

000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019 020 021 022 023 024 025 026 027 028 029 030 031 032 033 034 035 036 037 038 039 040 041 042 043 044 045 046 047 048 049 050 051 052 053 SEARCH INSPIRED EXPLORATION FOR REINFORCEMENT LEARNING

Anonymous authors

Paper under double-blind review

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

054 generate sub-goals that extend progressively farther from the start state. In doing so, our method
 055 effectively constructs a curriculum without requiring manually designed tasks or environments of
 056 varying difficulty.

057 In this work, we introduce *Search-Inspired Exploration* (SIERL), a novel approach that guides ex-
 058 ploration by setting sub-goals informed by the agent’s learning progress. Our main contributions are:
 059 1. We propose a principled sub-goal selection mechanism that systematically expands exploration by
 060 defining a frontier of experience and prioritizing sub-goals using cost-to-come and cost-to-go esti-
 061 mates. 2. We design a novel *Hallway* environment that enables fine-grained control over exploration
 062 difficulty by varying the length of action sequences required to succeed. 3. We show that SIERL
 063 leads to more efficient exploration in *discrete* sparse-reward settings. 4. We present an empirical
 064 study that disentangles the contributions of individual components and identifies which mechanisms
 065 most effectively improve exploration for goal-conditioned agents.

NEW

066 The remainder of this paper is organized as follows: section 2 reviews related literature; section 3 in-
 067 troduces the necessary preliminaries; section 4 details our algorithm; section 5 presents experimental
 068 results; and sections 6 and 7 discuss conclusions and future directions.

070 2 RELATED WORK

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

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

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

106 **Frontier- and confidence-driven exploration.** Several recent methods refine goal-based explo-
 107 ration 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 (TLD) (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 alternates 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 gappx: 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

162

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

163

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

172

173

174 a goal state and zero and small negative rewards (costs) for all other transitions. Thus, instead of the
 175 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
 176 for each action. Goal states $s_G \in \mathcal{S}$ can be absorbing, meaning $P_a(s_G | a, s_G) = 1$ for all $a \in \mathcal{A}$,
 177 and cost free, meaning $c(a, s_G) = 0$ for all $a \in \mathcal{A}$. This transforms the problem into one of finding
 178 an optimal path or policy for each goal, with the objective of achieving a desired terminal goal or
 179 condition. It has been shown that (partially observable) MDPs can be transformed into equivalent
 180 (partially observable) Goal MDPs (Bertsekas)

181

3.3 EXPLORATION VS EXPLOITATION DILEMMA

182

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

189

190

4 METHOD

191

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

198

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

205

206 A pseudo-code description of SIERL is provided in algorithm 1, with full implementation details in
 207 Appendix A.

208

4.1 TWO-PHASE EXPLORATION STRATEGY

209

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

213

214

Phase 1: Frontier Reaching and Expansion. In the first phase, the agent is assigned a frontier
 215 sub-goal $s_{SG} \in \mathcal{F}$, where \mathcal{F} denotes the frontier set extracted from the replay buffer \mathcal{RB} (see
 216 subsection 4.2). The agent then executes the goal-conditioned policy $\pi(a | s, s_{SG})$ to deliberately
 217 reach s_{SG} . By incrementally selecting such frontier sub-goals, the agent systematically expands the

216 explored region of the state space in a curriculum-like fashion, while simultaneously improving its
 217 estimates of local dynamics and value functions.

218 **Phase 2: Main-Goal Exploration.** After reaching the frontier sub-goal s_{SG} , the agent transitions
 219 to the second phase and executes $\pi(a | s, s_G)$, where s_G denotes the main task goal. Starting ex-
 220 ploration from s_{SG} makes reaching s_G more efficient, as the agent benefits from previously acquired
 221 experience near the boundary of known states.

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

229 4.2 FRONTIER EXTRACTION

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

$$239 \quad \mathcal{F} = \{(s, a) \in \mathcal{RB} : F_{\pi}(s) < F_{\pi}^{thr}\} \quad (1)$$

241 The potential sub-goals are obtained from the same state-actions being inserted in the replay buffer
 242 \mathcal{RB} , which are filtered to maintain a continuously updated frontier, in the same manner an *Open list*
 243 and a *Closed list* is used in search. The frontier is populated with all state-actions that have been
 244 visited at least once and have a familiarity score below a threshold F_{π}^{thr} , as well as those actions
 245 on the newly states that have not yet been tried. More specifically, when a new state s is visited for
 246 the first time, we insert all possible state-action pairs (s, a_i) for all available actions $a_i \in \mathcal{A}$ into the
 247 frontier. **For the edge-case when the frontier set obtained happens to be empty, we populate it with** NEW
 248 **only the main goal, effectively turning that episode into a typical main-goal pursuit.**

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

251 **Definition 1** (State Familiarity). *Let \mathcal{RB} denote the replay buffer containing all past experiences of*
 252 *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} .*
 253 *The familiarity of s with respect to \mathcal{RB} is defined as*

$$254 \quad F_{\mathcal{RB}}(s, a) = \frac{1}{1 + N_{\mathcal{RB}}(s, a)^{-1}}. \quad (2)$$

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

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

$$262 \quad F_{\pi}(s_k) = \prod_{i=1}^k \frac{1}{1 + N_{\mathcal{RB}}(s_i)^{-1}}. \quad (3)$$

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

266 Motivation for such a definition is that when reaching a sub-goal, if the current policy succeeds
 267 in reaching it through familiar states, that should indicate that the agent has mastered reaching

270 that state and can focus on further states. Using products in the calculation ensures the balance
 271 of the influence of trajectory length and the effect of familiarity of individual states. This strategy,
 272 particularly when combined with the probabilistic early switching mechanism, ensures that while the
 273 frontier is gradually populated with states near the expanding boundary of the familiar region, the
 274 agent concurrently gains experiences in states that are adjacent and relevant to each chosen sub-goal.
 275 This promotes a more consistent and thorough exploration.

