JUMPY PLANNING OVER POLICY OCCUPANCIES IMPROVES ONLINE GOAL-CONDITIONED RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Online goal-conditioned reinforcement learning aims to learn a single policy for many goals, but sparse feedback makes long-horizon behaviors difficult to discover from interaction alone. Planning with temporal abstractions through Geometric Policy Composition could accelerate the acquisition of such behaviors but existing methods have largely focused on fixed, pretrained policies and world models. We introduce *Online Geometric Policy Composition* (OGPC), which plans over short subgoal sequences to guide exploration while learning both the policy and its world model online *from scratch*. OGPC uses a Geometric Horizon Model to predict which states the policy reaches while pursuing each subgoal, and chains these predictions to evaluate candidate sequences. To score candidate plans reliably under sparse rewards, we replace the final sampled reward in the existing Monte Carlo estimator with a learned estimate of its conditional success probability. With this estimator, OGPC improves sample efficiency and final performance over a strong, depth-scaled Contrastive Reinforcement Learning policy across nine JaxGCRL tasks, including high-dimensional Humanoid environments. The gains persist without planning at evaluation, showing that temporally abstract planning improves the learned policy rather than only decision-time behavior.

1 INTRODUCTION

Goal-Conditioned Reinforcement Learning (GCRL) provides a promising avenue for learning policies that can reach a diverse set of goals, without manually specifying a separate reward function for each goal. Acquiring this broad repertoire of behaviors becomes challenging in long-horizon tasks that require coordinated behavior over extended sequences of decisions, while the task-level feedback remains sparse. This challenge is particularly pronounced in online settings, where the agent needs to learn only through interactions with its environment.

Decision-time planning can improve online policy learning by evaluating candidate behaviors before execution and using these evaluations to guide interaction (Silver et al., 2018; Schrittwieser et al., 2020; Hamrick et al., 2021). In long-horizon GCRL, however, goal-reaching feedback is often sparse and may only appear after many actions. This presents significant challenges for conventional planning approaches that search over primitive-action sequences using repeated single-step predictions (Chua et al., 2018; Hansen et al., 2024). Explicitly evaluating these distant outcomes requires long model rollouts and search over long action sequences, increasing computational cost and exposure to compounding prediction errors (Asadi et al., 2019; Wang et al., 2019). Planners can keep this search tractable by shortening rollouts and using a learned terminal value function to evaluate outcomes beyond the rollout horizon. However, this makes action selection more dependent on potentially biased value estimates that can cause search to favor suboptimal plans with overestimated returns (Feinberg et al., 2018; Chang et al., 2026).

We introduce *Online Geometric Policy Composition* (OGPC), which uses a learned, temporally abstract world model to plan over short sequences of subgoals rather than long sequences of primitive actions. OGPC uses receding-horizon Model-Predictive Control (MPC) to optimize a first action and a short sequence of subgoals (Figure 1). Geometric Horizon Models (GHMs) (Janner et al., 2020; Thakoor et al., 2022) act as *jumpy world models*, predicting the policy occupancies (Dayan, 1993; Blier et al., 2021)[1] induced as the current goal-conditioned policy pursues each subgoal, while

[1] We represent policy occupancy by the normalized successor measure defined in Section 2.

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.

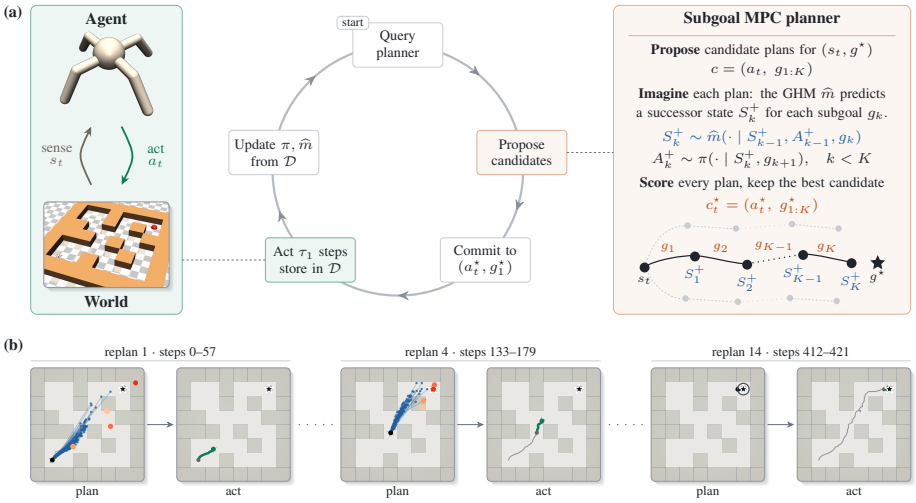


Figure 1: **Online Geometric Policy Composition (OGPC).** (a) At each planning step, OGPC proposes candidate first actions and subgoal sequences, predicts their outcomes using the current policy π and GHM $\widehat{m}$, and selects the highest-scoring plan. The agent executes the selected first action, then pursues the first subgoal for the remainder of a geometrically sampled τ_1-step commitment before replanning. Transitions are stored in the replay buffer $\mathcal{D}$. (b) Selected planning steps from one episode. Orange markers represent planned subgoals (g_1 light to g_K dark), and blue points are sampled GHM successor states. Green and gray segments show trajectories executed during and before the commitment, respectively. A single ring around the star marks a plan whose subgoals all coincide with the task goal $g^\star$.

Geometric Policy Composition (GPC) (Thakoor et al., 2022) composes these predictions to evaluate long-horizon behavior in a few jumps. Unlike prior empirical applications with fixed, pretrained policies and GHMs (Thakoor et al., 2022; Farebrother et al., 2026), OGPC learns a shared goal-conditioned policy and its GHM concurrently *from scratch* within the online interaction loop.

However, temporal abstraction is only part of the solution. With sparse goal rewards and a limited sampling budget, the Monte Carlo scoring used in prior GPC work (Thakoor et al., 2022; Farebrother et al., 2026) can assign identical zero scores to plans with different chances of reaching the goal. These zero-score ties can render planning ineffective at guiding the agent's interactions as the scores provide no basis for choosing among the tied plans. To mitigate this effect, we introduce a terminal-conditional GPC estimator that replaces the final sampled reward with the expected probability of success given the simulated trajectory so far. Intuitively, a promising plan receives credit according to its probability of reaching the goal, rather than only when a terminal sample happens to succeed. In continuous-state settings, we train a neural network online to approximate this probability (Figure 3). Our analysis shows that using the exact conditional occupancy preserves the expected plan scores under the current GHM and removes the additional variance from sampling the final reward.

We evaluate OGPC on nine navigation and control tasks from the JaxGCRL benchmark (Bortkiewicz et al., 2025). The terminal-conditioned variant (OGPC-TC) improves sample efficiency and final time at goal over planner-free depth-scaled CRL (Wang et al., 2025), with its clearest advantage over the literal Monte Carlo variant (OGPC-MC) on Ant Big Maze, Ant Hardest Maze, and Humanoid U-Maze. The Humanoid results show that this advantage extends beyond Ant navigation to higher-dimensional control with a deeper policy. Because planning is removed at evaluation, these gains reflect a stronger learned goal-conditioned policy rather than additional test-time computation.

In summary, our key contributions are:

1. **Online jumpy planning.** We introduce OGPC, which uses GPC to train a goal-conditioned policy and its GHM from scratch in an online GCRL setting.
2. **Terminal-conditioned plan scoring.** We derive a terminal-conditional GPC estimator, characterize its variance reduction, and learn a neural approximation to the required conditional probability.

3. **Sample-efficient policy improvement.** Across nine JaxGCRL tasks, we show that OGPC-TC improves sample efficiency and final Time at Goal over planner-free depth-scaled CRL.

2 BACKGROUND

We model the environment as a goal-conditioned Markov Decision Process (MDP) $\mathcal{M}_{\mathcal{G}} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces, $\mathcal{G}$ the goal space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ the transition kernel, $r : \mathcal{S} \times \mathcal{G} \to \{0, 1\}$ the reward, and $\gamma \in [0, 1)$ the discount factor. Each goal $g \in \mathcal{G}$ specifies a goal region $\mathcal{X}_g \subseteq \mathcal{S}$, the set of states that satisfy the task's success criterion (Appendix E.1). The goal-conditioned reward is the indicator of that region, $r_g(s') := r(s', g) = \mathbb{1}\{s' \in \mathcal{X}_g\}$, evaluated on the arrival state s'.

For a goal-conditioned policy $\pi_\theta : \mathcal{S} \times \mathcal{G} \to \Delta(\mathcal{A})$, we write $\pi_g(a \mid s) := \pi_\theta(a \mid s, g)$. The commanded task goal $g^\star$ determines the reward, while candidate subgoals $g_{1:K}$ condition the policy's behavior. All subgoal-conditioned policies share parameters θ. Bare g denotes a generic goal in definitions and learning objectives. We use t for environment timesteps and k for compositional hops. We write S' for an immediate next state, S^+ for a future successor state and S_k^+ for the successor after k composed hops. Uppercase state and action symbols denote random variables, and lowercase symbols denote fixed values. A comprehensive list of notations is reported in Appendix A .

Successor Measure We recall the definition of a (normalized) successor measure (Dayan, 1993; Blier et al., 2021). For any measurable subset $\mathcal{X} \subseteq \mathcal{S}$,

$$m_\gamma^\pi(\mathcal{X} \mid s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr^\pi \left(S_{t+1} \in \mathcal{X} \mid S_0 = s, A_0 = a\right). \tag{1}$$

Here $\Pr^\pi$ denotes probability under transition kernel P, starting from (s, a) and following π thereafter. The normalization factor $1 - \gamma$ ensures that $m_\gamma^\pi(\mathcal{S} \mid s, a) = 1$. The action-value function associated with the goal-conditioned reward r_g can be expressed as

$$Q_{\gamma,g}^\pi(s, a) = (1 - \gamma)^{-1} \mathbb{E}_{S^+ \sim m_\gamma^\pi(\cdot|s,a)} \left[r_g(S^+)\right] = (1 - \gamma)^{-1} m_\gamma^\pi(\mathcal{X}_g \mid s, a). \tag{2}$$

Geometric Horizon Models A geometric horizon model (GHM) is a generative model of the successor measure (Janner et al., 2020; Thakoor et al., 2022). We write $\widehat{m}_\gamma^\pi(\cdot \mid s, a)$ for its output distribution, which approximates $m_\gamma^\pi(\cdot \mid s, a)$. The normalized successor measure satisfies the Bellman identity

$$m_\gamma^\pi(\cdot \mid s, a) = (1 - \gamma) P(\cdot \mid s, a) + \gamma \mathbb{E}_{S', A'} \left[m_\gamma^\pi(\cdot \mid S', A')\right], \tag{3}$$

where $S' \sim P(\cdot \mid s, a)$ and $A' \sim \pi(\cdot \mid S')$. In this work, we instantiate the GHM as a Temporal Difference Flows (TD-Flows) model (Farebrother et al., 2025), trained using the Temporal Difference Horizon Consistency (TD-HC) objective (Farebrother et al., 2026). We defer the complete derivation of the learning objectives to Appendix B.3.

Geometric Policy Composition Consider K policies $\pi_{g_1}, \ldots, \pi_{g_K}$. After each step, component $k < K$ switches to the next policy with probability $\alpha_k \in [0, 1]$, or continues with probability $1 - \alpha_k$. This defines a *Geometric Switching Policy* (GSP) (Thakoor et al., 2022):

$$\nu := \pi_{g_1} \xrightarrow{\alpha_1} \pi_{g_2} \cdots \xrightarrow{\alpha_{K-1}} \pi_{g_K}. \tag{4}$$

Combining the switching rule with discounting gives the effective GHM discount $\beta_k := \gamma(1 - \alpha_k)$. Since the final component never switches, we have $\alpha_K = 0$ and $\beta_K = \gamma$.

Definition 2.1. Let $\nu := \pi_{g_1} \xrightarrow{\alpha_1} \pi_{g_2} \cdots \xrightarrow{\alpha_{K-1}} \pi_{g_K}$ be a geometric switching policy with discount factor $\gamma \in [0, 1)$ and effective discounts $\beta_k = \gamma(1-\alpha_k)$ for $k = 1, \ldots, K$. The weight of component k is

$$w_k := \frac{1 - \gamma}{1 - \beta_k} \prod_{i=1}^{k-1} \frac{\gamma - \beta_i}{1 - \beta_i}.$$

w_k characterizes the contribution of component k to the successor measure induced by ν (Farebrother et al., 2026, Theorem 1). To evaluate the composition, set $(S_0^+, A_0^+) = (s, a)$ and recursively sample

$$S_k^+ \sim m_{\beta_k}^{\pi_{g_k}}\left(\cdot \mid S_{k-1}^+, A_{k-1}^+\right), \quad k = 1, \ldots, K, \qquad A_k^+ \sim \pi_{g_{k+1}}(\cdot \mid S_k^+), \quad k < K. \tag{5}$$

The successor measure of ν is a mixture of the endpoint distributions of S_k^+, with weights w_k (Appendices B.1–B.2). Applying equation 2 gives the value of ν for the commanded goal $g^\star$:

$$Q_{\gamma,g^\star}^{\nu}(s,a) = (1-\gamma)^{-1} \sum_{k=1}^{K} w_k \mathbb{E}\left[r_{g^\star}(S_k^+)\right] = (1-\gamma)^{-1} \sum_{k=1}^{K} w_k \Pr\left(S_k^+ \in \mathcal{X}_{g^\star}\right). \tag{6}$$

Averaging the composed rewards over N independently sampled successor sequences, indexed by n, yields the Monte Carlo estimator (Farebrother et al., 2026, Lemma 1)

$$\overline{Q}_{\gamma,g^\star,N}^{\nu,\mathrm{MC}}(s,a) := \frac{1}{N(1-\gamma)} \sum_{n=1}^{N} \sum_{k=1}^{K} w_k r_{g^\star}\left(S_k^{+,(n)}\right). \tag{7}$$

With exact successor measures, this average is unbiased for $Q_{\gamma,g^\star}^{\nu}(s,a)$, with variance $1/N$ that of a single sequence. With learned GHMs, it is unbiased for the corresponding model value.

