BIDIRECTIONAL VORONOI-BIASED EXPLORATION CURRICULUM FOR REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

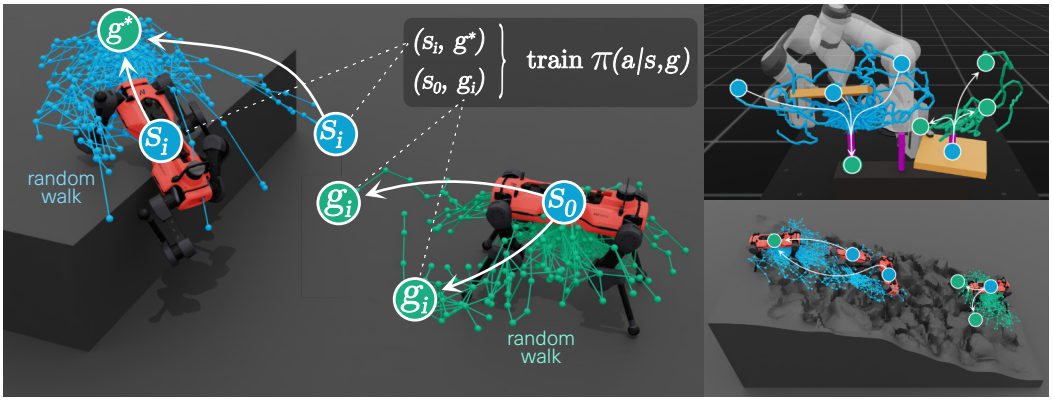


Figure 1: **BVER grows the curriculum from both ends of the task.** Intermediate starts s_i (blue) grow from the target goal $g^\star$ and intermediate goals g_i (green) from the initial distribution $s_0 \sim \rho_0$; one policy trains on both (white arrows) with sparse rewards and no demonstrations. Shown: *box climbing* (left), *ring-on-peg transfer* (top right), and a scanned terrain (bottom right).

ABSTRACT

Long-horizon tasks with sparse rewards pose an exploration bottleneck for goal-conditioned reinforcement learning: a policy started from the initial state rarely reaches the goal and receives no learning signal. Reference motions, hand-designed curricula, and shaped rewards supply this signal but require demonstrations or task-specific engineering; automatic start-state and goal curricula avoid this but typically expand from one side only, so the full distance to the target must be covered from that side. We propose the Bidirectional Voronoi-biased Exploration curriculum for Reinforcement learning (BVER), which expands from both ends at once. Inspired by bidirectional RRT planning, BVER grows start states outward from the goal and goals outward from the initial state distribution, biases both toward unexplored task space, and steers them toward each other, training one goal-conditioned policy on both. On point-mass mazes, quadrupedal box climbing, and robot-arm ring-on-peg transfer, BVER learns faster than all compared reference-free curricula. On box climbing, it reaches 95% success on a $0.4\,\mathrm{m}$ box in roughly 65% fewer iterations than the best of them, is the only one of them to learn to climb a $0.7\,\mathrm{m}$ box, and yields a policy robust to start, goal, and yaw variation. Without a demonstration, it approaches the sample efficiency of reference-based curricula on the $0.4\,\mathrm{m}$ box and on ring-on-peg transfer. Ablations show that expanding from both ends outperforms either direction alone. Project website, videos, and code: `https://bver-rl.github.io/`

1 INTRODUCTION

Many robotic tasks are naturally posed as reaching a goal: a quadruped that should climb onto a box is rewarded for standing on top of it, not for how it gets there. With such a sparse reward and a long horizon, a policy started from the initial state distribution rarely reaches the goal and thus receives

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

no learning signal (Florensa et al., 2017): policy gradient methods such as PPO (Schulman et al., 2017) learn from the returns of the trajectories the policy currently generates, and a reward that none of them collects cannot guide it.

In robot control, the missing signal is typically supplied by terrain curricula with dense rewards (Rudin et al., 2022; Lee et al., 2020), reference motions (Fuchioka et al., 2023), hand-designed curricula (Atanassov et al., 2025), or curiosity rewards (Schwarke et al., 2023). These have enabled highly capable controllers but require a demonstration, task-specific engineering, or reward terms beyond the task objective.

Automatic curriculum learning replaces this design effort with intermediate tasks generated during training (Narvekar et al., 2020; Portelas et al., 2020b). Start-state curricula begin closer to the goal, with start states taken from a reference trajectory (Peng et al., 2018; Resnick et al., 2018) or from random walks outward from the goal (Florensa et al., 2017), and goal curricula keep the start fixed and propose intermediate goals (Florensa et al., 2018; Ren et al., 2019; Cho et al., 2023). Both expand from one end only: the frontier must cover the entire distance to the fixed end, and the policy trains on the target task from that side only.

Two mechanisms from sampling-based motion planning address this limitation. Rapidly-exploring random trees (RRT; LaValle & Kuffner, 2001) extend the vertex nearest to a uniform sample of the space, so growth is biased toward large unexplored regions (the Voronoi bias), and RRT-Connect (Kuffner & LaValle, 2000) grows one tree from the start and one from the goal and extends each toward the other until they meet. BVER (Bidirectional Voronoi-biased Exploration curriculum for Reinforcement learning) transfers both to curriculum generation. It grows start states outward from the goal by random walks (the backwards expansion) and goals outward from the initial state distribution in the same way (the forwards expansion), connects the two sets, and trains a single goal-conditioned policy on both. Like reverse curriculum generation (RC; Florensa et al., 2017), BVER keeps only tasks of intermediate difficulty for the current policy, while its expansion covers the task space (Figure 2).

Our contributions are:

- A bidirectional start–goal curriculum for sparse-reward, long-horizon goal-conditioned RL that generates both intermediate starts and goals, without reference motions, shaped rewards, or a hand-designed curriculum.
- An expansion mechanism that carries the Voronoi bias of RRT and the connect step of RRT-Connect over to curriculum generation to propose feasible states as tasks.
- Experiments showing that BVER learns fastest among the reference-free curricula on all six compared tasks, is the only one of them to learn to climb the $0.7\,\mathrm{m}$ box, transfers to scanned terrains with the same hyperparameters, and outperforms either expansion direction alone.

2 Related work

Automatic curriculum learning: Automatic curriculum learning (ACL) adapts the training tasks to the learner (Narvekar et al., 2020; Portelas et al., 2020b). Environment curricula sample generator parameters by learning progress (Portelas et al., 2020a), replay levels by value loss (Jiang et al., 2021), generate regret-maximizing levels (Dennis et al., 2020; Parker-Holder et al., 2022), or widen domain randomization as performance improves (Akkaya et al., 2019); reward curricula add intrinsic rewards such as RND (Burda et al., 2019), whose authors highlight global exploration as an important direction. BVER is complementary, changing only where episodes start and which goal they pursue.

Start-state curricula: Reverse curriculum generation (RC; Florensa et al., 2017) grows start states outward from the goal by random-action rollouts and keeps those of intermediate difficulty, using resets but no demonstrations; BaRC (Ivanovic et al., 2019) uses backward reachable sets of an approximate model instead. Demonstration-based methods reset to states of a reference trajectory (Salimans & Chen, 2018; Nair et al., 2018), sampled uniformly (RSI; Peng et al., 2018), moved backward on a schedule (Backplay; Resnick et al., 2018), or prioritized by success rate (RFCL; Tao et al., 2024). Go-Explore (Ecoffet et al., 2021) and PBCS (Matheron et al., 2020) return to rarely visited states from an archive and explore from there. Similarly, BVER seeds its random walks at stored states.