276

277 4.3 SUB-GOAL SELECTION

278

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

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

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

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

292 Thus, the score used for prioritizing the filtered goals can be formulated as the sum of each one’s
 293 cost-to-come c_c and cost-to-go c_g , weighted by w , multiplied by the novelty cost, which is also
 294 weighted with a weight-exponent w_n . **The sub-goal is sampled with probability assigned by applying**
 295 **a softmax to this set of scores for the frontier state-actions:**

$$P((s, a) = (s_{SG}, a_{SG})) = \text{softmax}_{(s, a) \in \mathcal{F}} (c_n(s, a)^{w_n} \mathbf{w}^\top \mathbf{c}(s)). \quad (4)$$

296 Where:

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

297 NEW

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

302

303 5 EXPERIMENTS

304

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

310

311 5.1 SETUP

312

313 Our experiments are designed to evaluate the performance of SIERL in scenarios that demand de-
 314 liberate exploration and a comprehensive understanding of the state space. We use discrete state
 315 and action environments, where goal discovery is non-trivial due to sparse rewards and deceptive
 316 rewards from “trap” obstacles. We adjusted the rewards such that a signal of -1 is given for each
 317 step, and a reward of 0 upon reaching the goal, effectively turning the task into a shortest path

324 problem. During evaluation, we run 10 main-goal reaching episodes as well as 10 random-goal
 325 reaching episodes for each method, reporting the mean success rate and standard error, in order to
 326 capture the methods’ capacity to generalize while learning with a specific goal. All methods are
 327 run with 10 seeds to account for variance. The environments used are a subset of the MiniGrid
 328 framework (Chevalier-Boisvert et al.) and the experiments were set up using RLHive (Patil et al.).
 329

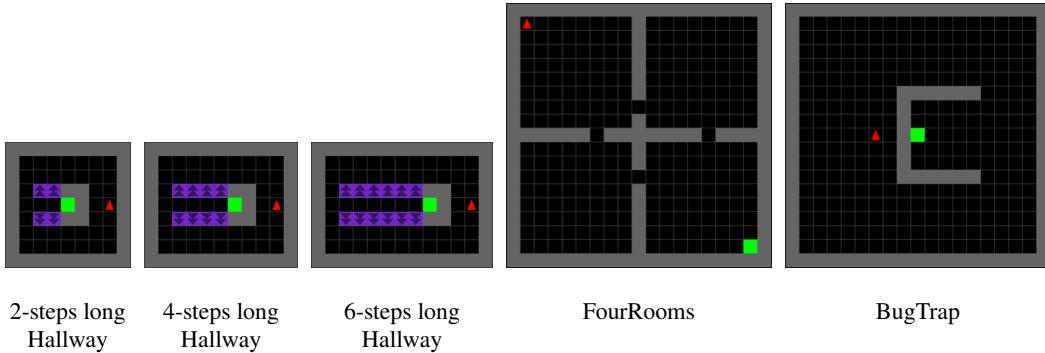


Figure 1: The MiniGrid room variants.

344 These MiniGrid environments are minimalist 2D grid worlds set up with a discrete action space
 345 representing moving left, right, up, or down. The state space is fully observable, with the agent’s goal
 346 being to reach a specific static goal state. The agent’s observation is a grid containing information
 347 about its location, the walls/obstacles, and the goal location. We specifically used several custom-
 348 made variants of a *Hallway* room, the *FourRooms* room, and a typical *BugTrap* room.
 349

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

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

357 **FourRooms:** The agent is required to navigate from one corner of a square space comprised of
 358 4 rooms to the diagonally opposite corner, through doors between the rooms. Reaching arbitrary
 359 locations in this more segregated space is a harder task than in the other cases.

360 5.2 RESULTS

361 **Main goal success rates:** The evaluation performance in the success rate for reaching the main goal
 362 is shown in the upper part of Figure 2. In all three Hallway variants SIERL performed on par with the
 363 most competitive baselines, such as Novelty bonuses, while outperforming HER, and Q-Learning.
 364 More details about the baselines are presented in Appendix C. Specifically, on the small enough
 365 2- and 4-step long Hallways, Random-goals Q-learning performs similarly as well; however, its
 366 performance is hampered on the larger 6-step long variant, following closely behind that of Novelty
 367 bonuses, whose performance is also impacted, albeit to a smaller degree. Nonetheless, SIERL is
 368 always able to discover and learn the main goal for all seeds.
 369

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

376 **Random goal success rates:** The success rate during evaluation for reaching uniformly sampled
 377 random goals is shown in the lower part of Figure 2. In all cases, the only methods capable of solving
 378 for arbitrarily set goals are SIERL and Random-goals Q-learning. In the smallest 2-step Hallway

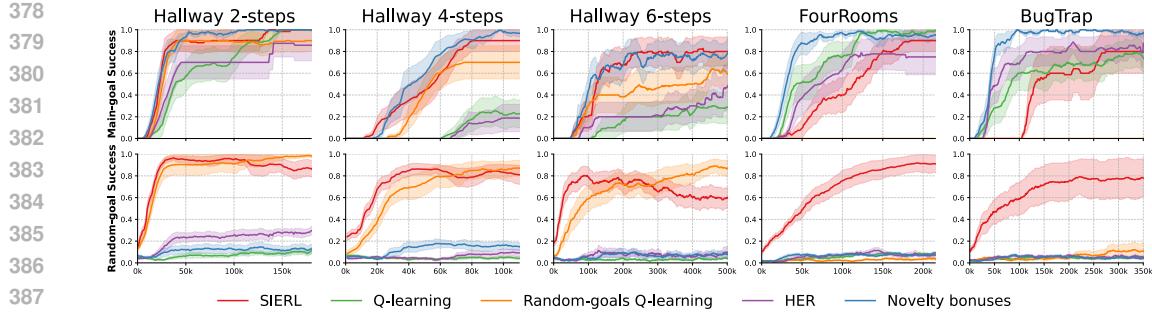


Figure 2: Main-goal (top row) and random-goal (bottom row) performance for the Hallway variant in columns. SIERL achieves a remarkable performance for both criteria at the same time, matched by no other method.