3 Online Geometric Policy Composition

OGPC learns a goal-conditioned policy and its GHM from scratch, using temporally abstract planning for collection and the policy alone for evaluation (Figure 1).

3.1 Planning and online learning

At a replanning boundary t, OGPC uses the current GHM $\widehat{m}_t$ to score candidates $c = (a, g_{1:K})$, each specifying an initial action and a sequence of subgoals. Its planning objective is

$$c_t^\star := (a_t^\star, g_{1:K}^\star) \in \arg\max_{c=(a,g_{1:K})} Q_{\gamma,g^\star}^{\nu_c,\widehat{m}_t}(S_t, a). \tag{8}$$

We approximately solve equation 8 with iCEM (Pinneri et al., 2021), using candidate scores estimated from N independent GHM simulations. In our implementation, we seed the initial population with the *direct-goal plan* c^0 shown in Algorithm 1. We keep the highest-scoring candidate seen across all rounds, retaining the earlier candidate on ties. The Boolean keep_seed indicates whether this original seed remains selected.

After selecting $c_t^\star$, we execute $a_t^\star$ and follow $\pi_{g_1^\star}$ for the remainder of a commitment lasting $\tau_1 \sim \mathrm{Geom}(1-\beta_1)$ environment steps, capped at $\tau_{\max}$. The planner replans when this commitment ends or a new episode starts, but not when the policy reaches its subgoal. If the original seed c^0 is retained, we instead sample a new action from $\pi_{g^\star}$ at each step of the commitment, including the first. Full algorithms and planner settings are given in Appendices C and E.3.3.

Algorithm 1 OGPC: online loop

Initialize $\pi_\theta, \widehat{m}_\vartheta$, and replay $\mathcal{D}$.
if terminal scoring: initialize head q_ϕ.
for each step after planner-free warm-up **do**
  if a new commitment starts **then**
    $a^0 \sim \pi_\theta(\cdot \mid s, g^\star)$
    $c^0 \leftarrow (a^0, g^\star, \ldots, g^\star)$
    if literal scoring: $\overline{Q}_N \leftarrow$ MC average (7)
    else $\overline{Q}_N \leftarrow$ learned score (16)
    $(c^\star, \mathsf{keep_seed}) \leftarrow \mathrm{iCEM}_{\overline{Q}_N}(s, g^\star; c^0)$
    $\tau_1 \sim \min\{\mathrm{Geom}(1-\beta_1), \tau_{\max}\}$
  $a \sim \pi_\theta(\cdot \mid s, g_1^\star)$
  if first step and **not** keep_seed: $a \leftarrow a^\star$
  Execute a and observe the next state s'.
  Store the transition in $\mathcal{D}$ and set $s \leftarrow s'$.
  if update scheduled **then**
    Update $\pi_\theta, \widehat{m}_\vartheta$ from $\mathcal{D}$.
    if terminal scoring: update q_ϕ from $\mathcal{D}$.
return π_θ; evaluate without planning.

3.2 Terminal-conditioned plan scoring

OGPC with literal Monte Carlo scoring, denoted **OGPC-MC**, uses an unbiased estimator of the value induced by the current GHM. However, sparse goal rewards and a limited sampling budget

can leave different candidates tied at zero. We replace the final sampled indicator by its conditional success probability given the simulated prefix. This Rao–Blackwellization (Rao, 1945; Blackwell, 1947) preserves the expected model-relative score while removing terminal sampling noise.

Fix the current GHM, a query $(s, a, g^\star)$, and a candidate ν with the composition weights defined in the background. For one composed simulation, write

$$R_{<K} := \sum_{k=1}^{K-1} w_k r_{g^\star}(S_k^+), \qquad B_K := r_{g^\star}(S_K^+) \in \{0, 1\}. \tag{9}$$

Let $\mathcal{F}_{K-1}$ denote the information in the simulated prefix through the state–action input to the final hop. Its conditional terminal occupancy is

$$q_{\widehat{m},g^\star} := \mathbb{E}_{\widehat{m}}[B_K \mid \mathcal{F}_{K-1}] = \widehat{m}_\gamma^{\pi_{g_K}}(\mathcal{X}_{g^\star} \mid S_{K-1}^+, A_{K-1}^+). \tag{10}$$

For $K = 1$, this input is (s, a) and the simulated prefix is empty. For this fixed query and candidate, the two single-simulation estimators are

$$\widehat{Q}^{\mathrm{MC}} := (1-\gamma)^{-1}(R_{<K} + w_K B_K), \tag{11}$$

$$\widehat{Q}^{\mathrm{RB}} := (1-\gamma)^{-1}(R_{<K} + w_K q_{\widehat{m},g^\star}). \tag{12}$$

Theorem 1 (Terminal Rao–Blackwellization)**.** *For a fixed current GHM and candidate ν,*

$$\widehat{Q}^{\mathrm{RB}} = \mathbb{E}_{\widehat{m}}[\widehat{Q}^{\mathrm{MC}} \mid \mathcal{F}_{K-1}]. \tag{13}$$

Consequently, exact terminal conditioning preserves the expected current-GHM value and removes the conditional terminal-reward variance:

$$\mathbb{E}_{\widehat{m}}[\widehat{Q}^{\mathrm{RB}}] = \mathbb{E}_{\widehat{m}}[\widehat{Q}^{\mathrm{MC}}] = Q_{\gamma,g^\star}^{\nu,\widehat{m}}(s, a), \tag{14}$$

$$\mathrm{Var}_{\widehat{m}}(\widehat{Q}^{\mathrm{MC}}) - \mathrm{Var}_{\widehat{m}}(\widehat{Q}^{\mathrm{RB}}) = \frac{w_K^2}{(1-\gamma)^2}\mathbb{E}_{\widehat{m}}[q_{\widehat{m},g^\star}(1 - q_{\widehat{m},g^\star})] \geq 0. \tag{15}$$

The inequality is strict whenever $w_K > 0$ and terminal success remains uncertain after observing the prefix with positive probability.

The proof and sample-ratio result are in Appendices D.2 and D.3. Averaging N independent rows divides either variance by N. The identities also hold conditional on each call's training history, which fixes the policy and GHM (Appendix D.6).

From scoring variance to plan selection. With zero prefix rewards, MC scores tie at zero whenever all terminal samples miss. Terminal conditioning instead retains their conditional success probabilities, providing information that distinguishes candidates even without a sampled terminal hit. We now ask whether this translates to performance improvement. For a fixed candidate set, selection regret $Q_\star - Q_{\hat{i}}^E$, where $E \in \{\mathrm{MC}, \mathrm{RB}\}$, measures the value lost by selecting candidate $\hat{i}$ instead of the best candidate under the current GHM.

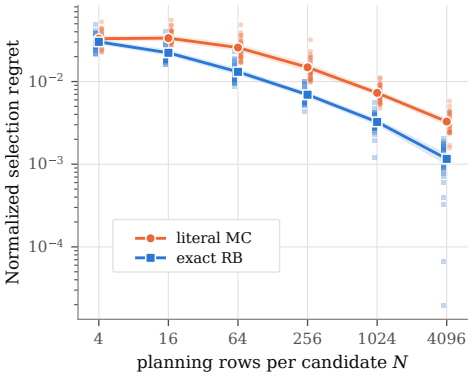


Figure 2: **Tabular selection regret.** Literal MC and exact RB on 30 instances. Mean regret is normalized by the best candidate's value $Q_\star$.

Does more reliable scoring result in lower regret? Theorem 1 shows that exact terminal conditioning reduces scoring variance without changing expected candidate values. Figure 2 illustrates the resulting selection benefit: at matched simulation budgets, exact RB achieves lower mean regret over the same candidate sets. Appendix D.4 relates this advantage to higher planning signal-to-noise ratio (SNR), derives a variance-sensitive selection-regret bound, and tests whether SNR explains the observed ranking reliability. We provide complementary evidence for learned terminal conditioning in the continuous domains (Appendices D.4 and F.5), supporting more reliable planning as a possible contributor to OGPC-TC's faster learning in Section 4. Additional estimator comparisons and the tabular setup are shown in Figure 8 (Appendix D). Appendix D.7 reports a theoretical study on the impact of number of subgoals in continuous domains.

Learning the terminal occupancy. In continuous domains, the conditional occupancy in equation 10 is unavailable in closed form, so we learn a terminal head q_ϕ online from replay (Figure 3). We call this terminal-conditioned variant **OGPC-TC**. For a replay pair (s, a), a command g_K, and a task goal $g^\star$, we draw L endpoints $\widetilde{S}^+_\ell \sim \widehat{m}^{\pi_{g_K}}_\gamma(\cdot \mid s, a)$ from the current GHM and fit $q_\phi := q_\phi(s, a, g_K, g^\star, \gamma)$ to the fraction of sampled endpoints that lie in the goal region, using a soft binary cross-entropy loss,

$$\mathcal{L}(\phi) = \mathbb{E}[\mathrm{BCE}(q_\phi, \widehat{y})], \quad \widehat{y} := \frac{1}{L}\sum_{\ell=1}^{L} r_{g^\star}(\widetilde{S}^+_\ell).$$

The head's architecture, its replay-based command and task-goal sampling, and the online update are given in Appendix E.3.2 and Algorithm 7. For each of N independent simulated prefixes, we query the head at $(S^+_{K-1}, A^+_{K-1}, g_K, g^\star, \gamma)$, denoting its output by $q^{(n)}_\phi$. The candidate score averages these N simulations, retaining the weighted prefix rewards:

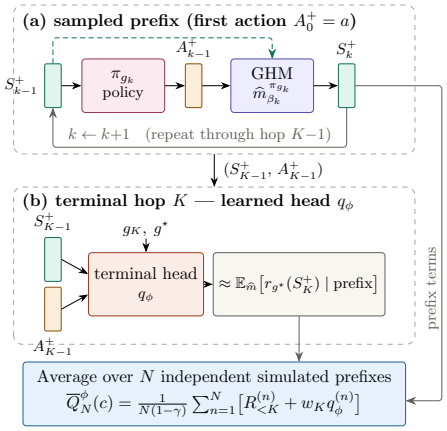


Figure 3: **Terminal conditional scoring.** The first $K-1$ hops form a sampled prefix. The learned head replaces the final sampled reward while retaining the weighted prefix rewards. Each candidate score averages N independent simulations.

$$\overline{Q}^\phi_N(c) := \tfrac{1}{N(1-\gamma)}\textstyle\sum_{n=1}^{N}\big(R^{(n)}_{<K} + w_K q^{(n)}_\phi\big), \tag{16}$$

Unlike exact terminal conditioning, the learned head introduces a further approximation error that can bias candidate scores. If this error is uniformly bounded by ϵ_q on the scored inputs, it adds at most $w_K\epsilon_q/(1-\gamma)$ to the exact-RB estimation radius. This error is separate from GHM error relative to the environment. The resulting selection-regret bound is given in Theorem 8 (Appendix D.5).

4 Experiments

We evaluate OGPC on nine JaxGCRL environments (Bortkiewicz et al., 2025). We then examine the planner's decisions during learning and the effect of changing the number of planned subgoals.

Benchmark and protocol All methods build on the same depth-scaled CRL implementation (Eysenbach et al., 2022; Wang et al., 2025) for the goal-conditioned policy, with the method-specific modifications described below. Each method trains its policy from scratch, and evaluation executes that policy alone, without planning. The Ant mazes (U-Maze, Big Maze, Hardest Maze), Pusher Easy, Ant Push, and Arm BinPick Easy use depth-8 policies, trained for 50M steps with 10 seeds. Humanoid and its U-Maze and Big Maze variants use depth-64 policies, trained for 100M steps with 5 seeds. On the Humanoid tasks, we compare CRL, OGPC-TC, OGPC-MC, DCP, and SF^2. We report *Time at Goal*: steps spent in the goal region during a 1,000-step episode, reflecting both arrival time and subsequent occupancy. *Success Rate* measures whether the goal is reached at least once (Appendix F).

Methods We compare three families of methods. *Temporally extended planning:* **OGPC-TC** and **OGPC-MC** share the planner and policy learning objective, differing only in their scoring rule: learned terminal conditioning versus literal Monte Carlo. Both variants plan over $K = 4$ subgoals, unless stated otherwise. *Planner-free:* **CRL** collects data with the policy alone; **DCP** conditions the actor on retrieved subgoal directions (Swaminathan et al., 2026); and **SF^2** adds a successor-flow objective to the CRL representation without planning (Shi et al., 2026). *Primitive-action planning:* **OGPC-SS** (single-step) sets $\alpha_1 = \ldots = \alpha_{K-1} = 1$, so the planner optimizes primitive actions through a single-step GHM (Appendix E.3.4), and **GC-TD-MPC2** (Rupf et al., 2025; Hansen et al., 2024) plans primitive actions in a learned latent model. Both use the same number of iCEM candidates and iterations as OGPC.