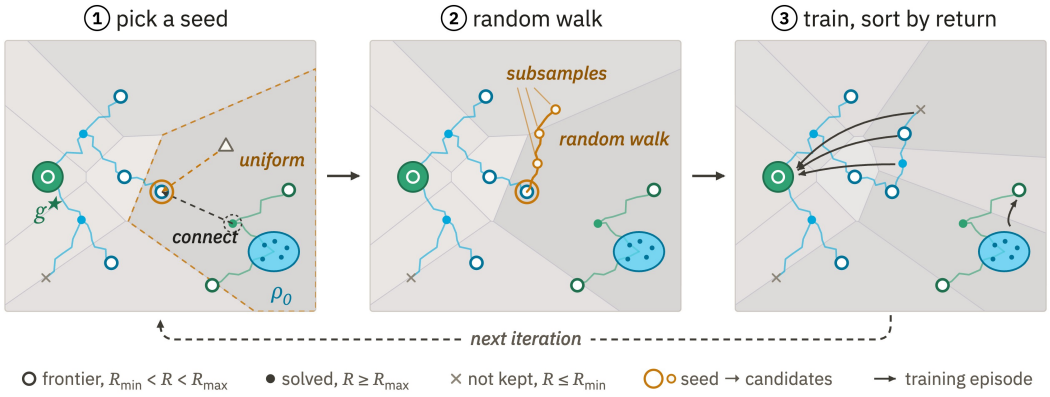


Figure 2: BVER's expansion mechanism. Intermediate starts s_i (blue) grow from the target goal $g^\star$ and goals g_i (green) from the initial distribution ρ_0. Random walks start from the states nearest to uniform task-space samples, favoring states whose Voronoi cell (shaded) reaches into unexplored space, or nearest to the other direction's solved states to connect the two (dashed).

Goal curricula: Goal curricula keep the initial state distribution and propose goals: GoalGAN (Florensa et al., 2018) generates goals of intermediate difficulty; HGG (Ren et al., 2019) and OUTPACE (Cho et al., 2023) select achieved states by progress toward the target, and DISCOVER (Diaz-Bone et al., 2025) by estimated achievability, novelty, and relevance; Skew-Fit (Pong et al., 2020) and MEGA (Pitis et al., 2020) select achieved states of low density; and C-Planning (Zhang et al., 2022) commands waypoints between start and goal under a learned distance. VaPRL (Sharma et al., 2021) and IBC (Kim et al., 2023) build such curricula from replay-buffer states and, unlike BVER, need no resets. BVER builds on both lines of work, growing start states and goals toward each other and seeding them by Voronoi bias instead of visit counts or density.

Sampling-based planning: A rapidly-exploring random tree (RRT) (LaValle & Kuffner, 2001) extends the vertex nearest to a random sample; vertices are thus selected in proportion to the area of their Voronoi regions, which biases the tree toward unexplored space (Kuffner & LaValle, 2000). RRT-Connect (Kuffner & LaValle, 2000) grows trees from start and goal and connects them. R3L (Morere et al., 2020) initializes a policy from RRT solutions, and SoRB (Eysenbach et al., 2019) searches a graph over replay-buffer states with a learned value function. Florensa et al. (2017) proposed combining start-state and goal generation as future work, citing RRT-Connect as the analogue; BVER follows this suggestion, growing a curriculum rather than a path.

Curricula and references in robot learning: Legged robots are mostly trained with dense rewards and terrain curricula (Rudin et al., 2022; Lee et al., 2020), with an inner-product reward for parkour (Cheng et al., 2024), or with RSI on a trajectory-optimization reference (Fuchioka et al., 2023). Hand-designed curricula start jumps at random heights and upward velocities (Atanassov et al., 2025) or grow obstacle dimensions and navigation distances with performance (Hoeller et al., 2024), and Schwarke et al. (2023) extend RND with a curiosity state for sparse-reward loco-manipulation. These methods yield highly capable controllers; BVER targets settings without such guidance.

3 Preliminaries

3.1 Goal-Conditioned Markov Decision Process

We model the task as a goal-augmented Markov decision process (GA-MDP) (Liu et al., 2022) with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel P, discount factor γ, initial state distribution ρ_0, and goal space $\mathcal{G}$. A known mapping $\phi : \mathcal{S} \to \mathcal{G}$ projects a state onto the goal it achieves (Andrychowicz et al., 2017), and a goal g is reached in state s if $\|\phi(s) - g\| \leq \epsilon$ for a tolerance ϵ. The reward is

sparse and binary: the agent receives $+1$ in every step at which the goal is reached and 0 otherwise. Episodes have a fixed horizon H and do not terminate when the goal is reached.

The agent acts with a goal-conditioned policy $\pi_\theta(a \mid s, g)$ (Schaul et al., 2015). Since BVER only chooses the start–goal pairs of the rollouts, it is not tied to a particular learner; we use PPO (Schulman et al., 2017), which scales to thousands of parallel simulated environments (Rudin et al., 2022), rather than off-policy learning with hindsight relabeling (Andrychowicz et al., 2017). We measure a rollout $\tau = (s_0, \dots, s_{H-1})$ under goal g by its undiscounted return, normalized by the maximum attainable return H, i.e., the fraction of the horizon spent at the goal:

$$R(\tau, g) = \frac{1}{H} \sum_{t=0}^{H-1} \mathbf{1}\big[\|\phi(s_t) - g\| \le \epsilon\big] \in [0, 1]. \tag{1}$$

Since the reward accrues in every step at the goal, R reflects how quickly the policy reaches the goal and how reliably it stays there, rather than binary success. The *target task* is to reach a single target goal $g^\star \in \mathcal{G}$ from the initial state distribution ρ_0, and the learning objective is

$$J(\pi) = \mathbb{E}_{s_0 \sim \rho_0,\, \tau \sim \pi(\cdot \mid \cdot,\, g^\star)}\big[R(\tau, g^\star)\big]. \tag{2}$$

Following Florensa et al. (2017), we assume that the simulator can be reset to arbitrary states and simulated from them under random actions, that ρ_0 can be sampled, that a goal state $s^g \in \mathcal{S}$ with $\phi(s^g) = g^\star$ is known, and that a bounded task space $\mathcal{T}$ containing the target task is known and can be sampled uniformly.

3.2 Curriculum

Curriculum learning trains on a sequence of tasks of increasing difficulty (Narvekar et al., 2020; Portelas et al., 2020b). In our setting, a task is a start–goal pair, and a *curriculum* decides in every training iteration which pairs the policy is trained on, based on the experience collected with the current policy (Tao et al., 2024); the intermediate tasks are auxiliary, and the objective remains the target task. Formally, in iteration k the curriculum replaces the start–goal distribution $\rho_0 \times \delta_{g^\star}$ of Equation 2 by a distribution q_k over start–goal pairs, while the transition kernel P and the reward, and thus R, remain unchanged:

$$J_k(\pi) = \mathbb{E}_{(s_0, g) \sim q_k,\, \tau \sim \pi(\cdot \mid \cdot,\, g)}\big[R(\tau, g)\big]. \tag{3}$$

A *start-state curriculum* keeps the goal fixed at $g^\star$ and proposes intermediate start states, $q_k = \sigma_k \times \delta_{g^\star}$ (Florensa et al., 2017; Tao et al., 2024), whereas a *goal curriculum* keeps the start states drawn from ρ_0 and proposes intermediate goals, $q_k = \rho_0 \times \mu_k$ (Florensa et al., 2018; Ren et al., 2019). BVER does both at once with one shared goal-conditioned policy (Section 4), mixing the two with a ratio $\alpha \in [0, 1]$:

$$q_k = \alpha\, \sigma_k \times \delta_{g^\star} + (1 - \alpha)\, \rho_0 \times \mu_k. \tag{4}$$

4 Method

BVER maintains finite sets of states from which the starts and goals of each training iteration are sampled uniformly at fixed ratios. We first describe the backwards expansion, which extends RC's random walks with task space coverage, then apply it to the forwards expansion and combine the two. Algorithm 1 summarizes the method, with its subroutines in Appendix A.1; Figure 7 in the appendix gives an overview.