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 beneficial 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 performance on learning for the main goal in FourRooms. Likewise, ablating early-switching slightly worsens main-goal performance, although random-goal performance appears more stable on 6-steps Hallway. 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 demon-

432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
strates the capacity to stably balance learning on both types of goals with the same sample-efficiency as other competitive single-goal focused methods.

NEW

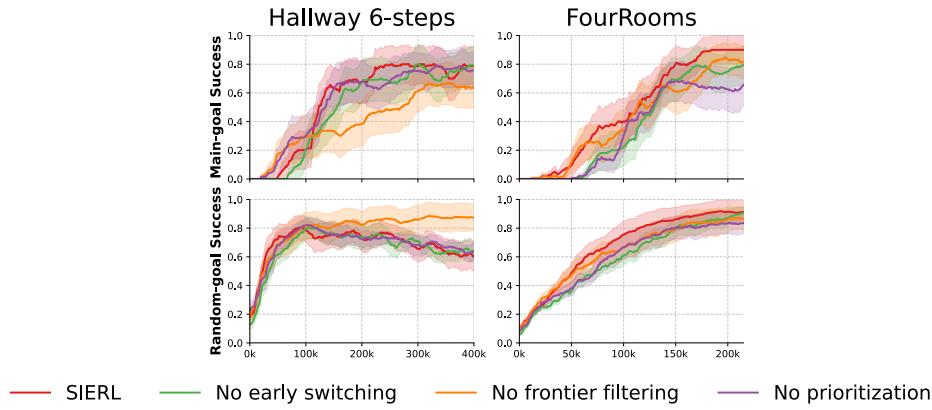


Figure 3: Success rate for reaching: the main goal (top), and random goals (bottom) for ablated variants, in 6-steps Hallway (left) and FourRooms (right). Most notably, removing frontier filtering or prioritization worsens SIERL’s main-goal success, while removing early switching shows a smaller negative influence.

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. In discrete settings, SIERL exhibits competitive performance in reaching the main goal, while simultaneously learns to reach any other state within range of its familiar region; a property all of the baselines lack.

NEW

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

486 more reliant on the agent’s learning at each point during training. Ideally, selecting a new goal
 487 and determining the right time to do so should be done, aiming to balance pursuing novelty and
 488 providing “practicing” for a goal-conditioned agent.
 489

490 **LLM USAGE STATEMENT**
 491

492 During the preparation of this document, an LLM was used for grammar, punctuation, and wording
 493 improvements. The core ideas, research, and conclusions are the authors’ own.
 494

495 **REFERENCES**
 496

497 Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey
 498 of exploration methods in reinforcement learning. URL <http://arxiv.org/abs/2109.00157>.
 499

500 Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit
 501 problem*. 47(2):235–256. ISSN 08856125. doi: 10.1023/A:1013689704352. URL <http://link.springer.com/10.1023/A:1013689704352>.
 502

503 Junik Bae, Kwanyoung Park, and Youngwoon Lee. TLDR: Unsupervised goal-conditioned RL via
 504 temporal distance-aware representations. URL <https://openreview.net/forum?id=deywgeWmL5>.
 505

506 Dimitri P. Bertsekas. Approximate dynamic programming.
 507

509 Homanga Bharadhwaj, Animesh Garg, and Florian Shkurti. LEAF: Latent exploration along the
 510 frontier. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 677–
 511 684. doi: 10.1109/ICRA48506.2021.9560922. URL <https://ieeexplore.ieee.org/document/9560922>. ISSN: 2577-087X.
 512

513 Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random
 514 network distillation. In *7th International Conference on Learning Representations (ICLR*
 515 *2019*), pp. 1–17. URL <https://www.research.ed.ac.uk/en/publications/exploration-by-random-network-distillation>.
 516

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

522 Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return,
 523 then explore. 590(7847):580–586. ISSN 1476-4687. doi: 10.1038/s41586-020-03157-9. URL
 524 <https://www.nature.com/articles/s41586-020-03157-9>. Publisher: Nature
 525 Publishing Group.
 526

527 Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. Curriculum-guided hindsight
 528 experience replay. In *Advances in Neural Information Processing Systems*, volume 32. Cur-
 529 ran Associates, Inc. URL <https://proceedings.neurips.cc/paper/2019/hash/83715fd4755b33f9c3958e1a9ee221e1-Abstract.html>.
 530

531 Yohei Hayamizu, Saeid Amiri, Kishan Chandan, Keiki Takadama, and Shiqi Zhang. Guiding
 532 robot exploration in reinforcement learning via automated planning. In *Proceedings of the*
 533 *International Conference on Automated Planning and Scheduling*, volume 31, pp. 625–633.
 534 doi: 10.1609/icaps.v31i1.16011. URL <https://ojs.aaai.org/index.php/ICAPS/article/view/16011>. ISSN: 2334-0843.
 535

537 Mikael Henaff, Roberta Raileanu, Minqi Jiang, and Tim Rocktäschel. Ex-
 538 ploration via elliptical episodic bonuses. 35:37631–37646. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/f4f79698d48bd1a6dec20583724182b-Abstract-Conference.html.
 539

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

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

545 Paweł Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement
 546 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>.