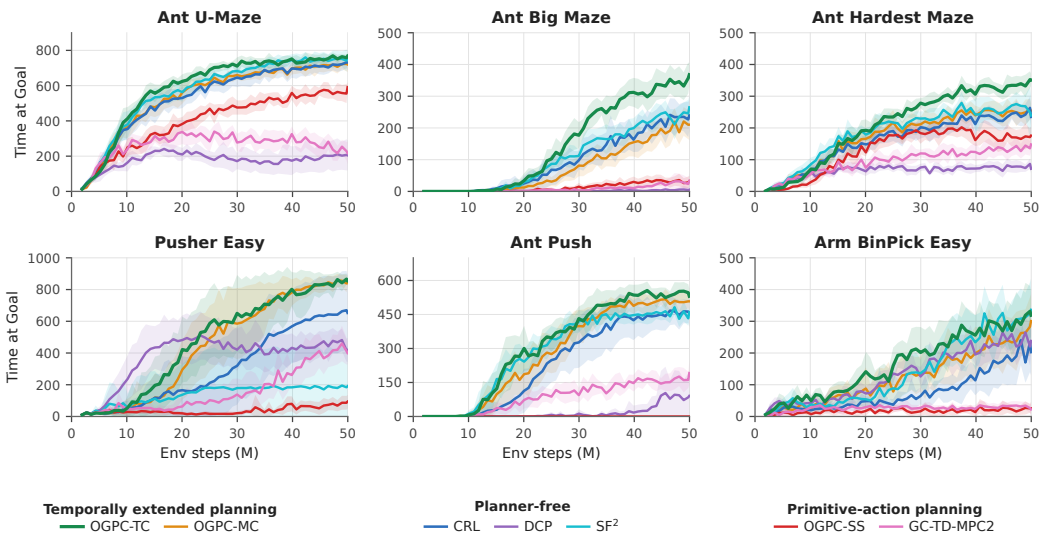


Figure 4: **Time at Goal during training on six JaxGCRL tasks.** Evaluation uses the learned policy without planning. Unsmoothed mean and 95% seed-t CI over 10 seeds.

4.1 Q1: Does planner-collected data improve the planner-free policy?

OGPC-TC improves final mean Time at Goal over CRL on all six tasks in Figure 4, with the largest gains on Ant Big Maze and Ant Hardest Maze. OGPC-MC performs comparably on Pusher Easy, Ant Push and Arm BinPick Easy. However, it achieves similar final Time at Goal to CRL on U-Maze and Hardest Maze, and performs worse on Big Maze. OGPC-SS and GC-TD-MPC2 achieve lower mean final Time at Goal than CRL on all six tasks.[2] DCP initially learns faster on Pusher Easy before plateauing early, and achieves slightly higher mean Time at Goal than CRL on Arm BinPick Easy. However, its performance remains substantially below CRL on the Ant mazes and Ant Push. Interestingly, we find that SF^2 achieves comparable final Time at Goal to OGPC-TC on Arm BinPick Easy without planning. Success Rate curves and per-seed final results are shown in Appendix F.1. A further breakdown of how Time at Goal varies with geodesic distance to the goals is detailed in Appendix F.2.

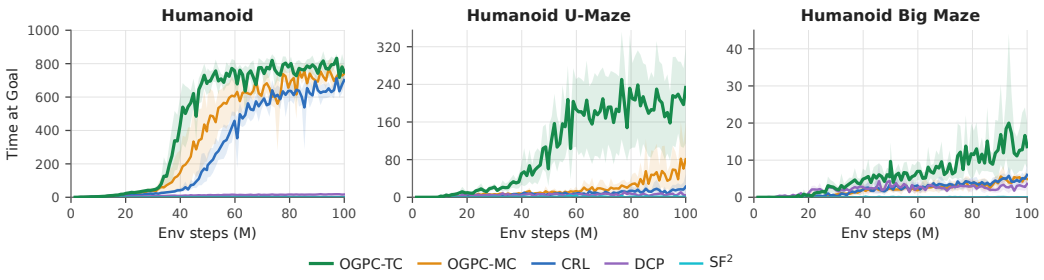


Figure 5: **Time at Goal on the Humanoid tasks** with a depth-64 CRL policy. Unsmoothed mean and pointwise 95% seed-bootstrap CI over 5 seeds.

4.2 Q2: Does OGPC extend to high-dimensional control?

To test whether OGPC scales to higher-dimensional control tasks, we compare OGPC-TC and OGPC-MC with planner-free baselines on Humanoid and its maze variants (Figure 5). On Humanoid, OGPC-TC learns faster than CRL and moderately improves final Time at Goal. On Humanoid U-Maze, OGPC-TC shows a significant improvement over both. Big Maze remains difficult

[2]We also evaluate GC-TD-MPC2 with a policy prior, using both the original and a larger candidate population. Neither configuration closes the gap to planner-free CRL in final Time at Goal (Appendix F.8.1).

for all the evaluated methods with a budget of 100M environment steps, though we note that OGPC-TC still achieves the highest final Time at Goal across all 5 seeds. This corroborates our earlier findings on the Ant Big Maze and Hardest Maze, and terminal conditioning's advantage over literal MC estimator extends to high-dimensional control settings. Additional results appear in Appendix F.1.

4.3 Q3: How does subgoal planning guide data collection and learning?

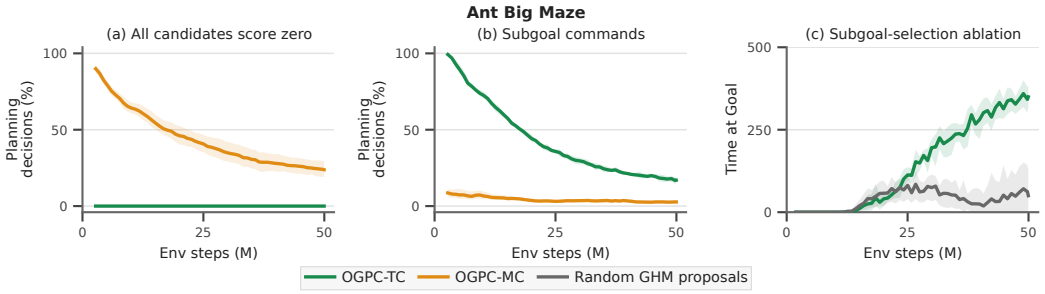


Figure 6: **Planning on Ant Big Maze.** Percentage of planning decisions during training with (a) all candidate scores equal to zero or (b) a first subgoal more than one success radius from the task goal. (c) Planner-free Time at Goal for OGPC-TC and random GHM proposals. Lines show five-seed means, and shaded regions denote 95% bootstrap CIs.

We record online planning decisions on Ant Big Maze, where most MC searches initially score every candidate zero. In these searches, the planner retains the initial direct-goal seed, and the policy acts towards the commanded goal without subgoal guidance. While all-zero searches become less frequent during training, intermediate-subgoal commands remain rare (Figure 6a,b). We hypothesize that persistent all-zero searches restrict the planner's ability to shape the experience from which the policy learns. On the other hand, TC commands intermediate subgoals frequently early in training, and less often as the policy improves. At the same sampling budget, TC supports subgoal-guided collection even when sparse MC scores leave the planner unable to distinguish candidate plans. To test whether subgoal proposals alone are sufficient, we evaluate randomly sampled subgoal sequences without scored search or comparison with the direct-goal plan. We find that this control yields substantially lower Time at Goal than OGPC-TC (Figure 6c). We further examine how terminal conditioning improves planning SNR and ranking reliability, at matched sampling budget, using identical candidate populations under the same frozen GHMs (Appendices F.5 and D.4).

4.4 Q4: How does the number of planned subgoals affect learning?

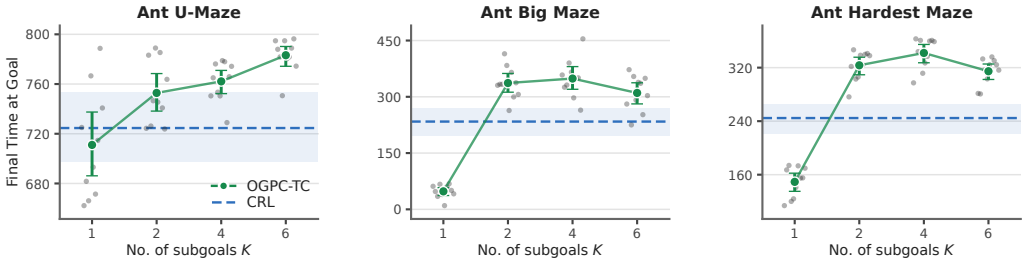


Figure 7: **Number of subgoals during learning.** Final Time at Goal after 50M steps, evaluated without planning, against the number of subgoals used during collection. Markers: seed means with 95% bootstrap CIs. Dots: individual seeds (per K). Dashed line and band: CRL mean and CI.

We vary the number of subgoals K in OGPC-TC's plans during training and evaluate the resulting goal-conditioned policy without planning (Figure 7). With $K = 1$, the planner evaluates an initial action followed by a single subgoal-conditioned policy, without composing GHM predictions across subgoals. On Ant U-Maze, the mean continues to increase through six subgoals. On Ant Big Maze and Ant Hardest Maze however, the mean final Time at Goal increases from one to four subgoals, then decreases with six. With a single subgoal ($K = 1$), final Time at Goal is substantially lower than planner-free CRL on both larger mazes. Planning over more subgoals can improve policy learning, but it does not necessarily yield monotonic improvement over the planner-free pol-

icy across environments. We examine the effect of varying K at inference time after training in Appendix F.3.

5 Related Work

Successor Representation and Geometric Policy Composition. GHMs build upon Successor representations (Dayan, 1993) and Temporal Difference models (Sutton, 1995) . Thakoor et al. (2022) establish compositional evaluation of GSPs and Geometric Generalized Policy Improvement (GGPI), including application to policy iteration. TD-Flows (Farebrother et al., 2025) learn GHMs with flow matching. CompPlan (Farebrother et al., 2026) adds horizon consistency and component-specific timescales, demonstrating planning with policies and models trained from offline data. Successor Flow Features (Shi et al., 2026) use a successor-flow objective to improve CRL representations without planning. OGPC instead uses a learned occupancy model to evaluate candidate subgoal sequences during data collection. Unlike the prior empirical settings of Thakoor et al. (2022) and Farebrother et al. (2026), we study online GCRL with the goal-conditioned policy and its GHM learned together from scratch. We also improve upon the original MC estimator by replacing the final sampled reward with a learned conditional occupancy estimate.

Decision-time planning with temporal abstractions. The options framework formalizes temporally extended control (Sutton et al., 1999). TD-MPC2 (Hansen et al., 2024) optimizes primitive-action sequences using latent dynamics and a terminal value. ZILOT's goal-conditioned adaptation (Rupf et al., 2025) informs our GC-TD-MPC2 baseline. SkiMo (Shi et al., 2023) and Zhang et al. (2023) use pretrained skills and multi-step models for downstream planning and learning. SPlaTES (Gürtler & Martius, 2025) jointly learns predictable latent skills and an abstract model online. OGPC instead investigates planning over subgoals of a shared goal-conditioned policy learned concurrently with its GHM *tabula rasa*. Planning over learned skill or intention variables (Zheng et al., 2026) using their associated policies and GHMs is a natural extension beyond the scope of this work.

Subgoals for online goal-conditioned learning. Prior works have explored learning intermediate goals and reusable goal-reaching behavior (Schmidhuber, 1990; 1991; Schmidhuber & Wahnsiedler, 1993; Kaelbling, 1993). SoRB (Eysenbach et al., 2019) and L3P (Zhang et al., 2021) search graphs of replay states or learned landmarks, while C-Planning (Zhang et al., 2022) uses a waypoint curriculum during training and removes search at evaluation. Hierarchical RL approaches such as HIRO (Nachum et al., 2018), HRAC (Zhang et al., 2020), and HIGL (Kim et al., 2021) learn high-level subgoal policies. HIGL additionally uses graph search to guide the subgoal policy's training. More recently, DCP (Swaminathan et al., 2026) combines online contrastive learning with visited-state subgoal scoring and a direction-conditioned actor. These approaches motivate our investigation of occupancy-based planning on top of depth-scaled CRL (Eysenbach et al., 2022; Wang et al., 2025). Our HRAC and HIGL adaptations with a CRL policy did not improve over planner-free CRL in the tested settings. Appendix F.7 reports these results alongside a replay-graph baseline.

6 Discussion

OGPC learns a GHM alongside a goal-conditioned policy and uses temporally abstract planning to guide data collection. Across nine JaxGCRL tasks, our results show improvements in sample efficiency and in the performance of the learned policy, evaluated without planning. Terminal conditioning provides a scoring signal when sparse Monte Carlo estimates are dominated by zeros, with its clearest advantages over literal scoring on challenging navigation tasks, including Ant Big Maze, Ant Hardest Maze, and Humanoid U-Maze. The main practical limitation is training cost. Learning the GHM and terminal head, together with repeatedly scoring candidate compositions, makes OGPC substantially more expensive to train than planner-free CRL (Appendix E.7). To reduce this cost, a goal-conditioned high-level policy could learn to select subgoal sequences under iCEM guidance, amortizing the planner's search through search-to-policy learning as in MuZero (Schrittwieser et al., 2020). The learned policy could provide a prior over the iCEM candidate population, analogous to the policy priors in TD-MPC2 (Hansen et al., 2024), or propose subgoal sequences directly without running iCEM. Finally, extending OGPC to pixel-based control could build on latent GHM learning previously explored in offline behavioral cloning (Lawson et al., 2026).

AI USE STATEMENT

In this work, we used LLMs (ChatGPT and Claude) to refine the writing, draft layouts for schematic illustrations (Figures 1(a) and 3) that were later edited by the authors, and support implementation and downstream data analysis. This statement was written by the authors, and minimally polished with LLM assistance. All AI-assisted and AI-generated content was thoroughly reviewed by the authors, who take full responsibility for the accuracy and integrity of the final content presented in the submitted paper.

REPRODUCIBILITY STATEMENT

We provide anonymized source code and instructions to reproduce the main results in the supplementary materials. For every experiment, we specify the hyperparameter settings used. We document the implementation and the experimental setup to facilitate reproducibility in Appendices C and E. We also provide a short snippet of JAX implementation for the two main components of OGPC in Appendix C.5. Within the source code we provide the scripts to reproduce the theoretical bounds and plots reported in the appendix.

REFERENCES

Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michael L. Littman. Combating the compounding-error problem with a multi-step model. arXiv preprint arXiv:1905.13320, 2019. URL https://arxiv.org/abs/1905.13320.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. In *NIPS Deep Learning Symposium*, 2016. URL https://arxiv.org/abs/1607.06450.

David Blackwell. Conditional expectation and unbiased sequential estimation. *The Annals of Mathematical Statistics*, 18(1):105–110, 1947. ISSN 00034851. URL http://www.jstor.org/stable/2236107.