4.1 Voronoi-Biased Expansion

Exploration: As in RC (Florensa et al., 2017), training starts from valid states obtained by simulating random actions from the goal state (blue in Figure 2). After each training iteration, we sort these states s_i by the normalized return (Equation 1) of their rollouts τ_i into frontier and solved states,

$$F = \{s_i : R_{\min} < R(\tau_i, g^\star) < R_{\max}\}, \qquad S = \{s_i : R(\tau_i, g^\star) \ge R_{\max}\}, \tag{5}$$

and discard states with $R \le R_{\min}$ as too hard for the current policy. Mixing the two buffers at a fixed ratio yields the *mixed* training set, which thus holds mostly states of intermediate difficulty.

Algorithm 1 BVER (subroutines in Appendix A.1)

Require: initial distribution ρ_0, target goal $g^\star$ with goal state s^g, task space $\mathcal{T}$, goal-conditioned policy π, environment split $N_{\text{explore}}, N_{\text{exploit}}$

$\mathcal{I}^{\text{fwd}} \leftarrow \{s_0 \sim \rho_0\}$, $\mathcal{I}^{\text{bwd}} \leftarrow \{s^g\}$; $C^d, F^d, S^d \leftarrow \emptyset$ for $d \in \{\text{fwd}, \text{bwd}\}$
for each iteration **do**
  for $(d, \bar{d}) \in \{(\text{fwd}, \text{bwd}), (\text{bwd}, \text{fwd})\}$ **do** ▷ environments split according to Equation 4
    $C^d \leftarrow C^d \cup \textsc{RandomWalk}(\mathcal{I}^d, \mathcal{T}, S^{\bar{d}})$ ▷ seeds from Equation 6
    $D^d \leftarrow \textsc{Rollout}(\pi, C^d, d, N_{\text{explore}}/2)$ ▷ explore: $(s_0 \sim \rho_0, \phi(\cdot))$ if fwd, $(\cdot, g^\star)$ if bwd
    $D^d \leftarrow D^d \cup \textsc{Rollout}(\pi, \mathcal{I}^d, d, N_{\text{exploit}}/2)$ ▷ exploit
    $F^d, S^d \leftarrow \textsc{Threshold}(D^d, F^d, S^d, d)$ ▷ append; threshold according to Equation 5
    $\mathcal{I}^d \leftarrow \textsc{Mix}(F^d, S^d)$ ▷ replay a fraction ν of solved states
  $\pi \leftarrow \textsc{Train}(\pi, D^{\text{fwd}} \cup D^{\text{bwd}})$ ▷ one shared policy update

To choose the seeds of the next random walks, we add a criterion that promotes coverage of the task space and thus drives the frontier toward the initial distribution: we (1) sample points uniformly from the task space $\mathcal{T}$, (2) find for each sample its nearest neighbor in the mixed training set under the projection ϕ, and (3) use these neighbors as seeds. A state is thereby selected with probability proportional to the volume of its Voronoi cell in $\mathcal{T}$ (shaded in Figure 2), which is largest for states at the edge of the explored region, whose cells reach far into unexplored space. As with the Voronoi bias of RRT (LaValle & Kuffner, 2001), this steers the expansion outward instead of refining regions the curriculum already covers. Each walk then simulates T_B random actions from its seed, and N_s states subsampled along it become candidates for the next rollouts.

Exploitation: In parallel, N_{exploit} of the parallel environments continue training on the mixed set, while N_{explore} train on the new random-walk states. Training on the reward frontier improves the policy, and replaying solved states mitigates catastrophic forgetting. This step is optional: on tasks without forgetting, we omit it, and then the mixed set only seeds the random walks (Appendix A.1). Both steps repeat until the policy solves the task from the initial distribution.

4.2 Bidirectional Expansion

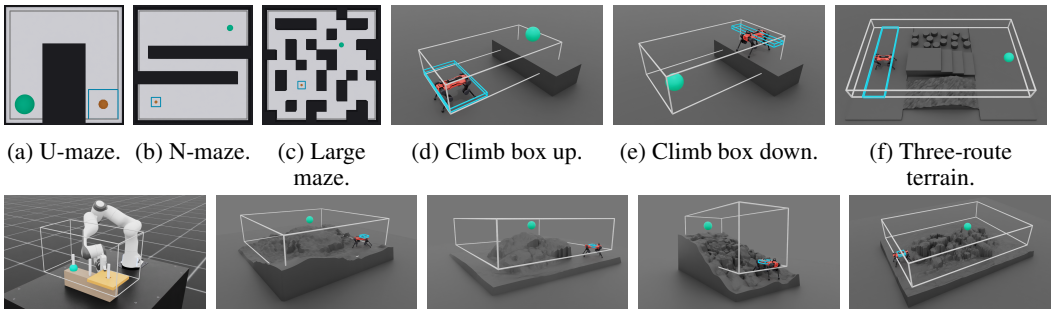
Forwards expansion: The forwards expansion applies the same mechanism with the roles of start and goal exchanged: the policy is trained on pairs $(s_0 \sim \rho_0,\, g_i)$ with goals g_i from the forwards frontier, rather than $(s_i,\, g^\star)$. The random walks are then seeded at states that can be reached, rather than states from which the goal can be reached. Because the walks must be reset to full simulator states, the forwards buffers store the final state of each rollout, sorted by the return of its commanded goal; a rollout that reached and held its goal ends within ϵ of it (Appendix A.1).

Bidirectional expansion: Combining the two, we train one shared goal-conditioned policy on both $(s_0 \sim \rho_0,\, g_i)$ and $(s_i,\, g^\star)$ (arrows in Figure 2), with the training budget split evenly between the two expansions ($\alpha = \frac{1}{2}$ in Equation 4). The curriculum thus covers the reachable task space from two directions, and the policy learns to reach the target goal from a wide range of start states and to reach a wide range of goals from the initial distribution.

Connecting the two frontiers: As in RRT-Connect (Kuffner & LaValle, 2000), which grows two trees toward each other until they connect, we steer the two frontiers together by replacing the samples from $\mathcal{T}$ in step (1) with samples from the other direction's solved set (dashed line in Figure 2). When expanding the forwards frontier, for example, we sample start states from which the policy already reaches $g^\star$ and seed the walks at their nearest neighbors in the forwards mixed training set, so the intermediate goals grow toward $g^\star$; symmetrically, the intermediate starts grow toward ρ_0. Each expansion thus moves into the region the other has already made solvable instead of spreading uniformly over $\mathcal{T}$. To balance coverage against connection, we draw each sample from the mixture

$$x_{\text{rand}} \sim p_{\text{connect}}\,\mathcal{U}\Big(\phi\Big(S^{\bar{d}}\Big)\Big) + (1 - p_{\text{connect}})\,\mathcal{U}(\mathcal{T}), \tag{6}$$

where $S^{\bar{d}}$ is the other direction's solved set and p_{connect} the connect ratio.



(a) U-maze. (b) N-maze. (c) Large maze. (d) Climb box up. (e) Climb box down. (f) Three-route terrain.

(g) Ring-on-peg transfer. (h) Freestanding boulder I. (i) Freestanding boulder II. (j) Rocky ascent. (k) Small stone field.