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

549 Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G.
 550 Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Pe-
 551 tersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran,
 552 Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep rein-
 553 forcement learning. 518(7540):529–533. ISSN 1476-4687. doi: 10.1038/nature14236. URL
 554 <https://www.nature.com/articles/nature14236>. Publisher: Nature Publishing
 555 Group.

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

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

561 Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain
 562 exploration for long horizon multi-goal reinforcement learning. In *Proceedings of the 37th Inter-
 563 national Conference on Machine Learning*, volume 119 of *ICML’20*, pp. 7750–7761. JMLR.org.

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

568 Lenart Treven, Jonas Hübotter, Florian Dorfler, and Andreas Krause. Efficient ex-
 569 ploration in continuous-time model-based reinforcement learning. In *Advances
 570 in Neural Information Processing Systems*, volume 36, pp. 42119–42147. URL
 571 https://proceedings.neurips.cc/paper_files/paper/2023/hash/836012122f3de08aeeeae67369b087964-Abstract-Conference.html.

572

573

574

575

576

577

578

579

580

581

582

583 **A SIERL IMPLEMENTATION**

584

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

590 At the start of each iteration is the decision to switch sub-goals, determined based on the following
 591 conditions: First, if the agent’s current state s' matches the main goal s_G , then a reset is performed
 592 and a sub-goal is generated, thus starting Phase 1. The sub-goal is selected. Otherwise, if the current
 593 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

594 case either, then if it is time for an early switch from Phase 1, or if the current trajectory length t has
 595 reached a predefined maximum M , then the transition from Phase 1 to 2 takes place likewise.
 596

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

603 The sub-goal for Phase 1 is obtained by calling `get_subgoal()` (algorithm 3), which filters and
 604 prioritizes states from the replay buffer. This is achieved by first calling the `get_frontier()` method
 605 (algorithm 5) to obtain a list of candidate frontier state-action pairs. By iterating through the agent's
 606 replay buffer, the pairs (s, a) whose familiarity `agent.familiarity[s, a]` is higher than the maximum
 607 threshold F_{π}^{thr} , and those whose counts are above the minimum allowed percentile threshold
 608 $P_{10}(N)$. Subsequently, for each candidate frontier (s_f, a_f) pair from this list, a cost is calculated as
 609 was described in subsection 4.2. The new sub-goal is then sampled, biased towards the state-action
 610 pair with the minimum calculated cost (e.g., by using a softmax distribution over the costs).

611 Subsequently, the agent selects an action a based on its current state s and the active agent's `goal`
 612 using its goal-conditioned policy (e.g., ε -greedy). Upon executing the action, the environment trans-
 613 itions to a new state s' and provides a reward r . The agent then updates its internal model, its
 614 Q-values or policy, using the experience $(s, a, r, s', \text{agent.goal})$, as well as its familiarity for the
 615 last state-action with `update_familiarity()`, and the step counter t for the current phase is incre-
 616 mented. In this update step the batch of randomly sampled experiences can contain transitions with
 617 either the main goal or any other previous frontier sub-goal.

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

Algorithm 2: SIERL Algorithm

Input: Agent `agent`; Environment `env`

```

622
623
624
625 while true do
626   if  $s_G = \text{null} \vee s = s_G$  then
627     |  $s, s_G \leftarrow \text{env.reset}(); t \leftarrow 0; f \leftarrow 0$                                 // episode reset
628     | agent.goal = agent.get_subgoal( $s$ )
629   else if  $s = \text{agent.goal}$  then
630     | agent.goal  $\leftarrow s_G; t \leftarrow 0$                                 // sub-goal reached, switch
631   else if (agent.early_switch( $s$ )  $\wedge \text{agent.goal} \neq s_G$ )  $\vee (t \geq M)$  then
632     | agent.goal  $\leftarrow s_G; t \leftarrow 0$                                 // unwanted exploration or timeout
633   else
634     |  $a \leftarrow \text{agent.}\pi(s, \text{agent.goal})$ 
635     |  $s', r \leftarrow \text{env.step}(s, a)$ 
636     | agent.}\pi.\text{update}(s, a, r, s', \text{agent.goal})
637     | agent.familiarity[ $s, a$ ]  $\leftarrow \text{update_familiarity}(s, a, f)$ 
638     |  $f \leftarrow \text{agent.familiarity}[s, a]$ 
639     | foreach  $a' \in \mathcal{A}$  do                                         // add all possible  $(s', a)$ 
640       | | agent.frontier.insert( $s', a'$ )
641     | end
642     | if  $f < F_{\pi}^{\text{thr}}$  then                                // exclude too familiar  $(s, a)$ 
643       | | agent.frontier  $\leftarrow \text{agent.frontier} \cup \{(s, a)\}$ 
644     | end
645     |  $t \leftarrow t + 1$ 
646     |  $s \leftarrow s'$ 
647   end
end