Léonard Blier, Corentin Tallec, and Yann Ollivier. Learning successor states and goal-dependent values: A mathematical viewpoint. *ArXiv*, abs/2101.07123, 2021. URL https://arxiv.org/abs/2101.07123.

Michał Bortkiewicz, Władek Pałucki, Vivek Myers, Tadeusz Dziarmaga, Tomasz Arczewski, Łukasz Kuciński, and Benjamin Eysenbach. Accelerating Goal-Conditioned RL Algorithms and Research. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/pdf/2408.11052.

Wei-Di Chang, Mikael Henaff, Brandon Amos, Gregory Dudek, and Scott Fujimoto. The surprising difficulty of search in model-based reinforcement learning. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026. URL https://arxiv.org/abs/2601.21306.

Sydney Chapman. On the Brownian displacements and thermal diffusion of grains suspended in a non-uniform fluid. *Proceedings of the Royal Society of London. Series A*, 119(781):34–54, 1928.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/hash/69386f6bb1dfed68692a24c8686939b9-Abstract.html.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/3de568f8597b94bda53149c7d7f5958c-Paper.pdf.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Comput.*, 5(4):613–624, July 1993. ISSN 0899-7667. doi: 10.1162/neco.1993.5.4.613. URL https://doi.org/10.1162/neco.1993.5.4.613.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. doi: 10.1016/j.neunet.2017.12.012. URL https://www.sciencedirect.com/science/article/pii/S0893608017302976.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: bridging planning and reinforcement learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc. URL https://proceedings.neurips.cc/paper/2019/hash/5c48ff18e0a47baaf81d8b8ea51eec92-Abstract.html.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 35603–35620. Curran Associates, Inc., 2022. doi: 10.52202/068431-2580. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/e7663e974c4ee7a2b475a4775201ce1f-Paper-Conference.pdf.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Rémi Munos, Alessandro Lazaric, and Ahmed Touati. Temporal difference flows. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Jesse Farebrother, Matteo Pirotta, Andrea Tirinzoni, Marc G. Bellemare, Alessandro Lazaric, and Ahmed Touati. Compositional planning with jumpy world models. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning, 2018. URL https://arxiv.org/abs/1803.00101.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1587–1596. PMLR, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Nico Gürtler and Georg Martius. Long-horizon planning with predictable skills. *Reinforcement Learning Journal*, 6:1247–1272, 2025. URL https://rlj.cs.umass.edu/2025/papers/Paper136.html.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870. PMLR, 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Jessica B. Hamrick, Abram L. Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Buesing, Petar Veličković, and Théophane Weber. On the role of planning in model-based deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=IrM64DGB21.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. URL https://arxiv.org/abs/1606.08415.

Michael Janner, Igor Mordatch, and Sergey Levine. Gamma-models: Generative temporal difference learning for infinite-horizon prediction. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Leslie Pack Kaelbling. Learning to achieve goals. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1993. URL https://people.csail.mit.edu/lpk/publications.html.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 267–274. Morgan Kaufmann, 2002. URL https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/readings/KakadeLangford-icml2002.pdf.

Junsu Kim, Younggyo Seo, and Jinwoo Shin. Landmark-guided subgoal generation in hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/file/ee39e503b6bedf0c98c388b7e8589aca-Paper.pdf.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Andrey Nikolaevich Kolmogorov. Über die analytischen Methoden in der Wahrscheinlichkeitsrechnung. *Mathematische Annalen*, 104:415–458, 1931.

Daniel Lawson, Dhruv Sreenivas, Adriana Hugessen, Khimya Khetarpal, and Glen Berseth. Imagining goals for policy generalization with geometric horizon models. In *Third Reinforcement Learning Beyond Rewards Workshop at RLC*, 2026. URL https://openreview.net/forum?id=rL2Jrd7SRX.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Andreas Maurer and Massimiliano Pontil. Empirical Bernstein bounds and sample variance penalization. In *Proceedings of the 22nd Annual Conference on Learning Theory*, 2009. URL https://www.cs.mcgill.ca/~colt2009/papers/012.pdf.

Diganta Misra. Mish: A self regularized non-monotonic activation function. In *Proceedings of the 31st British Machine Vision Conference*, 2020. doi: 10.5244/C.34.191. URL https://bmva-archive.org.uk/bmvc/2020/conference/papers/paper_0928.html.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/e6384711491713d29bc63fc5eeb5ba4f-Abstract.html.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pp. 3942–3951, 2018. doi: 10.1609/aaai.v32i1.11671. URL https://ojs.aaai.org/index.php/AAAI/article/view/11671.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Jörg Stückler, Michal Rolínek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning*, pp. 1049–1065. PMLR, 2021. URL https://proceedings.mlr.press/v155/pinneri21a.html.

C. Radhakrishna Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, 37(3):81–91, 1945. URL https://www.ias.ac.in/article/fulltext/reso/020/01/0076-0090.

Thomas Rupf, Marco Bagatella, Nico Gürtler, Jonas Frey, and Georg Martius. Zero-shot offline imitation learning via optimal transport. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52345–52381. PMLR, 2025. URL https://proceedings.mlr.press/v267/rupf25a.html.

Jürgen Schmidhuber. Towards compositional learning in dynamic networks. 1990. URL https://api.semanticscholar.org/CorpusID:17783975.

Jürgen Schmidhuber. Learning to generate sub-goals for action sequences. In Teuvo Kohonen, Kai Mäkisara, Olli Simula, and Jari Kangas (eds.), *Artificial Neural Networks*, pp. 967–972. Elsevier Science Publishers, 1991. URL https://mediatum.ub.tum.de/814955.

Jürgen Schmidhuber and Reiner Wahnsiedler. Planning simple trajectories using neural subgoal generators. In Jean-Arcady Meyer, Herbert L. Roitblat, and Stewart W. Wilson (eds.), *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*. The MIT Press, 04 1993. ISBN 9780262287159. doi: 10.7551/mitpress/3116.003.0027. URL https://doi.org/10.7551/mitpress/3116.003.0027.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, and David Silver. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

Haosen Shi, Jianda Chen, and Sinno Jialin Pan. Bridging successor measure and online policy learning with flow matching-based representations. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/48acf4b231771e693f42305b4c9b4c9f-Paper-Conference.pdf.

Lucy Xiaoyang Shi, Joseph J. Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 2262–2272. PMLR, 2023. URL https://proceedings.mlr.press/v205/shi23a.html.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404. URL https://www.science.org/doi/abs/10.1126/science.aar6404.

Richard S. Sutton. TD models: Modeling the world at a mixture of time scales. In *Machine Learning Proceedings 1995*, pp. 531–539. Morgan Kaufmann, 1995. doi: 10.1016/B978-1-55860-377-6.50072-4. URL https://www.sciencedirect.com/science/article/abs/pii/B9781558603776500724.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2):181–211, 1999. doi: 10.1016/S0004-3702(99)00052-1. URL https://www-anw.cs.umass.edu/~barto/courses/cs687/Sutton-Precup-Singh-AIJ99.pdf.

S K Swaminathan, Damiya Gondha, Theyanesh Eswaramoorthy Rajahkrishnan, and Aritra Hazra. Direction-conditioned policies via compositional subgoal scoring for online goal-conditioned reinforcement learning. In *2nd Workshop on Compositional Learning: Safety, Interpretability, and Agents*, 2026. URL https://openreview.net/forum?id=WksFWDQXY6.

Shantanu Thakoor, Mark Rowland, Diana Borsa, Will Dabney, Rémi Munos, and André Barreto. Generalised policy improvement with geometric policy composition. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*. PMLR, 2022.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzciński, and Benjamin Eysenbach. 1000 layer networks for self-supervised RL: Scaling depth can enable new goal-reaching capabilities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=s0JVsx3bx1.

Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. arXiv preprint arXiv:1907.02057, 2019. URL https://arxiv.org/abs/1907.02057.

Jingwei Zhang, Jost Tobias Springenberg, Arunkumar Byravan, Leonard Hasenclever, Abbas Abdolmaleki, Dushyant Rao, Nicolas Heess, and Martin Riedmiller. Leveraging jumpy models for planning and fast learning in robotic domains. arXiv preprint arXiv:2302.12617, 2023. URL https://arxiv.org/abs/2302.12617.

Lunjun Zhang, Ge Yang, and Bradly C. Stadie. World model as a graph: Learning latent landmarks for planning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12611–12620. PMLR, 2021. URL https://proceedings.mlr.press/v139/zhang21x.html.

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. URL https://openreview.net/forum?id=K2JfSnLBD9.

Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/file/f5f3b8d720f34ebebceb7765e447268b-Paper.pdf.

Chongyi Zheng, Seohong Park, Sergey Levine, and Benjamin Eysenbach. Intention-conditioned flow occupancy models. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=j6D83Mf6LG.

APPENDIX

A Nomenclature **17**

A.1 General conventions . . . 17
A.2 Environment, goals, and policies . . . 17
A.3 Successor measures, horizons, and composition . . . 17
A.4 Learned models and parameters . . . 18
A.5 Planning: candidates, scores, and execution . . . 18
A.6 Theoretical analysis . . . 19
A.7 Correspondence with prior notation . . . 19

B Extended Background **20**

B.1 Composition of successor measures . . . 20
B.2 Derivation of the geometric policy composition value . . . 20
B.3 Temporal Difference Flows and horizon consistency . . . 21

C Algorithms and Reference Implementation **23**

C.1 Online loop . . . 23
C.2 Planning at a commitment boundary . . . 24
C.3 Scoring a candidate . . . 26
C.4 Training the terminal head . . . 26
C.5 JAX reference implementation . . . 27

D Theoretical Results **28**

D.1 Formal setting . . . 28
D.2 Proof of terminal Rao–Blackwellization . . . 29
D.3 Variance-equivalent planning samples . . . 29
D.4 Decision signal-to-noise and selection regret . . . 30
D.5 Candidate selection with an approximate head . . . 34
D.6 Conditional result during online training . . . 36
D.7 Theoretical bounds for compositional planning . . . 36

E Experimental Details **40**

E.1 Benchmark and evaluation protocol . . . 40
E.2 Shared online loop and goal-conditioned policy . . . 41
E.3 OGPC . . . 41
E.4 DCP . . . 44
E.5 SF^2 . . . 44
E.6 GC-TD-MPC2 . . . 45
E.7 Compute . . . 46

F Additional Results **48**

F.1 Headline results: Success Rate and final performance . . . 48
F.2 Goal-distance analysis on all navigation tasks . . . 51
F.3 Number of planned subgoals during learning and evaluation . . . 52
F.4 Subgoal-selection ablation . . . 53

F.5 Terminal conditioning: scoring noise and planner decisions 53
F.6 Proposal distribution: goal-conditioned versus unconditional GHM proposals 57
F.7 Alternative subgoal selection during collection 58
F.8 GC-TD-MPC2: policy priors and search budget 60

G Qualitative Planner Rollouts: Subgoal Plans and GHM Proposals **63**
G.1 Ant U-Maze 63
G.2 Ant Big Maze 65
G.3 Ant Hardest Maze 66
G.4 Ant Push 67
G.5 Humanoid 68
G.6 Humanoid U-Maze 69

A NOMENCLATURE

This section lists the notation used throughout the paper, grouped by topic. Appendix A.1 fixes typographic conventions, Appendices A.2–A.5 cover the objects of the main text in the order they are introduced, Appendix A.6 collects the symbols specific to the theoretical analysis, and Appendix A.7 maps our notation onto that of prior work.

A.1 GENERAL CONVENTIONS

Notation	Meaning
S_t, A_t vs. s, a	Uppercase symbols are random variables; lowercase symbols are fixed values.
t, k	Environment timestep and compositional hop index, respectively.
S', A'	Immediate next state and the action drawn there.
S^+, S_k^+, A_k^+	A successor state at a geometric horizon; the successor after k composed jumps; the action drawn at S_k^+ from the next component's policy. $(S_0^+, A_0^+) = (s, a)$.
$\Pr^\pi(\cdot \mid S_0 = s, A_0 = a)$	Probability under P, starting from (s, a) and following π thereafter. The first action is fixed.
$\mathbb{E}_{\widehat{m}}$	Expectation over simulations under the current GHM (Section 3.2).
$\mathbb{1}\{\cdot\}$	Indicator function.
$\mathrm{Geom}(p)$	Geometric distribution on $\{1, 2, \dots\}$ with success probability p.
$x \sim \mathcal{U}[a, b]$	Uniform draw on $[a, b]$.

A.2 ENVIRONMENT, GOALS, AND POLICIES

Notation	Meaning
$\mathcal{M}_\mathcal{G} = (\mathcal{S}, \mathcal{A}, \mathcal{G}, P, r, \gamma)$	Goal-conditioned MDP.
$\mathcal{S}, \mathcal{A}, \mathcal{G}$	State, action, and goal spaces.
$P(\cdot \mid s, a)$	Environment transition kernel.
γ	Discount factor of the goal-conditioned return.
$r_g,\ \mathcal{X}_g$	Binary goal reward and goal region: $r_g(s') = \mathbb{1}\{s' \in \mathcal{X}_g\}$, evaluated at the next state s'.
g, $g^\star$, g_k	Generic goal; commanded task goal, which specifies the evaluation reward; k-th subgoal, which specifies the policy behavior.
$g_{1:K}$	Candidate subgoal sequence.
π_θ, π_g	Goal-conditioned policy with parameters θ, and the policy conditioned on goal g: $\pi_g(\cdot \mid s) = \pi_\theta(\cdot \mid s, g)$. All subgoal-conditioned policies share θ.
$Q^\pi_{\gamma,g}(s, a)$	Action value of π for goal g, $\mathbb{E}^\pi[\sum_{t\geq 0} \gamma^t r_g(S_{t+1})]$, starting from (s, a).