Figure 3: Task overview: three *point-mass mazes* (a–c), quadrupedal *box climbing* (d, e), the *three-route terrain* (f), *ring-on-peg transfer* (g), and four scanned terrains (h–k). Cyan indicates the initial distribution, green the target goal, and the white box represents the task space.

5 Experiments

We study BVER's sample efficiency and robustness compared to existing curricula (Section 5.1), whether its training covers the whole task space (Section 5.2), how it transfers to terrains scanned from real environments (Section 5.3), and the contribution of its expansion directions and connect step (Section 5.4). All experiments are implemented in Isaac Lab (NVIDIA et al., 2025) and trained with PPO (Schulman et al., 2017); Figure 3 shows the simulated tasks. Unless stated otherwise, the reward is sparse, without shaping terms. The task space $\mathcal{T}$ is a box in position space, i.e., $\phi(s)$ projects the point-mass, base, or ring position to (x, y) or (x, y, z), which makes goals easy to specify and gives a Euclidean metric for the nearest-neighbor search. Observation and action spaces, rewards, and hyperparameters are detailed in Appendix A.2. Learning curves show the mean success rate over five seeds (standard error shaded) against the training iteration; all methods use the same number of parallel environments and simulation steps per iteration. The demonstrations for the reference-based baselines are rollouts of policies trained with BVER.

5.1 Sample Efficiency and Robustness across Domains

We first test whether BVER improves sample efficiency and robustness over curriculum baselines, and whether these gains hold across embodiments. In the *point-mass mazes* (Figures 3a–3c), similar to those used in prior work (Florensa et al., 2017; Cho et al., 2023), a velocity-controlled point mass must find a path from the initial distribution to the goal. In *box climbing* (Figures 3d and 3e), the quadruped must reach a goal on top of a box from an initial distribution in front of it or, in the symmetric climb-down task, a goal on the ground from the top of the box. In *ring-on-peg transfer* (Figure 3g), a Franka Panda arm holding a rigidly attached ring must move it from the bottom of one peg to the bottom of another, a reaching task constrained by the pegs. On all tasks, we compare against PPO without a curriculum (*vanilla PPO*), a simple curriculum that samples intermediate starts and goals uniformly from the free task space (*random curriculum*), and the reference-free reverse curriculum (RC) (Florensa et al., 2017). On *ring-on-peg transfer* and *box climbing*, we additionally compare against two reference-based methods that require a demonstration (marked with $\star$): reference state initialization (RSI) (Peng et al., 2018) and Backplay (Resnick et al., 2018).

Sample Efficiency: Figure 4 shows the success rate over training. Vanilla PPO succeeds only for a few seeds in the simplest *U-maze* and never on any other task. Initializing the agent closer to the goal makes the reward attainable, but not on every task: the random curriculum plateaus at about 40% success in the *N-maze*, likely because few of its uniform samples fall into the narrow passages; RC stays below about 35% in the *N-maze*, on *ring-on-peg transfer*, and on the $0.4\,\mathrm{m}$ box, as does BVER's backwards expansion alone on the first two (Table 1); and neither succeeds on the $0.7\,\mathrm{m}$ box. We attribute RC's low success rate to the lack of an explicit incentive to expand toward ρ_0: on *ring-on-peg transfer*, for example, we observed that its start states spread into the free space above the pegs rather than toward the starting peg.

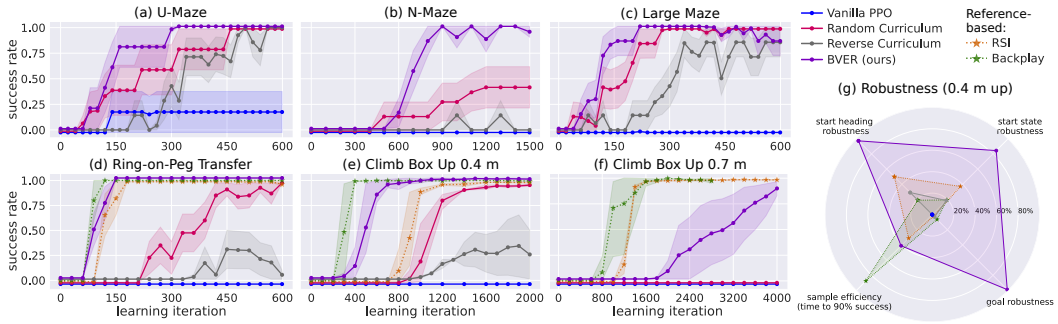


Figure 4: Sample efficiency across domains (a–f; mean over five seeds, standard error shaded) and robustness on *box climbing* (g). BVER learns fastest among the reference-free curricula on all tasks, and on *ring-on-peg transfer* and the $0.4\,\mathrm{m}$ box even faster than the reference-based RSI (marked with $\star$). Panel (g) uses the same colors as (a–f).

BVER learns fastest among the reference-free curricula on all tasks. On *ring-on-peg transfer*, it exceeds 95% success in about 150 iterations, four times fewer than the random curriculum, and lies between the two reference-based methods. Reference-based methods are expected to learn fastest, since the demonstration places their starts on a known solution: on the $0.4\,\mathrm{m}$ box (Figure 4(e)), Backplay converges fastest, and its advantage over RSI suggests that ordering the starts over training matters. BVER learns faster than RSI there and reaches 95% success in roughly 65% fewer training iterations than the best reference-free baseline. On the $0.7\,\mathrm{m}$ box (Figure 4(f)), BVER is the only reference-free curriculum that learns the task, reaching about 90% success after 4000 iterations, whereas the demonstration lets RSI and Backplay solve it in about 1600. Since this task requires two skills, walking and climbing, we hypothesize that the bidirectional expansion helps not only through spatial coverage but also by training both skills at the same time.

Robustness: We evaluate the policies trained on *box climbing* under (1) initial yaw perturbations of up to $\pm 90^\circ$, (2) start positions sampled uniformly from the task space (goal set to the target goal), and (3) goal positions sampled uniformly from the task space (initialized on the initial distribution). Figure 4(g) shows that BVER's policy maintains a high success rate under all three variations. RSI and Backplay train on states near the reference trajectory and thus see few states outside it, which likely explains their lower robustness. RC varies only the start states and always trains on the target goal, so its lower goal robustness is expected. The sample-efficiency axis (time to 90% success) is given relative to the fastest run over all methods and seeds (Backplay, 300 iterations).

5.2 Training Coverage

Since BVER biases its expansion toward unexplored task space, we expect it to train the policy on the whole terrain rather than only on the easiest way to the goal. To test this, we design a task with three qualitatively different obstacles between the initial distribution and the goal: stairs in the center, pillars on one side, and a ramp on the other (Figure 3f). The initial distribution spans the terrain's full width, so each obstacle is closest for roughly a third of the initial states. Only on this task do we add a dense shaping term (Ng et al., 1999) that rewards progress toward the goal, so that vanilla PPO finds a solution at all; both methods receive the same reward, and BVER's reward band still uses the sparse term only (Appendix A.2). We then roll out both policies from the initial distribution (one seed, 128 episodes each) and record which obstacle each successful episode traverses.

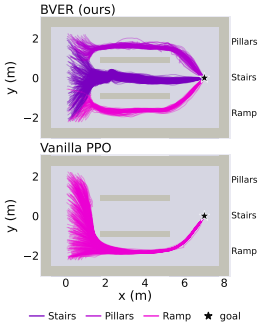


Figure 5: Successful trajectories of BVER and vanilla PPO on the *three-route terrain*.