```

648
649
650
651 **Algorithm 3:** `get_subgoal()` method
652
653 **Input:** Current state: s
654
655 $costs \leftarrow \{\}$
656 **foreach** $(s_f, a_f) \in \text{agent.get_frontier}()$ **do**
657 $c \leftarrow \sigma(-z(\text{agent.visitations}[s_f, a_f]))^{w_n}$ // novelty cost
658 $\cdot \sigma(z(w_r \text{agent.Q}(s, s_f) +$ // cost-to-reach
659 $w_c \text{agent.Q}(s_l, s_f) +$ // cost-to-come
660 $w_g \text{agent.Q}(s_f, s_g)))$ // cost-to-go
661 $costs \leftarrow costs \cup \{c\}$
662 **end**
663 $subgoal \leftarrow \text{sample}(\text{softmin}(costs))$ // sample based on minimum cost
664 **return** $subgoal$
665
666
667
668
669
670
671 **Algorithm 4:** `early_switch()` method
672
673 **Input:** Current state: s
674 Switching probabilities: P_{switch}
675 $state_is_novel \leftarrow \text{agent.visitations}[s] == 0$
676 $early_switch \leftarrow \text{random}() < P_{\text{switch}}(state_is_novel)$
677 **return** $early_switch$
678
679
680
681
682
683
684
685 **Algorithm 5:** `get_frontier()` method
686
687 **Input:** Familiarity threshold: F_π^{thr}
688 $frontier \leftarrow \text{agent.open_list}$
689 **foreach** $(s, a) \in frontier$ **do**
690 **if** $\text{agent.familiarity}[s, a] > F_\pi^{\text{thr}}$ **then**
691 $frontier \leftarrow frontier \setminus \{(s, a)\}$ // exclude too familiar (s, a) pairs
692 **end**
693 **end**
694 **foreach** $(s, a) \in frontier$ **do**
695 **if** $\text{agent.visitations}[s, a] < P_{10}(N)$ **then**
696 $frontier \leftarrow frontier \setminus \{(s, a)\}$ // exclude too novel (s, a) pairs
697 **end**
698 **end**
699 **return** $frontier$
700
701

702 B AGENT DETAILS

703 NEW

704 All agents for our experiments use DQN (Mnih et al.) as the base agent. The corresponding hyper-
 705 parameters commonly used in all experiments are shown in Table 1. The additional hyperparameters
 706 for SIERL are shown in Table 2.

711 Hyperparameter	711 Value
c_n, c_c, c_g	[1.5, 1.0, 0.5]
P_{switch}	100%
H_1, H_2	[episode length - 1, 1]
softmax temperature	0.5
Q-Network LR	3×10^{-4}
Optimizer	Adam
Target hard update frequency	episode length
Initial Collect Steps	128
Batch Size	128
Discount Factor	0.95
ϵ -greedy init value	1.0
ϵ -greedy end value	0.1
ϵ -greedy decay steps	20,000
Seeds	18995728, 64493317, 49789456, 22114861, 50259734, 99918123, 71729146, 10365956, 83575762, 35232230

720 Table 1: Common hyperparameters for all environments.

	Hyperparameter	Value
Hallway 2-steps	F_π^{thr}	0.9
	Q-Network	Conv(3 \times [16])
	Replay Buffer	FC([16])
	episode length	100,000
Hallway 4-steps	F_π^{thr}	0.9
	Q-Network	Conv(3 \times [16])
	Replay Buffer	FC([16])
	episode length	300
Hallway 6-steps	F_π^{thr}	0.95
	Q-Network	Conv(3 \times [16])
	Replay Buffer	FC([16])
	episode length	400
FourRooms	F_π^{thr}	0.8
	Q-Network	Conv(7 \times [16])
	Replay Buffer	FC([16])
	episode length	300,000
BugTrap	F_π^{thr}	0.7
	Q-Network	Conv(7 \times [16])
	Replay Buffer	FC([16])
	episode length	300,000

725 Table 2: Environment-specific hyperparameters.

737 B.1 EXPERIENCE AND FRONTIER MANAGEMENT

738 SIERL’s frontier is used in the same way as the Open list is used in search algorithms. However,
 739 although the Open list is often formulated as being updated on the fly, with entries being added
 740 and removed during every step, SIERL’s frontier population is optimized to minimize excessive list
 741 manipulation actions, and redundant data storage to optimize memory use. In practice, the frontier
 742 is obtained from the Replay Buffer in a “lazy” manner whenever it is required in order to obtain a
 743 sub-goal.

744 To that end, a separate list is maintained containing a single entry for each unique state in the
 745 Replay Buffer, along with additional metadata, and it is the only SIERL’s data structure that is being
 746 updated after each agent’s step. This is a dictionary that contains the aforementioned unique state-
 747 action arrays (converted to hashable data types) as its entries’ indices or keys, and metadata such
 748 as visitation counts and familiarity values for its entries’ values. After each step, the newly inserted
 749 transition is used to update this list. This entails updating the transitions’ involved states’ metadata,
 750 inserting new entries for newly visited states if needed, as well as removing entries that were last
 751 pushed out of the Replay Buffer due to new transition insertions.

752 Therefore, using this dictionary that contains all relevant metadata about the agent’s experiences in
 753 an easily traversable format, the frontier is generated on the fly. By iterating once over it, filtering
 754 out all entries with a familiarity above the specified threshold, and inserting copies of the remaining
 755 to a new object. Thus, owing to its “lazy” evaluation (applying filters while traversing long

756 pre-populated lists) SIERL’s frontier is obtained with as little computation and memory usage as
 757 possible. The filtering algorithm is shown in pseudo-code in algorithm 5.
 758

759 C BASELINES 760

761 To evaluate SIERL’s performance, we compared it’s performance against four baseline methods
 762 with different exploration strategies. All baselines, like SIERL, are built upon a goal-conditioned
 763 framework, meaning the agent’s policy takes both the current state and a desired goal as input. This
 764 shared structure allows us to isolate the impact of different exploration and goal-selection strategies.
 765

766 The first baseline is a Q-learning agent with ε -Greedy exploration, which is a well-established
 767 method for discrete state-action spaces. The second baseline is Q-learning augmented HER. HER
 768 addresses the sparse reward problem in goal-conditioned RL by re-purposing failed episodes by “re-
 769 labeling” their transitions. When an agent fails to reach its intended goal, HER modifies a portion of
 770 the transitions from that episode, but with their goal set to the state that the agent actually reached.
 771 This turns both phases in a failed episode into a successful trajectory from a different perspective,
 772 providing a less sparse reward signal and making learning more efficient.
 773