A.3 SUCCESSOR MEASURES, HORIZONS, AND COMPOSITION

Notation	Meaning
$m^\pi_\gamma(\cdot \mid s, a)$	Normalized successor measure of π at discount γ (Equation 1); a probability measure over successor states.
T	Geometric horizon used to sample successor states (Appendix B.1).
K	Number of planned subgoals.
$\nu = \pi_{g_1} \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{K-1}} \pi_{g_K}$	GSP over the subgoal-conditioned policies $\pi_{g_1}, \dots, \pi_{g_K}$.
α_k	Probability of switching from component k to $k+1$ after a step; $\alpha_K = 0$.
β_k	Effective discount of component k: $\beta_k = \gamma(1 - \alpha_k)$, with $\beta_K = \gamma$.
w_k	Weight of the k-th endpoint distribution in the GSP's successor measure (Appendix B.2).
p_k^{sw}	Probability that component k is exited by a switch rather than a halt, $p_k^{\mathrm{sw}} = (\gamma - \beta_k)/(1 - \beta_k)$, so that $w_k = (1 - p_k^{\mathrm{sw}}) \prod_{i<k} p_i^{\mathrm{sw}}$.
$Q^\nu_{\gamma,g^\star}(s, a)$	Value of the composite policy ν for the commanded goal (Equation 6).

Notation	Meaning (continued)
$K_2 \circ_{\pi'} K_1$	Chapman–Kolmogorov composition of two state–action kernels with the intermediate action drawn from π' (Equation 17).

A.4 Learned models and parameters

Notation	Meaning
θ, ϑ, ϕ	Parameters of the policy, the GHM, and the terminal head.
$\bar{\vartheta}$	Target GHM parameters, an exponential moving average of ϑ.
$\widehat{m}_\beta^{\pi_g}(\cdot \mid s, a)$	Learned GHM, a generative approximation of $m_\beta^{\pi_g}$; $\widehat{m}_t$ denotes the GHM available at environment step t.
$\widehat{P}(\cdot \mid s, a)$	Learned one-step transition model used by OGPC-SS (Appendix E.3.4).
β, ω	GHM discount drawn per training example, and the shorter local horizon used by the TD-HC bootstrap ($\omega \leq \beta$).
u, X_0, X_u, ψ_u^ϑ	Flow time in $[0, 1]$, Gaussian noise sample, interpolant at flow time u, and the flow map of the TD-Flow GHM (Appendix B.3).
v_u^ϑ	Velocity field of the TD-Flow GHM.
$\mathcal{L}_{\mathrm{ss}}$, $\mathcal{L}_{\mathrm{bs}}$, $\mathcal{L}_{\mathrm{TDHC}}$	Single-step flow-matching loss, bootstrap loss, and the TD-HC objective (Equations 24–27).
$q_\phi(s, a, g_K, g^\star, \gamma)$	Learned terminal head: predicted probability that the final GHM successor from (s, a) under π_{g_K} lies in $\mathcal{X}_{g^\star}$.
n_{head}, e	Number of terminal heads in the ensemble and the member index, $e = 1, \ldots, n_{\mathrm{head}}$.
$\widetilde{S}_\ell^+$	Independent head-training endpoint drawn from $\widehat{m}_\gamma^{\pi_{g_K}}(\cdot \mid s, a)$. These are repeated samples of one jump, not successive composed hops.
L, ℓ	Number of endpoints per head-training target and their sample index, $\ell = 1, \ldots, L$.
$\widehat{y}$	Soft head-training target, the fraction of the L sampled endpoints in $\mathcal{X}_{g^\star}$ (Equation 81).
Π	Projection from a (predicted) state to the goal space, used to turn GHM samples into subgoal proposals (Appendix C).

A.5 Planning: candidates, scores, and execution

Notation	Meaning
$c = (a, g_{1:K})$, ν_c	Candidate containing an initial action and a subgoal sequence, and its associated GSP.
$c_t^\star$, $a_t^\star$, $g_1^\star$	Candidate selected at step t, and its executed first action and first subgoal.
$(a, g^\star, \ldots, g^\star)$	Direct-goal plan: every subgoal equals the commanded goal.
τ_1	Commitment length in environment steps before replanning (Appendix E.3.3).
$\tau_{\max}$	Maximum commitment length in environment steps.
$R_{<K}$, B_K	Weighted reward collected on the simulated prefix, $R_{<K} = \sum_{k<K} w_k r_{g^\star}(S_k^+)$, and the terminal Bernoulli outcome $B_K = r_{g^\star}(S_K^+)$ (Equation 9).
$\mathcal{F}_{K-1}$	Sigma-field of the simulated prefix $(S_1^+, A_1^+, \ldots, S_{K-1}^+, A_{K-1}^+)$, the input to the final jump.
$q_{\widehat{m}, g^\star}$	Exact conditional probability under the current GHM that the final successor lies in $\mathcal{X}_{g^\star}$ given $\mathcal{F}_{K-1}$.
$Q_{\gamma, g^\star}^{\nu, \widehat{m}}(s, a)$	Expected candidate score under the current composed GHMs; the model score.
$Q_{\gamma, g^\star}^{\nu, \mathrm{env}}(s, a)$	Environment value of ν
$\widehat{Q}^{\nu, E}$, $\overline{Q}_N^{\nu, E}$	Single-simulation score (hat) and its mean over N independent simulations (overbar), for estimator $E \in \{\mathrm{MC}, \mathrm{RB}, \phi\}$: literal Monte Carlo, exact terminal conditioning, or the learned head.
$\overline{Q}_{\gamma, g^\star, N}^{\nu, \mathrm{MC}}(s, a)$	Literal MC candidate score averaged over N independent successor sequences (Equation 7).

Notation	Meaning (continued)
$\overline{Q}_N^{\phi}(c)$	Learned-head candidate score averaged over N independent simulated prefixes (Equation 16).
H_{SS}, $\widetilde{S}_h$	OGPC-SS planning horizon in primitive actions and the state predicted after h one-step transitions, with $\widetilde{S}_0 = s$.
$\widehat{Q}^{\mathrm{SS}}$	OGPC-SS candidate score from one simulated trajectory (Equation 82).
N, n	Number of scoring simulations per candidate and the simulation index, $n = 1, \ldots, N$.
$\mathcal{C} = \{c_i\}_{i=1}^{M}$, M	Candidate population of an iCEM round and its size

A.6 Theoretical analysis

This table collects notation used in the theoretical analysis of Appendix D.

Notation	Meaning
$(\Omega, \mathcal{A}, \mathbb{P}_{\widehat{m}})$	Probability space of one composed simulation under the current GHM (Assumption 2).
$\mathcal{H}_t$	Training history available before planner call t (Appendix D.6).
$\mathcal{C}_{i-1}^{\sigma}$	Sigma-field of all candidates, scores, and optimizer randomness observed before candidate i is scored.
Q_i, $Q_\star$, $\nu_\star$	Model score of candidate i, the population optimum $Q_\star = \max_i Q_i$, and the optimal candidate.
δ_i	Value gap of candidate i, $Q_\star - Q_i$.
$\widehat{\nu}_E$, $\widehat{i}$	Candidate selected by maximizing the empirical score under estimator E, and its index.
$\mathcal{R}_E$	Selection regret of estimator E, $Q_\star - Q_{\widehat{\nu}_E}$.
σ^2_{MC}, σ^2_{RB}, $\sigma^2_{i,E}$	Variance of one simulation score under literal Monte Carlo and exact terminal conditioning; the same for candidate i under estimator E.
Δ_ν	Terminal variance removed by conditioning, $\sigma^2_{\mathrm{MC}} - \sigma^2_{\mathrm{RB}}$.
Δ_i	Variance removed by exact terminal conditioning for candidate i, $\sigma^2_{i,\mathrm{MC}} - \sigma^2_{i,\mathrm{RB}}$.
ϵ_q	Uniform approximation error of the learned head on scored prefixes.
ϵ_{GHM}	Uniform error between model and environment candidate values.
η_N^{RB}, δ	Uniform finite-sample radius of exact terminal-conditioned scores over the population, and the associated failure probability.
$\bar{\tau}_t$	Expected commitment length at environment time t, accounting for the cap and remaining episode horizon (Appendix D.7.1).
n_{trial}	Number of independent trials used to estimate the execution gain and its confidence bound (Appendix D.7.1).
n_{branch}	Number of conditionally independent policy trajectories per selected planning call in an execution-measurement trial (Appendix D.7.1).

A.7 Correspondence with prior notation

The table below compares our notation with the prior formulations. We use the explicit subgoal g_k in place of the policy index z_k. Component-specific switching probabilities give $\beta_k = \gamma(1-\alpha_k)$, with $\alpha_k = \alpha$ in the shared-probability case. For values, all three use the next-state reward convention $\mathbb{E}[\sum_{t\geq 0} \gamma^t r_g(S_{t+1})]$, conditional on the initial state and action.

Quantity	Ours	Thakoor et al. (2022)	Farebrother et al. (2026)
Component policy	π_{g_k}	π_k	π_{z_k}
Successor measure	m^{π}_{β}	μ^{π}_{β}	m^{π}_{β}
Switching probability	α_k	α	α_k
Effective GHM discount	β_k	β	β_k
Composition weight	w_k	—	w_k
Number of components	K	n	n

B Extended Background

We derive the composition of successor measures and the weights used to evaluate a geometric switching policy in Section 2, then describe the objectives used to train our GHM.

B.1 Composition of successor measures

The successor measure $m^{\pi}_{\gamma}(\cdot \mid s, a)$ is the distribution of S_T, where $T \sim \mathrm{Geom}(1-\gamma)$ is drawn independently of the trajectory. The first transition uses a, and subsequent actions follow π. Stopping independently after each transition with probability $1-\gamma$ reproduces the geometric discount weights used to sample a successor state.

Successor measures compose like transition kernels Like P, a successor measure is a kernel from state–action pairs to states. Two such kernels K_1, K_2 compose by integrating over the intermediate state and an action supplied by π':

$$(K_2 \circ_{\pi'} K_1)(\mathcal{X} \mid s, a) := \int_{\mathcal{S}} \int_{\mathcal{A}} K_2(\mathcal{X} \mid s_1, a_1)\, \pi'(\mathrm{d}a_1 \mid s_1)\, K_1(\mathrm{d}s_1 \mid s, a). \tag{17}$$

This is the Chapman-Kolmogorov composition rule (Chapman, 1928; Kolmogorov, 1931). Taking $K_1 = K_2 = P$ gives two primitive transitions. Using successor measures instead gives two geometric-horizon hops (Farebrother et al., 2026, footnote 3). The planner of Section 3 is *jumpy* in this sense: the composition rule remains unchanged, and only the kernel being composed changes.

Repeating this construction gives the sampling recursion in equation 5, starting from $(S^+_0, A^+_0) = (s, a)$. The distribution of S^+_k is obtained by integrating over the preceding endpoints and intermediate actions. Each hop spans T_k transitions with $T_k \sim \mathrm{Geom}(1-\beta_k)$. For $k < K$, we sample $A^+_k \sim \pi_{g_{k+1}}(\cdot \mid S^+_k)$ to initialize the next hop. After the final hop, we evaluate $r_{g^\star}(S^+_K)$, so no A^+_K is required.

B.2 Derivation of the geometric policy composition value

The following construction gives the weights and successor measure of the GSP $\nu = \pi_{g_1} \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{K-1}} \pi_{g_K}$.

Component exits After each transition in component k, the discounted sampling process halts with probability $1-\gamma$, switches to the next component with probability $\gamma\alpha_k$, or continues with the same component with probability $\beta_k = \gamma(1-\alpha_k)$. The final component has $\alpha_K = 0$ and $\beta_K = \gamma$. Consequently, the number of transitions spent in component k has distribution $\mathrm{Geom}(1-\beta_k)$, and its endpoint follows $m^{\pi_{g_k}}_{\beta_k}(\cdot \mid S^+_{k-1}, A^+_{k-1})$. We can sample this endpoint first, then independently decide whether to halt there or switch to the next policy, using the probabilities derived below.

Composition weights Conditional on exiting component k, the probability of switching rather than halting is

$$p^{\mathrm{sw}}_k := \frac{\gamma\alpha_k}{(1-\gamma)+\gamma\alpha_k} = \frac{\gamma-\beta_k}{1-\beta_k}, \qquad 1-p^{\mathrm{sw}}_k = \frac{1-\gamma}{1-\beta_k}, \tag{18}$$

Halting in component k requires switching through the preceding $k-1$ components and then halting:

$$\Pr(\text{halt during component } k) = (1-p^{\mathrm{sw}}_k)\prod_{i=1}^{k-1} p^{\mathrm{sw}}_i = \frac{1-\gamma}{1-\beta_k}\prod_{i=1}^{k-1}\frac{\gamma-\beta_i}{1-\beta_i} = w_k, \tag{19}$$

These are the nonnegative weights of Definition 2.1, which sum to $1-\prod_{k=1}^{K} p^{\mathrm{sw}}_k = 1$ since $p^{\mathrm{sw}}_K = 0$.

Composition value Because each exit type is independent of its exit state, we can sample the full chain in equation 5 before drawing the component in which halting occurs. With the weights w_k derived above, Theorem 1 of Farebrother et al. (2026) expresses the successor measure of the GSP

as a mixture of the endpoint distributions obtained by composing successor kernels according to equation 17:

$$m_\gamma^\nu(\mathcal{X} \mid s,a) = \sum_{k=1}^{K} w_k \Pr(S_k^+ \in \mathcal{X} \mid s,a). \tag{20}$$

The k-th mixture component is the distribution after the first k composed hops. Applying the goal-value identity in equation 2 yields

$$\begin{aligned} Q_{\gamma,g^\star}^\nu(s,a) &= \frac{1}{1-\gamma}\, m_\gamma^\nu(\mathcal{X}_{g^\star} \mid s,a) \\ &= \frac{1}{1-\gamma} \sum_{k=1}^{K} w_k \Pr(S_k^+ \in \mathcal{X}_{g^\star} \mid s,a) = \frac{1}{1-\gamma} \sum_{k=1}^{K} w_k \,\mathbb{E}\big[r_{g^\star}(S_k^+) \mid s,a\big], \end{aligned} \tag{21}$$

recovering Equation 6.