Figure 5 shows that vanilla PPO only learns to traverse the ramp, the easiest of the three obstacles: regardless of the start, all successful episodes detour to it, even from in front of the stairs or pillars. In contrast, BVER's



(a) Iteration 150. (b) Iteration 500. (c) Iteration 1000. (d) Iteration 2000.

Figure 6: Samples of the explored regions on the *rocky ascent* terrain at different training iterations. Blue indicates the intermediate goals (forwards expansion), orange the intermediate starts (backwards expansion), cyan the initial distribution, and the black box represents the task space.

policy traverses all three obstacles (stairs 64%, pillars 21%, ramp 15% of successful episodes), consistent with a curriculum that covers the whole terrain and trains the skills each obstacle requires.

5.3 Scanned Terrains

Box climbing involves a single, simple obstacle. To test whether its hyperparameters transfer directly to more complex geometry, we evaluate BVER on four terrains scanned from real environments (Figures 3h–3k): two freestanding boulders to climb, a steep rocky slope, and a field of small stones, each requiring different locomotion skills. We reuse all PPO and BVER hyperparameters from *box climbing* (Section 5.1) without retuning; only the task space is chosen to roughly cover the terrain, the episode length ($10\,\mathrm{s}$ to $12\,\mathrm{s}$ instead of $4\,\mathrm{s}$) and the number of training iterations are adapted to the larger terrains, and the number of environments is halved for training speed at the same exploit-to-explore ratio (Table 3). On the *freestanding boulders* and the *small stone field*, the initial distribution is chosen such that no path to the goal is favored. At the end of training (3000 iterations, 5000 on the *rocky ascent*), BVER learns all four terrains, reaching a success rate (mean $\pm$ std. error over five seeds) of $91.0\pm1.7\%$ on *freestanding boulder I*, $84.4\pm2.1\%$ on *freestanding boulder II*, $77.2\pm2.2\%$ on the *rocky ascent*, and $59.8\pm4.4\%$ on the *small stone field*, which is the hardest of the four. Vanilla PPO, in contrast, does not succeed a single time on any of the four terrains, even with a higher entropy coefficient to encourage exploration. BVER's hyperparameters thus transfer across tasks within a domain.

Figure 6 shows the explored regions on the *rocky ascent* terrain over training. The intermediate starts and goals begin clustered around their roots and gradually spread over the reachable terrain as the policy improves. Although the task space (black box) is much larger than the reachable area, the random walks find a path to the goal without prior knowledge of the terrain, and the starts and goals cover the terrain rather than a single path, so the curriculum trains the policy on the whole terrain rather than only on the target task.

5.4 Ablation Studies

To isolate the contribution of BVER's components, we ablate the expansion direction on the three *point-mass mazes*, *ring-on-peg transfer*, and *box climbing*, and the connect step on the mazes.

Table 1: Ablation study: iterations until the mean success rate over five seeds reaches 75% (∞: not reached; n/a: not evaluated). PM-U/N/L: U-, N-, and large point-mass maze; RPT: ring-on-peg transfer; CU/CD: climbing up/down a box of the given height.

	Iterations to 75% Success Rate (↓)							
	PM-U	PM-N	PM-L	RPT	CU 0.4 m	CU 0.7 m	CD 0.4 m	CD 0.7 m
BVER (ours)	**160**	**800**	**140**	**120**	**600**	**3800**	**300**	**400**
BVER fwd. only	380	1100	440	∞	800	∞	400	600
BVER bwd. only	440	∞	400	∞	800	**3800**	1700	2800
BVER (no connect)	340	1000	200	n/a	n/a	n/a	n/a	n/a
BVER (greedy)	260	1500	240	n/a	n/a	n/a	n/a	n/a

Expansion direction: We compare BVER against forwards and backwards expansion alone (fwd. and bwd. only), each with the full environment budget, including *box climbing* up and down at $0.4\,\mathrm{m}$

and $0.7\,\mathrm{m}$. The bidirectional expansion reaches 75% success first on every task, tied only with the backwards expansion on climbing up the $0.7\,\mathrm{m}$ box, whereas each direction alone fails to reach 75% success on two of the eight tasks (Table 1). Which direction works depends on the task. On *box climbing*, the expansion that starts on top of the box, i.e., backwards when climbing up and forwards when climbing down, is clearly stronger, likely because the random walks are subject to gravity and thus traverse the box edge more easily downwards than upwards: at $0.7\,\mathrm{m}$, the expansion from the ground never reaches 75% success when climbing up and needs seven times more iterations than BVER when climbing down. On *ring-on-peg transfer*, neither direction alone reaches 75% success, as each expands from its peg into the free space in between but rarely reaches the other peg. In the mazes, BVER needs 1.4 to 2.9 times fewer iterations than the better unidirectional variant, and the backwards expansion alone fails in the *N-maze*. In the *large maze*, we observe that the unidirectional variants cover the space uniformly, whereas BVER focuses on the region between start and goal. Combining both directions thus removes the need to know in advance which direction suits a task. The learning curves are given in Appendix A.3.

Connecting the frontiers: On the three mazes, we compare the default connect ratio $p_{\text{connect}} = 0.5$ against seeding the random walks only from uniform task space samples ($p_{\text{connect}} = 0$, no connect) or only from the other direction's solved set ($p_{\text{connect}} = 1$, greedy). Both extremes slow down learning on all three mazes (Table 1). Without connecting, the two frontiers spread evenly over $\mathcal{T}$ and likely take longer to meet, requiring up to 2.1 times more iterations. Greedy connecting drops the coverage bias and requires up to 1.9 times more iterations, most notably in the *N-maze*, likely because its walls block the direct way toward the other frontier. Both coverage and connection thus contribute, and we use $p_{\text{connect}} = 0.5$ in all experiments (Table 3).

6 Conclusion

We propose BVER, a curriculum for sparse-reward, long-horizon goal-conditioned reinforcement learning that transfers bidirectional expansion from sampling-based planning (LaValle & Kuffner, 2001; Kuffner & LaValle, 2000) to curriculum generation: it grows start states from the goal and goals from the initial state distribution, biases both toward unexplored task space, and steers them toward each other. In simulation, this yields sample-efficient and robust policies in navigation, locomotion, and manipulation. The random walks run in the same parallelized simulator as the policy rollouts and increase the wall-clock training time by about 28% in our experiments.

BVER relies on a simulator that can be reset to arbitrary states, on random-action rollouts, and on a bounded task space with a meaningful distance metric. All task spaces in this work are two- or three-dimensional position boxes with a Euclidean metric; higher-dimensional or non-Euclidean task spaces remain untested and could use learned distances such as the dynamical distance of Hartikainen et al. (2020). Like RC (Florensa et al., 2017), BVER's backwards expansion further assumes approximately reversible dynamics: a random walk from the goal reaching a state does not imply that the goal is reachable from that state, as the gravity asymmetry in *box climbing* illustrates, and such states are dropped only after a rollout has shown them unsolvable ($R \leq R_{\min}$); learned dynamics models or rollouts of the current policy could replace the random walks. Finally, all results are in simulation, and hardware deployment is left to future work.

AI use statement

In this work, we used generative AI tools for creating scientific figures from non-AI-generated data, creating and editing software code, and editing the paper text to improve readability (proofreading of grammar and style). We have reviewed all AI-assisted work: generated figures were checked against the underlying data, AI-assisted code was verified and tested for correctness by the authors, and all suggested text edits were reviewed and accepted by the authors.

Reproducibility statement

Experiments can be reproduced with the code provided in the project website. The code includes all hyperparameters, random seeds, and instructions to run the experiments. We run all experiments on a single NVIDIA RTX 4090 GPU with five different random seeds each.

REFERENCES

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving Rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in Neural Information Processing Systems*, 30, 2017.