774 The third baseline extends the Q-learning agent by adding a *novelty-based exploration bonus*. ε -
 775 Greedy exploration is still used, but its goal-sampling strategy is guided by *visitation counts*. During
 776 training, the agent keeps track of how many times it has visited each state. It then augments
 777 the reward signal by adding a bonus inversely proportional to the visitations for the visited state.
 778 This intrinsic motivation encourages the agent to explore new and under-explored regions of the
 779 environment, a strategy present in many state-of-the-art exploration methods. It is expected that
 780 after a long enough exploration, the bonuses diminish and the value function converges to the true
 781 values (Tang et al.). This method was selected as a representative of the intrinsic reward family of
 782 methods. All such methods (pseudo-counts, intrinsic curiosity modules, random network distilla-
 783 tion error) define a reward bonus that is high if the current state is different from the previous states
 784 visited by the agent, and low if it is similar (Henaff et al.). These methods approximate the novelty
 785 each to a different degree. Using counts (when possible) provides the most precise way to quantify
 786 and reward novelty, compared to e.g. approximating surprise with RND. For this reason, and given
 787 that count-based novelty bonuses has shown good performance in discrete state-action settings, we
 788 used this method as an upper bound for all of them.
 789

790 Our fourth baseline is another variation of Q-learning that uses the same setup but is trained ex-
 791clusively on *random goals*. Unlike our other baselines and SIERL, this agent does not have a fixed
 792 “main goal” throughout the training. It samples all training goals uniformly at random from the state
 793 space. This serves as a critical benchmarking study, as it helps us understand the importance of goal-
 794 centric exploration. By comparing SIERL to this baseline, we can quantify the benefit of a method
 795 that deliberately focuses on gradually discovering and achieving specific, potentially difficult, goals
 796 as opposed to just exploring the entire state space uniformly at random.
 797

798 D ADDITIONAL RESULTS 799

800 In addition to the main experiments presented in the main body of this work, we have conducted
 801 additional experiments such as sensitivity studies on SIERL’s hyperparameters, running with prob-
 802 abilistic transitions, and additional harder environments with a maze-like 9 room arrangement. Our
 803 findings are presented in this appendix.
 804

805 D.1 SENSITIVITY STUDIES 806

807 As can be seen from the results in figures 4, 5, 6, and 7. SIERL demonstrated robust performance.
 808 Our results indicate that adjusting the weights, w_n, w_c, w_g , has a negligible impact on outcomes.
 809 However, the algorithm is moderately sensitive to the softmax temperature (specifically at larger
 810 values) and more sensitive to the familiarity threshold.
 811

812 As expected, performance diminishes with a higher softmax temperature; this results in a “softer”
 813 distribution and near-uniform random prioritization of sub-goals, which nullifies the benefits of this
 814 method. Regarding the familiarity threshold: an excessively high value slows frontier expansion and
 815

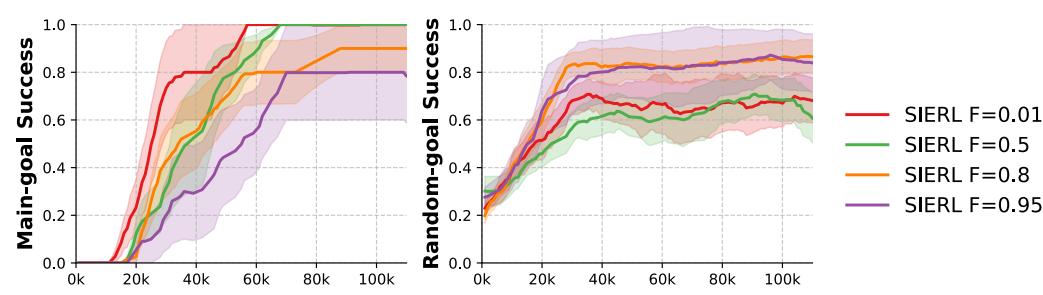


Figure 4: Success rate for the main-goal and random-goals in 4-step long Hallway for SIERL with varying familiarity thresholds, F_π^{thr} .