B.3 Temporal Difference Flows and horizon consistency

OGPC requires predictions under different subgoal-conditioned policies and at the discounts assigned to their geometric horizons. We learn these distributions with a single goal- and discount-conditioned TD-Flow (Farebrother et al., 2025), trained from replay using the TD-HC objective (Farebrother et al., 2026). Here g denotes the policy command and β the GHM discount. For the k-th subgoal in a candidate plan, the planner supplies $g = g_k$ and $\beta = \beta_k$.

Flow parameterization A TD-Flow represents the GHM $\widehat{m}_\beta^{\pi_g}(\cdot \mid s,a)$ as a continuous normalizing flow (Chen et al., 2018), trained using flow matching (Lipman et al., 2023). Its neural velocity field $v_u^\vartheta(x \mid s,a,g,\beta) \in \mathbb{R}^d$ specifies how a Gaussian sample $X_0 \sim p_0 = \mathcal{N}(0,I)$ is transformed into a predicted successor state S^+. We use ϑ for the model parameters and $\bar{\vartheta}$ for the target parameters. The target parameters are initialized to ϑ and updated by an exponential moving average of the model parameters during training. The transformation is described by the flow map $\psi_u^\vartheta(X_0 \mid s,a,g,\beta)$, which satisfies the following ordinary differential equation (ODE), written in integral form:

$$\begin{aligned} \psi_u^\vartheta(X_0 \mid s,a,g,\beta) &= X_0 + \int_0^u v_{u'}^\vartheta\big(\psi_{u'}^\vartheta(X_0 \mid s,a,g,\beta) \mid s,a,g,\beta\big)\,\mathrm{d}u', \\ S^+ &:= \psi_1^\vartheta(X_0 \mid s,a,g,\beta) \sim \widehat{m}_\beta^{\pi_g}(\cdot \mid s,a). \end{aligned} \tag{22}$$

Integrating over flow time from $u = 0$ to $u = 1$ produces one GHM successor-state sample. We approximate the integral using Euler integration steps (Table 10). Given a target sample x_1, conditional flow matching regresses v_u^ϑ onto the straight-line velocity $x_1 - X_0$ at the interpolant $X_u = (1-u)X_0 + u\,x_1$.

TD-Flow objective Following the Bellman mixture in equation 3, TD-Flow combines flow matching on replay successors S' with bootstrapping from the target flow. For a replay transition (s,a,S'), command g, and discount β, the loss is

$$\mathcal{L}_{\mathrm{TD}}(\vartheta;\bar{\vartheta}) = (1-\beta)\,\mathcal{L}_{\mathrm{ss}}(\vartheta) + \beta\,\mathcal{L}_{\mathrm{bs}}(\vartheta;\bar{\vartheta}), \tag{23}$$

$$\mathcal{L}_{\mathrm{ss}}(\vartheta) = \mathbb{E}_{u,X_0}\Big[\big\|v_u^\vartheta(X_u \mid s,a,g,\beta) - (S' - X_0)\big\|^2\Big], \tag{24}$$

$$\mathcal{L}_{\mathrm{bs}}(\vartheta;\bar{\vartheta}) = \mathbb{E}_{u,X_0,A'}\Big[\big\|v_u^\vartheta(\bar{X}_u \mid s,a,g,\beta) - v_u^{\bar{\vartheta}}(\bar{X}_u \mid S',A',g,\beta)\big\|^2\Big], \tag{25}$$

where $u \sim \mathcal{U}[0,1]$, $X_0 \sim p_0$, and $A' \sim \pi_g(\cdot \mid S')$. The single-step loss in equation 24 matches the velocity along a straight path to the observed next state S', evaluated at $X_u = (1-u)X_0 + u\,S'$, while the bootstrap loss in equation 25 matches the target flow's velocity at $\bar{X}_u = \psi_u^{\bar{\vartheta}}(X_0 \mid S',A',g,\beta)$, a point generated from (S',A'). Training averages these per-example losses over replay transitions and sampled commands and discounts.

Horizon consistency TD-HC uses the GHM's predictions at a shorter local horizon, with discount $\omega < \beta$, to train its predictions at discount β through a Bellman-like consistency relation. The successor measures satisfy the identity (Janner et al., 2020):

$$m_{\beta}^{\pi_g}(\cdot \mid s,a) = \frac{1-\beta}{1-\omega}\, m_{\omega}^{\pi_g}(\cdot \mid s,a) + \frac{\beta-\omega}{1-\omega}\, \mathbb{E}_{S^+\sim m_{\omega}^{\pi_g}(\cdot|s,a),\, A^+\sim\pi_g(\cdot|S^+)}\big[m_{\beta}^{\pi_g}(\cdot \mid S^+,A^+)\big]. \tag{26}$$

Applying equation 26 to the successor measure conditioned on (S',A') in equation 3 splits the bootstrap contribution into two terms, giving

$$\mathcal{L}_{\mathrm{TDHC}}(\vartheta;\bar{\vartheta}) = (1-\beta)\,\mathcal{L}_{\mathrm{ss}}(\vartheta) + \beta\,\frac{1-\beta}{1-\omega}\,\mathcal{L}_{\mathrm{bs}}^{\omega}(\vartheta;\bar{\vartheta}) + \beta\,\frac{\beta-\omega}{1-\omega}\,\mathcal{L}_{\mathrm{bs}}^{\beta,+}(\vartheta;\bar{\vartheta}). \tag{27}$$

In $\mathcal{L}_{\mathrm{bs}}^{\omega}$, the discount input to both the target flow and the target velocity in equation 25 is changed from β to ω. The learned velocity field remains conditioned on (s,a,g,β). For $\mathcal{L}_{\mathrm{bs}}^{\beta,+}$, we first sample S^+ from the target GHM at discount ω, conditioned on (S',A'), then sample $A^+\sim\pi_g(\cdot \mid S^+)$. Equation 25 is then applied with the target flow conditioned on (S^+,A^+) at discount β. Setting $\omega=\beta$ recovers the TD-Flow objective in equation 23. Sampling and masking details are given in Appendix E.3.1.

C Algorithms and Reference Implementation

This appendix gives the complete OGPC procedure as pseudocode and a compact JAX reference implementation of its two novel components: the candidate scorer (Algorithm 6) and the training of the terminal head q_ϕ (Algorithm 7). Lines that differ between OGPC-TC and OGPC-MC are colored as in the figures; the two variants differ in one line of the scorer and in whether the head is trained.

Notation follows Appendix A. Helper operations are defined alongside the algorithms that use them, and hyperparameters are collected in Appendix E.

C.1 Online loop

Algorithm 2 expands Algorithm 1 for one environment. The implementation runs 1,024 environments in parallel, each with its own commitment counter and elite cache. For the default composition, $\bar{\beta} := \beta_1 = \cdots = \beta_{K-1}$ is the execution discount, while the final model hop uses $\beta_K = \gamma$.

Algorithm 2 Online Geometric Policy Composition (OGPC)

Inputs: estimator $\mathsf{E} \in \{\mathsf{TC}, \mathsf{MC}\}$; budget T; warm-up W; block length H; discount $\bar{\beta}$ and cap tolerance ϵ_{cap}; randomly initialized policy π_θ, GHM $\widehat{m}_\vartheta$, and (TC only) head q_ϕ.
Replay $\mathcal{D} \leftarrow \varnothing$; $\tau_{\max} \leftarrow \lceil \log \epsilon_{\text{cap}} / \log \bar{\beta} \rceil$; commitment $\ell \leftarrow 0$; elite cache $\mathcal{C} \leftarrow \varnothing$
Reset the environment and observe the state s and the task goal $g^\star$
for $t = 1, \ldots, T$ **do**
  `# Act with the policy during warm-up, then follow the planner's command`
  if $t \le W$ **then**
    $g_{\text{cmd}} \leftarrow g^\star$; $a \sim \pi_\theta(\cdot \mid s, g^\star)$
  else
    if $\ell = 0$ **then** `# commitment expired: replan`
      $(a^\star, g^\star_{1:K}, \mathsf{keep_seed}, \mathcal{C}) \leftarrow \textsc{Plan}(s, g^\star, \mathcal{C})$ `# Algorithm 3`
      $g_{\text{cmd}} \leftarrow g^\star_1$; $\ell \sim \min\{\text{Geom}(1-\bar{\beta}), \tau_{\max}\}$; first $\leftarrow$ true
    end if
    $a \sim \pi_\theta(\cdot \mid s, g_{\text{cmd}})$
    if first **and not** keep_seed **then** $a \leftarrow a^\star$ `# planned first action, executed once`
    first $\leftarrow$ false; $\ell \leftarrow \ell - 1$
  end if

  `# Interact, store the transition with the task goal and the command`
  Execute a; observe s' and the episode flags; append $(s, a, s', g^\star, g_{\text{cmd}})$ to $\mathcal{D}$; $s \leftarrow s'$
  if the episode ended **then** reset the environment and observe $(s, g^\star)$; $\ell \leftarrow 0$; $\mathcal{C} \leftarrow \varnothing$

  `# Learn once per collection block, every component from the same replay`
  if $t \bmod H = 0$ **and** $|\mathcal{D}| \ge$ minimum replay size **then**
    Update π_θ on hindsight-relabeled goals
    Update $\widehat{m}_\vartheta$ and bootstrap through the current π_θ `# Appendix B.3`
    if $\mathsf{E} = \mathsf{TC}$ **then** $\phi \leftarrow \textsc{FitTerminalHead}(\mathcal{D}, \widehat{m}_\vartheta, \phi)$ `# Algorithm 7`
  end if
end for
return π_θ `# evaluated on` $g^\star$ `without the planner`

Commitment boundaries. The commitment length includes the first action. The planner is called again when this commitment expires or the episode resets. Note that reaching $g^\star_1$ does not terminate the commitment.

Executing the selected candidate. If the original direct-goal seed c^0 remains selected (keep_seed $=$ true), the policy samples a new action toward $g^\star$ at every step, including the first. Every other candidate executes its stored $a^\star$ once and then follows $\pi_\theta(\cdot \mid s, g^\star_1)$.

C.2 PLANNING AT A COMMITMENT BOUNDARY

Candidate coordinates. The optimizer represents $c = (a, g_{1:K})$ by a normalized vector $y \in \mathbb{R}^D$, where $D = \dim(\mathcal{A}) + K \dim(\mathcal{G})$. Encode concatenates the action and subgoals and rescales each coordinate from its prescribed bounds to $[-1, 1]$. Decode remaps the normalized coordinates back to their original coordinates, clips them to the bounds specified in Table 12, and returns $(a, g_{1:K})$. Within Algorithm 3, $\mathcal{C}$ stores the n_{elite} elite vectors retained from the preceding round or planner call.

Algorithm 3 PLAN: iCEM over action–subgoal compositions $c = (a, g_{1:K})$

Inputs: state s, task goal $g^\star$, elite cache $\mathcal{C}$; population P, iterations J, elites n_{elite}, GHM proposals n_G, momentum α, initial variance σ_0^2; current π_θ, $\widehat{m}_\vartheta$, and q_ϕ.
function PLAN($s, g^\star, \mathcal{C}$)
  `# Construct the direct-goal plan and` n_G `GHM chains`
  $c^0 \leftarrow (a,\ g^\star, \dots, g^\star)$ with $a \sim \pi_\theta(\cdot \mid s, g^\star)$
  $c_i^{\text{ghm}} \leftarrow$ GHMPROPOSAL(s) for $i = 1, \dots, n_G$ `# Algorithm 4`
  Encode each seed as $y = \text{Encode}(c)$
  $\mu \leftarrow \mathbf{0}$; $\sigma^2 \leftarrow \sigma_0^2 \mathbf{1}$; $Q^\star \leftarrow -\infty$; warm $\leftarrow [\mathcal{C} \neq \varnothing]$ `# fixed at entry`
  for $j = 1, \dots, J$ **do**

    `# Sample independent noise for` a`, correlated noise across subgoals` $g_{1:K}$
    Draw independent colored-noise vectors $\xi_{1:P}$ with unit marginal variance
    $y_i \leftarrow \text{clip}_{[-1,1]}(\mu + \sigma \odot \xi_i)$ for $i = 1, \dots, P$

    `# Injections overwrite population slots; every candidate is scored once`
    if $j = 1$ **then** slots $1, \dots, 1{+}n_G \leftarrow$ the seeds
    if $j = 1$ **and** warm **then** next n_{elite} slots $\leftarrow$ elites of $\mathcal{C}$ shifted one hop ($g_{2:K} \to g_{1:K-1}$)
    if $1 < j < J$ **and** warm **then** slots $1, \dots, n_{\text{elite}} \leftarrow$ current elites, unshifted
    if $j = J$ **and** $J > 1$ **then** slot $1 \leftarrow \mu$ `# the refined mean is scored, not assumed`

    `# Score each candidate using independent simulations`
    `# Keep track of the best candidate seen thus far`
    $Q_i \leftarrow$ SCORE($s, g^\star, \text{Decode}(y_i)$) for $i = 1, \dots, P$ `# Algorithm 6`
    $\mathcal{C} \leftarrow$ the n_{elite} highest-scoring y_i
    $i^\star \leftarrow \min \arg\max_i Q_i$ `# earlier slot on ties`
    if $Q_{i^\star} > Q^\star$ **then**
      $(y^\star, Q^\star) \leftarrow (y_{i^\star}, Q_{i^\star})$
      keep_seed $\leftarrow$ [the selected entry is the original seed c^0]
    end if
    $\mu \leftarrow \alpha\mu + (1 - \alpha)\,\text{mean}(\mathcal{C})$; $\sigma^2 \leftarrow \alpha\sigma^2 + (1 - \alpha)\,\text{var}(\mathcal{C})$
  end for
  return $\text{Decode}(y^\star) = (a^\star, g^\star_{1:K})$, keep_seed, $\mathcal{C}$
end function

Reusing candidates. At the next commitment boundary, a cached elite supplies $g_{2:K}$ for the new plan's first $K - 1$ subgoals. Its initial action and final subgoal are newly sampled. Within a planner call, reinjection instead copies the whole candidate. Both forms of reuse are scored again with independent simulations.