Vassil Atanassov, Jiatao Ding, Jens Kober, Ioannis Havoutis, and Cosimo Della Santina. Curriculum-based reinforcement learning for quadrupedal jumping: A reference-free design. *IEEE Robotics & Automation Magazine*, 32(2):35–48, 2025.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

Xuxin Cheng, Kexin Shi, Ananye Agarwal, and Deepak Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11443–11450. IEEE, 2024.

Daesol Cho, Seungjae Lee, and H. Jin Kim. Outcome-directed reinforcement learning by uncertainty & temporal distance-aware curriculum goal generation. In *International Conference on Learning Representations*, 2023.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in Neural Information Processing Systems*, 33:13049–13061, 2020.

Leander Diaz-Bone, Marco Bagatella, Jonas Hübotter, and Andreas Krause. DISCOVER: Automated curricula for sparse-reward reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. First return, then explore. *Nature*, 590(7847):580–586, 2021.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In *Conference on Robot Learning*, pp. 482–495. PMLR, 2017.

Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning*, pp. 1515–1528. PMLR, 2018.

Yuni Fuchioka, Zhaoming Xie, and Michiel Van de Panne. Opt-Mimic: Imitation of optimized trajectories for dynamic quadruped behaviors. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5092–5098. IEEE, 2023.

Kristian Hartikainen, Xinyang Geng, Tuomas Haarnoja, and Sergey Levine. Dynamical distance learning for semi-supervised and unsupervised skill discovery. In *International Conference on Learning Representations*, 2020.

David Hoeller, Nikita Rudin, Dhionis Sako, and Marco Hutter. ANYmal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024. doi: 10.1126/scirobotics.adi7566. URL https://www.science.org/doi/abs/10.1126/scirobotics.adi7566.

Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. BaRC: Backward reachability curriculum for robotic reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 15–21. IEEE, 2019.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pp. 4940–4950. PMLR, 2021.

Jigang Kim, Daesol Cho, and H Jin Kim. Demonstration-free autonomous reinforcement learning via implicit and bidirectional curriculum. In *International Conference on Machine Learning*, pp. 16441–16457. PMLR, 2023.

James J. Kuffner, Jr. and Steven M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, volume 2, pp. 995–1001. IEEE, 2000.

Steven M. LaValle and James J. Kuffner, Jr. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020.

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5502–5511, 2022.

Guillaume Matheron, Nicolas Perrin, and Olivier Sigaud. PBCS: Efficient exploration and exploitation using a synergy between reinforcement learning and motion planning. In *Artificial Neural Networks and Machine Learning – ICANN 2020*, volume 12397 of *Lecture Notes in Computer Science*, pp. 295–307. Springer, 2020.

Philippe Morere, Gilad Francis, Tom Blau, and Fabio Ramos. Reinforcement learning with probabilistically complete exploration. *arXiv preprint arXiv:2001.06940*, 2020.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning (ICML)*, pp. 278–287, 1999.

NVIDIA, Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, et al. Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pp. 17473–17498. PMLR, 2022.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

Silviu Pitis, Harris Chan, Stephen Zhao, Bradly Stadie, and Jimmy Ba. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning. In *International Conference on Machine Learning*, pp. 7750–7761. PMLR, 2020.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-Fit: State-covering self-supervised reinforcement learning. In *International Conference on Machine Learning*, pp. 7783–7792. PMLR, 2020.

Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. Teacher algorithms for curriculum learning of deep RL in continuously parameterized environments. In *Conference on Robot Learning*, pp. 835–853. PMLR, 2020a.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4819–4825, 2020b.

Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. *Advances in Neural Information Processing Systems*, 32, 2019.

Cinjon Resnick, Roberta Raileanu, Sanyam Kapoor, Alexander Peysakhovich, Kyunghyun Cho, and Joan Bruna. Backplay: "Man muss immer umkehren". *arXiv preprint arXiv:1807.06919*, 2018.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pp. 91–100. PMLR, 2022.

Tim Salimans and Richard Chen. Learning Montezuma's Revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*, 2018.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Clemens Schwarke, Victor Klemm, Matthijs van der Boon, Marko Bjelonic, and Marco Hutter. Curiosity-driven learning of joint locomotion and manipulation tasks. In *Conference on Robot Learning*, pp. 2594–2610. PMLR, 2023.

Clemens Schwarke, Mayank Mittal, Nikita Rudin, David Hoeller, and Marco Hutter. RSL-RL: A learning library for robotics research. *arXiv preprint arXiv:2509.10771*, 2025.

Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34:18474–18486, 2021.

Stone Tao, Arth Shukla, Tse-kai Chan, and Hao Su. Reverse forward curriculum learning for extreme sample and demo efficiency. In *International Conference on Learning Representations*, pp. 46955–46976, 2024.

Tianjun Zhang, Benjamin Eysenbach, Ruslan Salakhutdinov, Sergey Levine, and Joseph E. Gonzalez. C-Planning: An automatic curriculum for learning goal-reaching tasks. In *International Conference on Learning Representations*, 2022.

A Appendix

A.1 Algorithms

Algorithms 2–4 give the subroutines of the training loop in Algorithm 1, each with the hyperparameters it uses (Table 3). Each direction keeps the same four sets of states: a candidate set C of at most M_C random-walk states, a frontier buffer F holding the states whose rollout return lies strictly between $R_{\min}$ and $R_{\max}$, a solved buffer S holding the states with a return of at least $R_{\max}$, and the mixed set $\mathcal{I}$ from which the exploitation rollouts are drawn. The superscripts fwd and bwd distinguish the forwards from the backwards expansion; states in the forwards buffers enter the rollouts as intermediate goals through ϕ, states in the backwards buffers as intermediate start states. Algorithm 2 expands a frontier by simulating an action-noise random walk from the nearest neighbor of a sampled task space point, Algorithm 3 sorts the collected rollouts into F and S by their return, and Algorithm 4 rebuilds $\mathcal{I}$ from the two buffers, replaying a fraction ν of solved states.

Three details are worth noting. First, in the forwards direction, THRESHOLD stores the final state of a rollout rather than its commanded goal: the forwards buffers seed random walks, which must be reset to full simulator states, whereas a commanded goal is only a point in $\mathcal{G}$. The return of the commanded goal decides whether the rollout is sorted into F or S, so the stored state is the state the policy actually reached while attempting a goal of the corresponding difficulty, a form of hindsight relabeling; for a solved rollout, which reached and held its goal, this state typically lies within ϵ of the commanded goal. Second, the connect samples in RANDOMWALK are drawn from the other direction's solved buffer $S^{\bar{d}}$, i.e., from states the policy already reaches from ρ_0 or from which it already reaches $g^\star$. Third, exploitation is optional: with $N_{\text{exploit}} = 0$, the exploitation rollouts are omitted and the mixed set serves only to seed the random walks. We use this setting on the mazes and *ring-on-peg transfer*, where exploration rollouts alone did not lead to forgetting (Table 3).