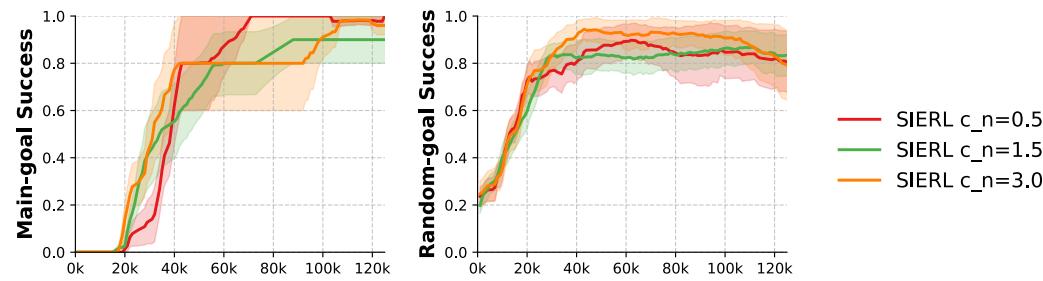
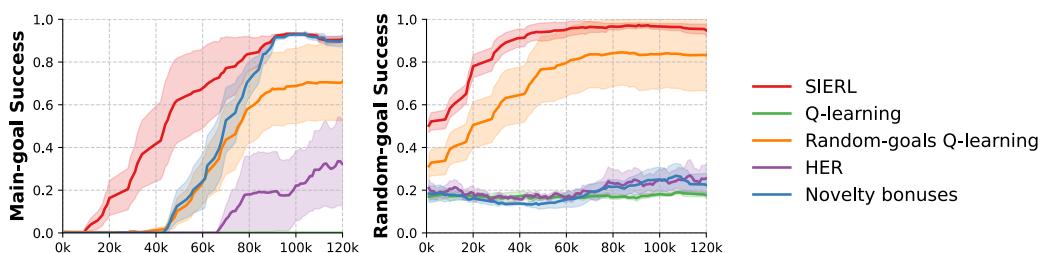
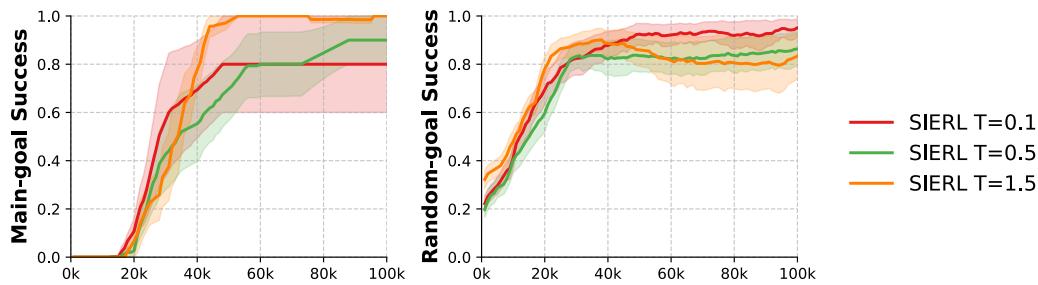
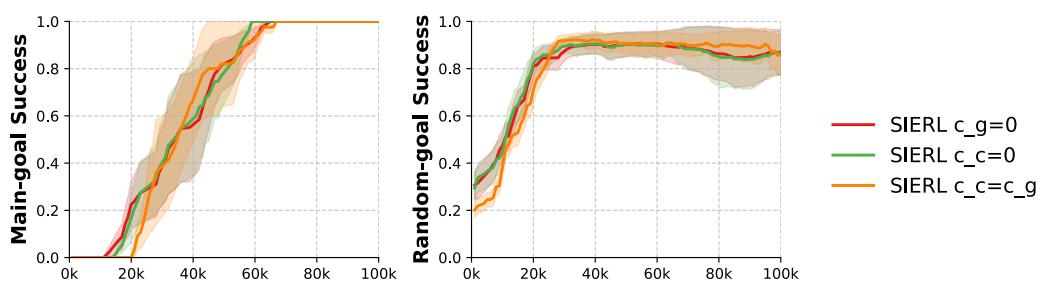


Figure 5: Success rate for the main-goal and random-goals in 4-step long Hallway for SIERL with varying novelty weights, w_n .

solution-trajectory discovery, though it maintains a stable random-goal success rate. Conversely, a lower threshold promotes aggressive novelty-seeking at the expense of “practicing” in familiar regions. This significantly reduces the random-goal success rate—while leaving main-goal success largely unaffected—unless the environment is sufficiently difficult that such aggressive expansion disrupts the learning process entirely.”

D.2 PROBABILISTIC TRANSITIONS

In order to evaluate SIERL’s performance outside of deterministic settings, we conducted an additional trial with probabilistic transitions. In particular, we used “slippery actions” such as those used in other discrete state-action settings in the literature. In this case, when the agent chooses an action, there is a small probability of executing another adjacent one. E.g., choosing to take the action “up” will result in the action taken being up with 80% chance, while there is a 10% chance of it being “right”, and 10% chance of it being “left”. The resulting performance of SIERL and all original baselines can be seen in Figure 8. It can be observed that SIERL is outperforming all baselines here as well. When compared to Figure 2, all baselines’ performance is observed to be more notably impacted by the noisy actions than SIERL’s. In main-goal performance SIERL is now performing better than novelty bonuses, the leading baseline, while still achieving random-goal success-rate above that of Random-goals Q-learning. Other than a slight decrease in main-goal success rate, no other significant impact is noticeable.



918 D.3 THE MULTIROOM AND DOORKEY ENVIRONMENTS
919

920 We conducted additional experiments on more challenging environments, including the nine-room
921 variants of MultiRoom and DoorKey, as shown in Figure 9. These domains feature larger spatial
922 structure, longer horizons, and more complex navigation dependencies, providing a stronger test of
923 scalable exploration and generalization.

924 The results in Figure 10 show that, in terms of main-goal performance, SIERL matches the strongest
925 novelty-bonus baselines, which explore aggressively in these settings. Importantly, SIERL achieves
926 substantially better random-goal generalization, demonstrating that the frontier-based curriculum
927 supports broader mastery of the environment rather than focusing solely on the shortest path to the
928 main goal.

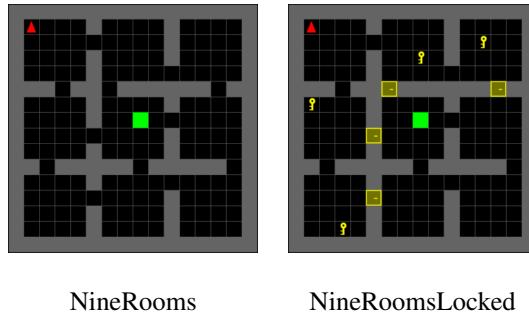
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

Figure 9: The nine rooms variants of MultiRoom and DoorKey.