Selection and ties. The planner returns the candidate with the highest observed score across all rounds. Ties within a round favor the earlier slot, and a later round replaces the stored candidate only on a strict score improvement. Because the original direct-goal seed is scored first, an all-zero search retains it with keep_seed $=$ true. A reused copy is scored as a separate candidate. If that copy is selected, keep_seed $=$ false and the policy executes its stored first action.

Implementation choices and source code. Our planner uses iCEM (Pinneri et al., 2021) and its source implementation.[3] Our shared optimizer follows the `mbrl-lib` [4] adaptation distributed with ZILOT (Rupf et al., 2025),[5] including variance smoothing, in-population elite replacement, and retention of the highest-scoring candidate across rounds.

Seeds and reused plans replace population slots, keeping the budget at $12 \times 4 = 48$ candidate evaluations. The search mean and variance reset at each call. Unlike the original implementation, elite reinjection requires a nonempty cache at call entry, so the first call after a reset has no reinjection.

Proposal distributions. The default proposal (Algorithm 4) follows Algorithm 4 of Farebrother et al. (2026). $\widehat{m}^{\varnothing}_{\beta;\vartheta}$ denotes the GHM with masked action and command inputs. It is trained from the replay transitions, using the stored next actions for bootstrapping (Appendix E.3.1). We apply the projection Π to each state sampled from the GHM to extract the goal coordinates that form the subgoals.

The alternative in Algorithm 5 follows Algorithm 3 of Farebrother et al. (2026). It conditions each jump on an action from $\pi_{g^\star}$ and on the task goal, so the proposed chain describes the model's predicted progress toward that goal. The proposal ablation is reported in Appendix F.6. Note that the default experiments use the unconditional proposal. Appendix G compares both proposal distributions (Figure 36) and shows rollouts planned with goal-conditioned proposals.

Algorithm 4 GHMPROPOSAL: unconditional GHM chain proposal

Inputs: state s, discounts $\beta_{1:K}$; current unconditional GHM $\widehat{m}^{\varnothing}_{\beta;\vartheta}$ and policy π_θ.
function GHMPROPOSAL(s)
    `# Chain GHM samples and project each one onto the goal coordinates`
    $\widetilde{S}_0 \leftarrow s$
    for $k = 1, \dots, K$ **do**
        $\widetilde{S}_k \sim \widehat{m}^{\varnothing}_{\beta_k;\vartheta}(\cdot \mid \widetilde{S}_{k-1})$; $g_k \leftarrow \Pi(\widetilde{S}_k)$
    end for
    $a \sim \pi_\theta(\cdot \mid s, g_1)$
    return $c = (a, g_{1:K})$
end function

Algorithm 5 GHMPROPOSAL-GC: goal-conditioned GHM chain proposal

Inputs: state s, task goal $g^\star$, discounts $\beta_{1:K}$; current GHM $\widehat{m}_{\beta;\vartheta}$ and policy π_θ.
function GHMPROPOSAL-GC($s, g^\star$)
    `# Chain the policy's own predicted progress toward` $g^\star$
    $\widetilde{S}_0 \leftarrow s$
    for $k = 1, \dots, K$ **do**
        $\widetilde{A}_{k-1} \sim \pi_\theta(\cdot \mid \widetilde{S}_{k-1}, g^\star)$
        $\widetilde{S}_k \sim \widehat{m}^{\pi_{g^\star}}_{\beta_k;\vartheta}(\cdot \mid \widetilde{S}_{k-1}, \widetilde{A}_{k-1})$; $g_k \leftarrow \Pi(\widetilde{S}_k)$
    end for
    $a \sim \pi_\theta(\cdot \mid s, g_1)$
    return $c = (a, g_{1:K})$
end function

[3] `https://github.com/martius-lab/iCEM/blob/main/icem/controllers/icem.py`
[4] `https://github.com/facebookresearch/mbrl-lib`
[5] ZILOT's `ICEMOptimizer`, revision `feee9af`.

C.3 SCORING A CANDIDATE

Algorithm 6 SCORE: scoring one candidate with the Monte Carlo (MC) or the Terminal Conditional (TC) estimator

Inputs: state s, task goal $g^\star$, candidate $c = (a, g_{1:K})$, estimator E, simulations N, discounts $\beta_{1:K}$ with $\beta_K = \gamma$, weights $w_{1:K}$; current π_θ, $\widehat{m}_\vartheta$, and q_ϕ.
function SCORE($s, g^\star, c$)
  for $n = 1, \dots, N$ independently **do**
    $(S_0^+, A_0^+) \leftarrow (s, a)$; $R \leftarrow 0$

    # Use the GHM to simulate the first $K-1$ jumps under the planned subgoals
    for $k = 1, \dots, K-1$ **do**
      $S_k^+ \sim \widehat{m}^{\pi_{g_k}}_{\beta_k;\vartheta}(\cdot \mid S_{k-1}^+, A_{k-1}^+)$
      $R \leftarrow R + w_k\, r_{g^\star}(S_k^+)$
      $A_k^+ \sim \pi_\theta(\cdot \mid S_k^+, g_{k+1})$ # the action that starts the next hop
    end for

    # Use the sampled reward (MC) or the terminal head's prediction (TC)
    if $\mathsf{E} = \mathsf{MC}$ **then** $S_K^+ \sim \widehat{m}^{\pi_{g_K}}_{\gamma;\vartheta}(\cdot \mid S_{K-1}^+, A_{K-1}^+)$; $u \leftarrow r_{g^\star}(S_K^+)$ # Eq. (11)
    if $\mathsf{E} = \mathsf{TC}$ **then** $u \leftarrow q_\phi(S_{K-1}^+, A_{K-1}^+, g_K, g^\star, \gamma)$ # Eq. (16)
    $\widehat{Q}^{(n)} \leftarrow (1-\gamma)^{-1}\big(R + w_K\, u\big)$
  end for
  return $\frac{1}{N}\sum_{n=1}^{N} \widehat{Q}^{(n)}$
end function

Scoring cost. Both estimators simulate the same prefix distribution. For every candidate, MC makes NK GHM draws, while TC makes $N(K-1)$ GHM draws and N head evaluations. The head is queried at the final-hop discount $\beta_K = \gamma$, matching its training targets.

C.4 TRAINING THE TERMINAL HEAD

Algorithm 7 constructs a minibatch of the soft targets defined in Appendix E.3.2. We use b to index replay examples and ℓ to index independent endpoints for each example. During training, the sampled command g_b is given as the head's policy-command input. During planning, this input is set to be the candidate's final subgoal g_K.

Algorithm 7 FITTERMINALHEAD: online distillation of the GHM's success probability (TC only)

Inputs: replay $\mathcal{D}$, current task-goal batch $\mathcal{G}_{\text{batch}}$, GHM $\widehat{m}_\vartheta$, head ensemble $\phi = (\phi_1, \dots, \phi_{n_{\text{head}}})$; batch size B, endpoints per target L, hindsight fraction p_{h}.
function FITTERMINALHEAD($\mathcal{D}, \widehat{m}_\vartheta, \phi$)
  for $b = 1, \dots, B$ **do**
    Sample a transition $(s_b, a_b, g_b^{\text{cmd}}) \sim \mathcal{D}$ and, independently, a task goal $g_b^\star \sim \text{Unif}(\mathcal{G}_{\text{batch}})$
    # Command: the executed subgoal, or a hindsight subgoal reached later
    $\Delta \sim \text{Geom}(1-\bar{\beta})$; $g_b \leftarrow \Pi(\text{state reached } \Delta \text{ steps after } s_b)$ w.p. p_{h}, else $g_b \leftarrow g_b^{\text{cmd}}$
    # Soft target: the current GHM's own success frequency
    $\widetilde{S}_{b\ell}^+ \sim \widehat{m}^{\pi_{g_b}}_{\gamma;\vartheta}(\cdot \mid s_b, a_b)$ for $\ell = 1, \dots, L$; $\widehat{y}_b \leftarrow \frac{1}{L}\sum_{\ell=1}^{L} r_{g_b^\star}(\widetilde{S}_{b\ell}^+)$
  end for
  $\mathcal{L}(\phi) \leftarrow \frac{1}{B n_{\text{head}}} \sum_{b=1}^{B} \sum_{e=1}^{n_{\text{head}}} \text{BCE}\big(q_{\phi_e}(s_b, a_b, g_b, g_b^\star, \gamma), \widehat{y}_b\big)$ # Section 3.2
  $\phi \leftarrow \text{Adam}(\phi, \nabla_\phi \mathcal{L})$ # gradient norm clipped
  return ϕ # planner reads $q_\phi = \frac{1}{n_{\text{head}}}\sum_{e=1}^{n_{\text{head}}} q_{\phi_e}$
end function

What the target estimates. The task goal is sampled independently of the replay state–action pair and the model endpoints. The target estimates success under the current GHM. All ensemble members fit the same target from separate random initializations. Here the gradients update only the head, not the GHM that generated these targets. Full architecture and update settings are described in Table 11.

C.5 JAX REFERENCE IMPLEMENTATION

Algorithm 8 JAX reference implementation of SCORE (both estimators) and the terminal-head loss

```
import jax, jax.numpy as jnp
from optax import sigmoid_binary_cross_entropy
# ghm.sample(s, a, g, beta, key)  -> S+ ~ m_beta^{pi_g}(.|s, a)    (one GHM draw)
# policy.sample(s, g, key)        -> a ~ pi_theta(.|s, g)          (goal-conditioned policy)
# head.logits(phi, s, a, g_K, g_star, beta) -> (n_head, ...) ensemble logits of g_phi
# reward(s, g_star)               -> 1{||proj(s) - g_star|| < eps} (sparse success)
N, L = 64, 16                     # simulations per candidate, endpoints per head target

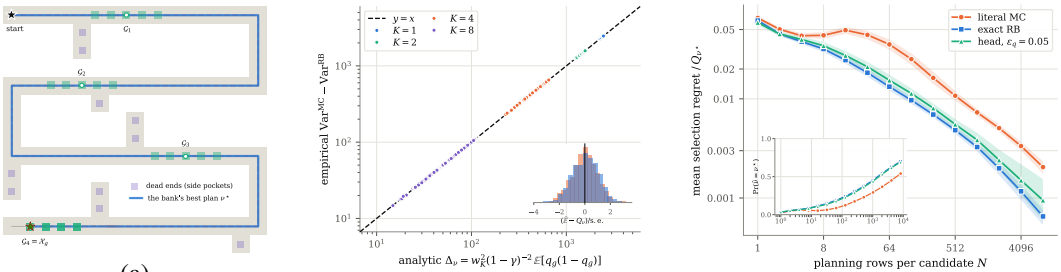
def composition_weights(betas, gamma):              # betas: (K,), betas[-1] == gamma
  ratios = (gamma - betas) / (1.0 - betas)
  prefix = jnp.concatenate([jnp.ones(1), jnp.cumprod(ratios)[:-1]])
  return (1.0 - gamma) / (1.0 - betas) * prefix     # w_k, empty product = 1

def score_candidate(s, a, subgoals, g_star, betas, gamma, phi, key,
                    terminal_conditioning):
  """Mean of N composed GHM simulations for one candidate c = (a, g_{1:K})."""
  K, w = subgoals.shape[0], composition_weights(betas, gamma)
  def one_simulation(key):
    keys = jax.random.split(key, K)
    def hop(carry, inp):                                # hops k = 1, ..., K-1
      s_prev, a_prev, R = carry
      g_k, g_next, beta_k, w_k, k = inp
      k_ghm, k_pi = jax.random.split(k)
      s_k = ghm.sample(s_prev, a_prev, g_k, beta_k, k_ghm)  # S_k+ under policy pi_{g_k}
      a_k = policy.sample(s_k, g_next, k_pi)               # A_k+ ~ pi_theta(.|S_k+, g_{k+1})
      return (s_k, a_k, R + w_k * reward(s_k, g_star)), None
    (s_prev, a_prev, R), _ = jax.lax.scan(hop, (s, a, 0.0),
        (subgoals[:-1], subgoals[1:], betas[:-1], w[:-1], keys[:-1]))
    if terminal_conditioning:                           # OGPC-TC: E[r_K | prefix] from the head
      logits = head.logits(phi, s_prev, a_prev, subgoals[-1], g_star, gamma)   # (n_head,)
      u = jax.nn.sigmoid(logits).mean(0)                # ensemble-mean success probability
    else:                                               # OGPC-MC: sample the final endpoint
      s_K = ghm.sample(s_prev, a_prev, subgoals[-1], betas[-1], keys[-1])
      u = reward(s_K, g_star)
    return (R + w[-1] * u) / (1.0 - gamma)

  return jax.vmap(one_simulation)(jax.random.split(key, N)).mean()   # (N,) -> scalar

def terminal_head_loss(phi, s, a, g_K, g_star, gamma, key):
  """Soft-target BCE against the GHM's success frequency"""
  keys = jax.random.split(key, L)
  endpoints = jax.vmap(lambda k: ghm.sample(s, a, g_K, gamma, k))(keys)   # (L, B, S)
  y_hat = jax.vmap(reward, in_axes=(0, None))(endpoints, g_star).mean(0)  # (B,) hits
  logits = head.logits(phi, s, a, g_K, g_star, gamma)                      # (n_head, B)
  return sigmoid_binary_cross_entropy(logits, jax.lax.stop_gradient(y_hat)).mean()
```

Scope of the code listing. The four interfaces specified in the opening comments correspond to the GHM, policy, head, and reward function. `score_candidate` implements Algorithm 6, using `jax.lax.scan` for the successive hops and `jax.vmap` for independent simulations. `terminal_head_loss` implements endpoint sampling and the loss for an already sampled replay minibatch. Replay relabeling and the optimizer step remain in Algorithm 7.