Algorithm 2 Random walk frontier expansion

Hyperparameters: connect ratio p_{connect}, action noise covariance Σ, random walk horizon T_B, number of subsamples per walk N_s

procedure RANDOMWALK($\mathcal{I}, \mathcal{T}, S^{\bar{d}}$)
  if $u < p_{\text{connect}}, \quad u \sim \mathcal{U}[0,1]$ **then**
    $x_{\text{rand}} \leftarrow \phi(\text{SAMPLE}(S^{\bar{d}}))$ ▷ connect: sample from the other direction's solved buffer
  else
    $x_{\text{rand}} \leftarrow \text{SAMPLE}(\mathcal{T})$ ▷ sample uniformly from task space
  $s_0 \leftarrow \text{NEARESTNEIGHBOR}(x_{\text{rand}}, \mathcal{I})$ ▷ w.r.t. $\|\phi(\cdot) - x_{\text{rand}}\|$
  for $t = 1, \ldots, T_B$ **do**
    $a_t \sim \mathcal{N}(0, \Sigma)$ ▷ random action
    $s_t \sim P(s_t \mid s_{t-1}, a_t)$ ▷ simulate random walk from nearest neighbor
  $\mathcal{C} \leftarrow \text{SUBSAMPLE}(\{s_1, \ldots, s_{T_B}\}, N_s)$ ▷ new candidates: N_s subsampled states
  return $\mathcal{C}$

Algorithm 3 Return-based thresholding of rollouts into frontier and solved buffers

Hyperparameters: return band $R_{\min}, R_{\max}$, maximum buffer sizes M_F, M_S

procedure THRESHOLD(D, F, S, d)
  for $(\tau, g) \in D$ **do** ▷ explore and exploit rollouts with return $R(\tau, g)$
    if $d =$ fwd **then**
      $x \leftarrow \tau.\text{LAST}(\,)$ ▷ final state, i.e., the achieved goal (Appendix A.1)
    else
      $x \leftarrow \tau.\text{FIRST}(\,)$ ▷ start state
    if $R_{\min} < R(\tau, g) < R_{\max}$ **then**
      $F.\text{APPEND}(x)$ ▷ frontier, at most M_F states
    else if $R(\tau, g) \geq R_{\max}$ **then**
      $S.\text{APPEND}(x)$ ▷ solved, at most M_S states
  return F, S

Algorithm 4 Rebuild of the mixed set from frontier and solved buffers

Hyperparameters: replay ratio ν, mixed set size M_I

procedure MIX(F, S)
  $n_S \leftarrow \min(\lfloor \nu M_I \rfloor, \ |S|)$ ▷ replayed solved states
  $n_F \leftarrow \min(M_I - n_S, \ |F|)$ ▷ rest from the frontier
  $\mathcal{I} \leftarrow \text{SUBSAMPLE}(S, n_S) \cup \text{SUBSAMPLE}(F, n_F)$ ▷ uniform, without replacement
  return $\mathcal{I}$

A.2 EXPERIMENT SETUP

All policies are trained with the PPO implementation of the RSL-RL framework (Schwarke et al., 2025). Table 2 lists the PPO hyperparameters and Table 3 the BVER hyperparameters of all experiments, using these abbreviations: Point-Maze (PM), Climb Up (CU), Climb Down (CD), Three-

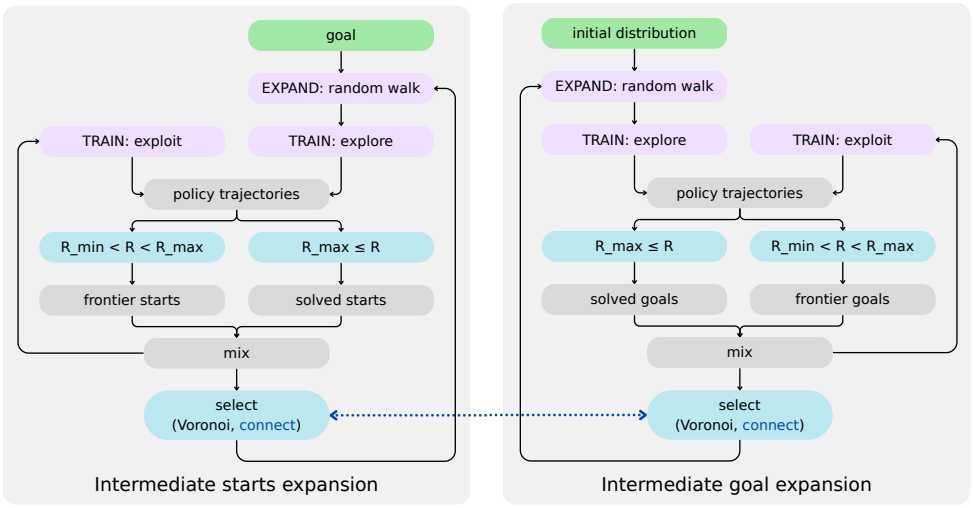


Figure 7: Overview of BVER. The backwards expansion (left) grows intermediate starts from the goal, the forwards expansion (right) intermediate goals from the initial distribution. In each, random walks expand the selected seeds, the policy trains on the new states (explore) and on the mixed set (exploit), the rollouts are sorted by return into frontier and solved sets, and the two are mixed. The next seeds are selected by the Voronoi bias or, with probability p_{connect}, toward the other expansion's solved set (dotted arrow).

Route Terrain (TRT), Freestanding Boulder I (FB I), Freestanding Boulder II (FB II), Rocky Ascent (RA), Small Stone Field (SSF), and Ring-on-peg Transfer (RPT).

Point-Maze (PM). The agent is a velocity-controlled point mass in a walled grid with $4\,\mathrm{m}$ cells, controlled at $20\,\mathrm{Hz}$. The observation is six-dimensional and noise-free: the planar position, the planar velocity, and the vector to the goal; actor and critic receive the same input. The action is a two-dimensional planar velocity command. The reward consists of a single sparse term that is nonzero only within $1\,\mathrm{m}$ of the goal, where it decreases linearly from 1 at the goal to 0 at this tolerance. The episode horizon is $9\,\mathrm{s}$ in the *U-maze*, $32\,\mathrm{s}$ in the *N-maze*, and $15\,\mathrm{s}$ in the *large maze*.

Climb Up/Down (CU, CD) and scans (FB I, FB II, RA, SSF). These tasks use ANYmal D with a $50\,\mathrm{Hz}$ joint-position controller; the 12-dimensional action is an offset from the default joint configuration. The policy observes the base position and angle, the base linear and angular velocity, the projected gravity, the three-dimensional goal position in the base frame, the joint positions and velocities, the previous action, and a height scan over a $2\,\mathrm{m} \times 1\,\mathrm{m}$ grid at $0.1\,\mathrm{m}$ resolution. The policy observations are perturbed with uniform noise; the critic receives the same terms without noise. The reward is sparse: a goal term that decreases linearly from 1 at the goal to 0 at a distance of $0.25\,\mathrm{m}$ and is zero beyond, and a termination penalty for illegal body contact or a bad base orientation. No shaping terms are used. Episodes last $4\,\mathrm{s}$ on the climb tasks, $12\,\mathrm{s}$ on RA, and $10\,\mathrm{s}$ on the other scans.

Three-Route Terrain (TRT). TRT shares the robot, the action space, and the observations, including the base position and yaw, with the tasks above. Unlike the climb and scan tasks, the reward is not purely sparse. In addition to the same sparse goal term, a shaping term based on the distance potential $-\|\phi(s) - g\|$ (Ng et al., 1999) is added to provide a reward signal strong enough for the no-curriculum PPO baseline to find solutions; vanilla PPO and BVER receive the same reward. The term rewards the closing speed, i.e., the base velocity v toward the goal, clipped to $1.5\,\mathrm{m\,s^{-1}}$ and

set to zero within $0.25\,\mathrm{m}$ of the goal:

$$r_{\text{progress}}(s, g) = \begin{cases} \text{clip}\left(v^\top \frac{g-\phi(s)}{\|g-\phi(s)\|}, -1.5, 1.5\right) & \text{if } \|g - \phi(s)\| > 0.25\,\mathrm{m}, \\ 0 & \text{otherwise.} \end{cases}$$

BVER's reward band is evaluated on the sparse term only. Episodes last $10\,\mathrm{s}$. For the route analysis in Section 5.2, we roll out each policy after 2000 training iterations (one seed) in 128 episodes and assign a successful episode to the stairs, pillars, or ramp if the base passes through a coordinate box around the respective obstacle.