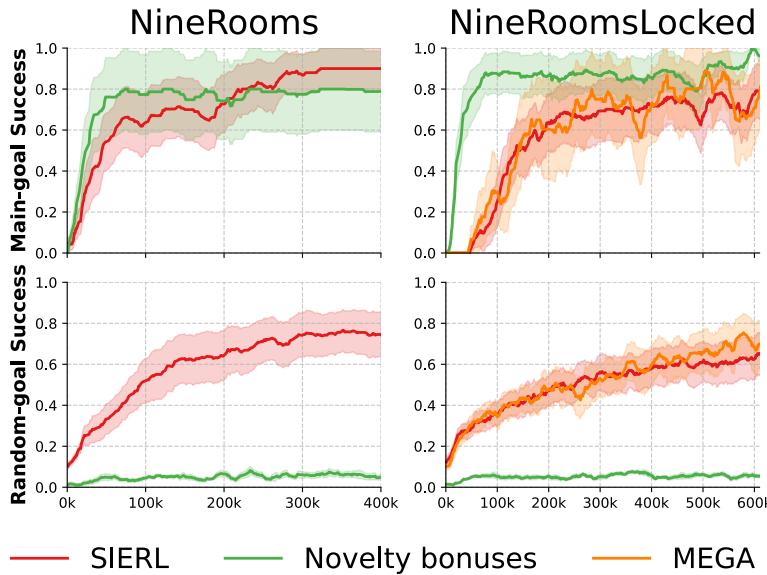


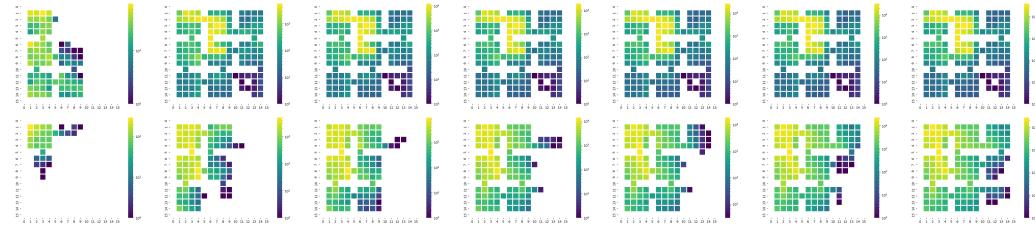
Figure 10: Success rate for the main-goal and random-goals in nine rooms environment from MultiRoom and DoorKey for SIERL and Novelty Bonuses.

966 D.4 COVERAGE
967

968 We additionally provide a coverage analysis to visualize how exploration progresses over time. Fig-
969 ure 11 shows state visitation counts in NineRoomsLocked for the novelty-bonus baseline (top) and
970 for SIERL (bottom) at 2k, 100k, 200k, 300k, 400k, 500k, and 600k environment steps. Visitation
971 frequencies are shown on a logarithmic scale. The agent starts in the top-left room, and the main
goal is located in the center room.

972 Under SIERL, coverage expands outward in a structured manner. After the agent first reaches the
 973 main goal (around 100k steps), the frontier continues to grow, and regions farther from both the ini-
 974 tial position and the goal receive progressively fewer visits. This reflects the incremental broadening
 975 of mastered states induced by the frontier-based curriculum.

976 In contrast, the novelty-bonus baseline explores aggressively early on but rapidly collapses toward
 977 the shortest path between the start and the goal, leading to limited coverage of alternative routes and
 978 peripheral states.



980
 981
 982
 983
 984
 985
 986
 987
 988 Figure 11: State visitations in NineRoomsLocked for Novelty bonuses (top) and SIERL (bottom)
 989 throughout training at 2k, 100k, 200k, 300k, 400k, 500k, and 600k steps. Agent starts in
 990 the top left corner and the goal is in the center of the central room.

991 992 D.5 COMPARISON WITH MEGA EXPLORATION

993 MEGA (Maximum Entropy Gain Exploration) (Pitis et al.) is an exploration strategy that defines
 994 exploration problem as distribution matching of achieved and encourages the agent to systematically
 995 expand its knowledge by maximizing the entropy of its achieved goal distribution. Instead of relying
 996 on random exploration or external rewards, the agent deliberately sets goals from low density regions
 997 of achieved goal distribution. Instead of directly setting random goals that might not be reachable,
 998 by successfully returning to these states and then exploring locally, MEGA allows to continually
 999 increase entropy of achieved goal distribution, creating an automatic curriculum that bridges the gap
 1000 between the starting state and distant, difficult task goals.

1001 MEGA’s desired behavior is fundamentally close to that of SIERL, albeit with a different sub-goal
 1002 selection strategy. In MEGA, The agent looks at all the goals it has successfully reached in the past
 1003 (stored in the replay buffer) and estimates how “crowded” or common each goal is using a density
 1004 model (i.e., a Kernel Density Estimator). Thus, MEGA’s frontier sampling involves selecting the
 1005 “most novel” goals with the lowest density. In addition, OMEGA extends MEGA by blending its
 1006 goal setting using uniformly random-goals, as MEGA’s original goal distribution matching become
 1007 more tractable.

1008 Since MEGA relies on a KL divergence estimator which becomes increasingly demanding as the
 1009 number of experiences increases, it is a more computationally intensive method. However, aside
 1010 from the specific density estimation used, MEGA can be effectively implemented as a special case of
 1011 SIERL’s by adjusting hyperparameters accordingly. By setting the familiarity threshold to $F_\pi^{\text{thr}} = 1$, in
 1012 order to sample sub-goals from the whole replay buffer, the novelty exponent $w_n = -1$ to prioritize
 1013 novelty instead, and the path weights to $w_c = w_g = 0$ to make the selection agnostic to path cost
 1014 estimates, we obtain an algorithm that closely emulates MEGA. To compare it with
 1015 SIERL, we evaluated its performance on the DoorKey environment in Figure 10. Its performance
 1016 matches closely that of SIERL with a slightly higher noise in main-goal success.

1017
 1018
 1019
 1020
 1021
 1022
 1023
 1024
 1025