, a)$  pairs
    end
end
foreach  $(s, a) \in frontier$  do
    if agent.visitations[ $s, a$ ]  $<$   $P_{10}(N)$  then
        | frontier  $\leftarrow$  frontier  $\setminus \{(s, a)\}$  // exclude too novel  $(s, a)$  pairs
    end
end
return frontier
```