Ring-on-peg Transfer (RPT). A Franka Panda with stiff joints holds a ring that is rigidly attached to its hand and must move it from the bottom of one peg to the bottom of another; control runs at $30\,\mathrm{Hz}$. The 27-dimensional observation consists of the seven arm joint positions and velocities, the ring center, the vector from the ring center to the goal, and the previous action. The seven-dimensional action is a relative joint-position target, so a zero action holds the current pose. The reward is the same as in the *point-mass mazes*, measured in three dimensions on the ring center with a tolerance of $3\,\mathrm{cm}$. The episode horizon is $8\,\mathrm{s}$.

Baselines. All baselines use the PPO hyperparameters of Table 2 and the same number of environments and simulation steps per iteration as BVER. The random curriculum draws intermediate starts and goals from a collection of states sampled offline uniformly from the task space and kept if they are collision-free; it does not filter them by return. RSI and Backplay reset to states of a single demonstration, a successful rollout of a trained BVER policy; any successful policy could provide it. We implemented RC in Isaac Lab following the original algorithm (Florensa et al., 2017) as closely as possible.

Table 2: PPO hyperparameters for all experiments.

	Experiments		
Hyperparameters	PM	CU, CD, TRT, FB I, FB II, RA, SSF	RPT
MLP hidden dimensions	$[128, 64, 16]$	$[512, 256, 128]$	$[256, 128, 64]$
Activation function	*ELU*	*ELU*	*ELU*
Value loss coefficient	1.0	1.0	1.0
Clip parameter	0.2	0.2	0.2
Entropy coefficient	0.01	0.0025	0.002
Learning rate	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
Discount factor γ	0.99	0.99	0.99
λ (GAE)	0.95	0.95	0.95
Desired KL divergence	0.01	0.01	0.01

A.3 Ablation learning curves

Figures 8 and 9 show the learning curves of the ablations summarized in Table 1. With a single expansion direction, the success rate often plateaus below that of BVER, most clearly on *ring-on-peg transfer*, where each unidirectional variant expands from its peg into the free space in between but rarely reaches the other peg. For the connect step, both $p_{\text{connect}} = 0$ and $p_{\text{connect}} = 1$ rise later than the default $p_{\text{connect}} = 0.5$ on all three mazes: without connecting, the two frontiers spread evenly over $\mathcal{T}$ and likely take longer to meet, and with greedy connecting, the expansion loses its coverage bias.

Table 3: BVER hyperparameters for all experiments. $R_{\min}$ and $R_{\max}$ are thresholds on the normalized return of Equation 1. The connect ratio p_{connect} and the number of subsamples per walk N_s are shared across all tasks; the other hyperparameters are set per domain, and T_B and the environment split also differ between the mazes. All hyperparameters were selected empirically per domain in preliminary runs.

	Experiments						
Hyperparameters	PM-U	PM-N	PM-L	TRT	CU, CD	FB I, FB II, RA, SSF	RPT
Env split N_{exploit}	0	0	0	2048	2048	1024	0
Env split N_{explore}	512	4096	4096	4096	4096	2048	2048
$R_{\min}$	0.02	0.02	0.02	0.05	0.05	0.05	0.1
$R_{\max}$	0.95	0.95	0.95	0.2	0.2	0.2	0.85
Replay ratio ν	0.2	0.2	0.2	0.05	0.05	0.05	0.1
p_{connect}	0.5	0.5	0.5	0.5	0.5	0.5	0.5
Frontier buffer size M_F	10^3	10^3	10^3	$2 \cdot 10^3$	$2 \cdot 10^3$	$2 \cdot 10^3$	10^3
Solved buffer size M_S	10^3	10^3	10^3	10^4	10^4	10^4	10^3
Candidate set size M_C	500	500	500	$5 \cdot 10^3$	$5 \cdot 10^3$	$5 \cdot 10^3$	500
Mixed set size M_I	10^3	10^3	10^3	$2 \cdot 10^3$	$2 \cdot 10^3$	$2 \cdot 10^3$	10^3
Subsamples per walk N_s	4	4	4	4	4	4	4
Random walk horizon T_B	80	200	80	128	64	64	10
Action noise Σ	$10 \cdot \mathbf{I}$	$10 \cdot \mathbf{I}$	$10 \cdot \mathbf{I}$	$10 \cdot \mathbf{I}$	$10 \cdot \mathbf{I}$	$10 \cdot \mathbf{I}$	$3 \cdot \mathbf{I}$

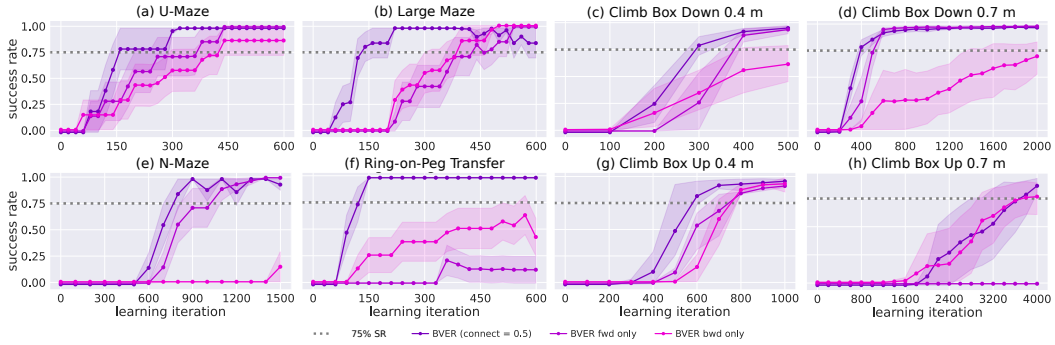


Figure 8: Expansion direction ablation. Combining forwards and backwards expansions reaches 75% success first on all tasks, tied with the backwards expansion on climbing up the 0.7 m box (h). Which direction alone works depends on the task: on *box climbing* (c, d, g, h), the expansion starting on top of the box is stronger, and on *ring-on-peg transfer* (f), neither direction alone reaches 75% success. The dashed line marks the 75% success rate (SR) used in Table 1.

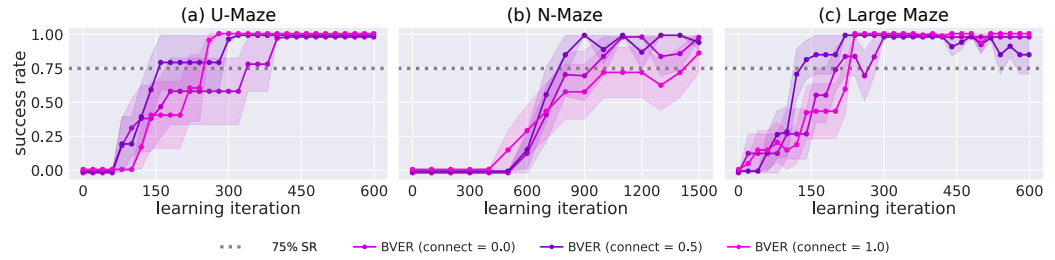


Figure 9: Connect ratio ablation on the three mazes. The default $p_{\text{connect}} = 0.5$ learns fastest on all three mazes; seeding the random walks only from uniform task space samples ($p_{\text{connect}} = 0$) or only from the other direction's solved set ($p_{\text{connect}} = 1$) slows down learning. The dashed line marks the 75% success rate (SR) used in Table 1.