(a) (b) (c)

Figure 8: Numerical verification over 30 tabular instances. **(a) Snake Maze**: Grid environment in which the agent must navigate a snake-shaped maze to reach the final goal while avoiding dead ends. **(b) Variance Gap**: Theoretical and empirical variance gaps between the MC and RB estimators, as predicted by Thm. 1, for various K (median relative error $9.1\mathrm{e}{-4}$). The inset depicts the empirical (unbiased) distributions of their Q-values. **(c) Mean Regret**: For a fixed candidate set, exact RB has lower mean selection regret than literal MC across planning budgets N; a learned head with $\epsilon_q \approx 0.05$ retains most of this advantage. The inset reports the probability of selecting the candidate set-optimal composition $\nu^\star$.

D Theoretical Results

Why sparse planning needs conditional structure: An unbiased planning score can still be uninformative at a limited simulation budget. With sparse goal rewards, many composed simulations receive zero reward even when the candidate has positive expected value, leaving different plans tied and difficult to rank. When success requires several successive transitions through bottlenecks, their combined success probability can become very small as the chain grows, so observing a positive terminal signal may require many simulations. Evaluating a plan monolithically does not by itself remove this difficulty if its score still relies on the same rare terminal hit-or-miss outcome. Conditional information offers a way to extract more signal from each simulation: a sampled intermediate state can support an estimate of the remaining goal occupancy without requiring a final sampled hit. OGPC uses this idea at the last compositional hop, replacing the terminal indicator by its conditional expectation. This preserves the current-GHM objective while reducing estimation noise, as established by Theorem 1. It addresses terminal sampling uncertainty, although reaching informative prefixes and accurately learning their conditional occupancies remain necessary.

All expectations and variances below are relative to the current GHM unless an environment superscript is shown explicitly. We will use $E \in \{\mathrm{MC}, \mathrm{RB}\}$ to refer to either of the two estimators.

D.1 Formal setting

Assumption 2 (One-row simulation law)**.** *For a fixed query $(s, a, g^\star)$, candidate ν, and current GHM $\widehat{m}$, one composed simulation row is defined on a probability space $(\Omega, \mathcal{A}, \mathbb{P}_{\widehat{m}})$. The sigma-field $\mathcal{F}_{K-1} \subseteq \mathcal{A}$ contains the simulated prefix through the input to the final jump. The variable $R_{<K}$ in Equation 9 is $\mathcal{F}_{K-1}$-measurable, $B_K \in \{0, 1\}$, and a version of $q_{\widehat{m},g^\star} = \mathbb{E}_{\widehat{m}}[B_K \mid \mathcal{F}_{K-1}]$ exists.*

Assumption 3 (Fresh evaluation rows)**.** *For a fixed candidate, its N evaluation rows are independent and identically distributed under the current GHM. For a sequential candidate set, let $\mathcal{C}^\sigma_{i-1}$ contain all candidates, scores, and optimizer randomness observed before candidate i is scored. Candidate i is $\mathcal{C}^\sigma_{i-1}$-measurable and, conditional on $\mathcal{C}^\sigma_{i-1}$, its evaluation rows are fresh and conditionally independent with the row law induced by that candidate and the current GHM.*

The second assumption covers candidates proposed adaptively across iCEM rounds; it requires the randomness used to score a candidate to be fresh conditional on the history that proposed it.

D.2 Proof of terminal Rao–Blackwellization

Proof of Theorem 1. Since $R_{<K}$ is $\mathcal{F}_{K-1}$-measurable and $q_{\widehat{m},g^\star} = \mathbb{E}_{\widehat{m}}[B_K \mid \mathcal{F}_{K-1}]$,

$$\mathbb{E}_{\widehat{m}}\left[\widehat{Q}^{\nu,\mathrm{MC}}_{\gamma,g^\star}(s,a) \mid \mathcal{F}_{K-1}\right] = \frac{1}{1-\gamma}\mathbb{E}_{\widehat{m}}[R_{<K} + w_K B_K \mid \mathcal{F}_{K-1}] \tag{28}$$

$$= \frac{1}{1-\gamma}\left(R_{<K} + w_K q_{\widehat{m},g^\star}\right) = \widehat{Q}^{\nu,\mathrm{RB}}_{\gamma,g^\star}(s,a). \tag{29}$$

This proves Equation 13. Taking expectations and using the tower property proves Equation 14. For the variance claim, the law of total variance gives

$$\mathrm{Var}_{\widehat{m}}\left(\widehat{Q}^{\nu,\mathrm{MC}}_{\gamma,g^\star}\right) = \mathrm{Var}_{\widehat{m}}\left(\mathbb{E}_{\widehat{m}}[\widehat{Q}^{\nu,\mathrm{MC}}_{\gamma,g^\star} \mid \mathcal{F}_{K-1}]\right) \tag{30}$$

$$+ \mathbb{E}_{\widehat{m}}\left[\mathrm{Var}_{\widehat{m}}\left(\widehat{Q}^{\nu,\mathrm{MC}}_{\gamma,g^\star} \mid \mathcal{F}_{K-1}\right)\right] \tag{31}$$

$$= \mathrm{Var}_{\widehat{m}}\left(\widehat{Q}^{\nu,\mathrm{RB}}_{\gamma,g^\star}\right) + \frac{w_K^2}{(1-\gamma)^2}\mathbb{E}_{\widehat{m}}[\mathrm{Var}_{\widehat{m}}(B_K \mid \mathcal{F}_{K-1})]\,. \tag{32}$$

Conditionally, B_K is Bernoulli with success probability $q_{\widehat{m},g^\star}$, hence $\mathrm{Var}_{\widehat{m}}(B_K \mid \mathcal{F}_{K-1}) = q_{\widehat{m},g^\star}(1 - q_{\widehat{m},g^\star})$. Substitution proves Equation 15. The integrand is nonnegative and is strictly positive on $\{0 < q_{\widehat{m},g^\star} < 1\}$, proving the strictness statement. □

Remark 4 (Nonbinary terminal rewards). If $B_K \in [0,1]$ is not Bernoulli, the same argument gives the exact gap

$$\frac{w_K^2}{(1-\gamma)^2}\mathbb{E}_{\widehat{m}}[\mathrm{Var}_{\widehat{m}}(B_K \mid \mathcal{F}_{K-1})]\,. \tag{33}$$

The expression $q(1-q)$ is specific to binary terminal rewards.

D.3 Variance-equivalent planning samples

Define

$$\sigma^2_{\mathrm{MC}} := \mathrm{Var}_{\widehat{m}}(\widehat{Q}^{\nu,\mathrm{MC}}_{\gamma,g^\star}), \quad \sigma^2_{\mathrm{RB}} := \mathrm{Var}_{\widehat{m}}(\widehat{Q}^{\nu,\mathrm{RB}}_{\gamma,g^\star}), \quad \Delta_\nu := \frac{w_K^2}{(1-\gamma)^2}\mathbb{E}_{\widehat{m}}[q_{\widehat{m},g^\star}(1-q_{\widehat{m},g^\star})], \tag{34}$$

so that $\sigma^2_{\mathrm{MC}} = \sigma^2_{\mathrm{RB}} + \Delta_\nu$.

Corollary 5 (Exact planning-sample ratio)**.** *Under Assumptions 2–3, let*

$$\overline{Q}^E_{\nu,N} := \frac{1}{N}\sum_{n=1}^{N} \widehat{Q}^{\nu,E}_{\gamma,g^\star,n}(s,a), \qquad E \in \{\mathrm{MC},\mathrm{RB}\}. \tag{35}$$

Then

$$\mathrm{Var}(\overline{Q}^{\mathrm{MC}}_{\nu,N}) = \frac{\sigma^2_{\mathrm{MC}}}{N}, \qquad \mathrm{Var}(\overline{Q}^{\mathrm{RB}}_{\nu,N}) = \frac{\sigma^2_{\mathrm{RB}}}{N}. \tag{36}$$

When both row variances are positive, equal sample-mean variance requires

$$\frac{N_{\mathrm{MC}}}{N_{\mathrm{RB}}} = \frac{\sigma^2_{\mathrm{MC}}}{\sigma^2_{\mathrm{RB}}} = 1 + \frac{\Delta_\nu}{\sigma^2_{\mathrm{RB}}}. \tag{37}$$

For integer counts, the smallest literal budget matching N_{RB} conditional rows is

$$N^{\mathrm{match}}_{\mathrm{MC}}(N_{\mathrm{RB}}) = \left\lceil \frac{\sigma^2_{\mathrm{MC}}}{\sigma^2_{\mathrm{RB}}} N_{\mathrm{RB}} \right\rceil. \tag{38}$$

If $\sigma^2_{\mathrm{RB}} = 0 < \sigma^2_{\mathrm{MC}}$, the RB row is deterministic and no finite literal budget matches its zero variance.

Proof. Independence gives, for either estimator E,

$$\mathrm{Var}(\overline{Q}^E_{\nu,N}) = \frac{1}{N^2}\sum_{n=1}^{N} \mathrm{Var}(\widehat{Q}^{\nu,E}_{\gamma,g^\star,n}) = \frac{\sigma^2_E}{N}. \tag{39}$$

Equating $\sigma^2_{\mathrm{RB}}/N_{\mathrm{RB}}$ and $\sigma^2_{\mathrm{MC}}/N_{\mathrm{MC}}$ proves Equation 37; solving the corresponding inequality for integer N_{MC} proves Equation 38. □

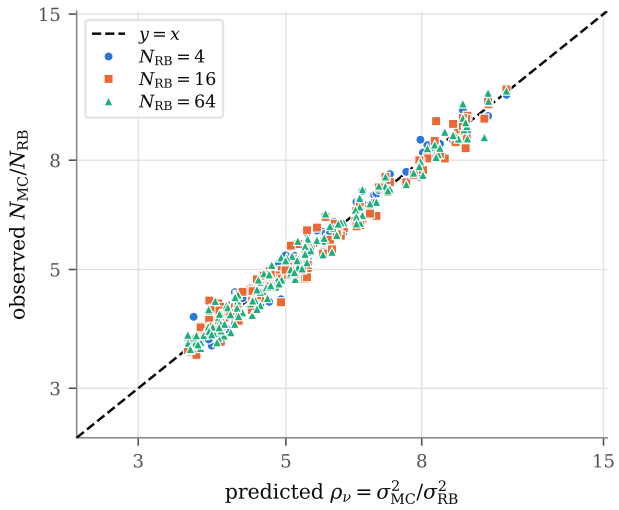


Figure 9: **Sample Ratio**: Empirical sampling ratio required to match the estimator variances, compared with the theoretical ratio from Cor. 5.

D.4 Decision signal-to-noise and selection regret

Sparse scores as a low-SNR decision problem. With sparse goal rewards, the literal estimator often assigns zero to a candidate because none of its sampled terminal events succeeds. Different candidates can therefore receive identical zero scores even when their expected values differ. This is the finite-sample manifestation of a weak planning signal: the value differences between candidates are small relative to the noise with which they are estimated. To make the connection precise, write the terminal contribution of one rollout, conditional on its simulated prefix $\mathcal{F}$, as

$$Y^{\mathrm{MC}} = \frac{w_K}{1-\gamma} B, \qquad B \mid \mathcal{F} \sim \mathrm{Bernoulli}(q), \qquad Y^{\mathrm{RB}} = \frac{w_K}{1-\gamma} q. \tag{40}$$

For N sampled prefixes with terminal occupancies $q_1, \ldots, q_N$, literal MC returns an all-zero terminal contribution with conditional probability $\prod_{n=1}^{N}(1-q_n)$. RB instead returns the fractional score $\frac{w_K}{N(1-\gamma)} \sum_n q_n$, which is zero only if every q_n is zero. Thus the zero-and-tie phenomenon and the variance gap in Theorem 1 are two views of the same randomness: RB removes the conditional Bernoulli variance that turns a prefix with $0 < q < 1$ into a hard success or failure.

We now connect this variance reduction to the quantity that matters to the planner: selecting the best candidate from a finite set. Condition on the realized candidate set $\mathcal{C} = \{\nu_1, \ldots, \nu_M\}$ and on the current GHM. This removes the randomness used by iCEM to propose the candidates and leaves only the independent composed-GHM rollouts used to score them. Let

$$Q_i := Q^{\nu_i, \widehat{m}}_{\gamma, g^\star}(s, a), \qquad \nu_\star := \arg\max_{\nu_i \in \mathcal{C}} Q_i, \qquad \delta_i := Q_\star - Q_i > 0 \quad (i \neq \star), \tag{41}$$

where we assume a unique optimum for clarity. Candidate i is evaluated by the sample mean $\overline{Q}^E_{i,N}$ of N independent rollouts under $E \in \{\mathrm{MC}, \mathrm{RB}\}$. Let $\sigma^2_{i,E}$ denote its single-rollout variance. Because different candidates are evaluated with independent rollouts, the variance of the estimated score difference is

$$\mathrm{Var}\left(\overline{Q}^E_{i,N} - \overline{Q}^E_{\star,N}\right) = \frac{V^E_i}{N}, \qquad V^E_i := \sigma^2_{i,E} + \sigma^2_{\star,E}. \tag{42}$$

The relevant decision signal is the value margin δ_i, while the relevant noise is the standard deviation of this pairwise comparison. We therefore define the squared decision signal-to-noise ratio

$$S^E_i := \frac{N\delta_i^2}{V^E_i}, \tag{43}$$

with the convention $S^E_i = \infty$ when $V^E_i = 0$. Large S^E_i means that the true value margin is large relative to finite-rollout estimation noise, so candidate i is unlikely to be ranked above the optimum.

Proposition 1 (Variance-sensitive selection-regret bound)**.** For each $i \neq \star$, let

$$\Delta_i := \sigma^2_{i,\mathrm{MC}} - \sigma^2_{i,\mathrm{RB}} \tag{44}$$

be the exact single-candidate variance gap from Theorem 1, and set $D_i := \Delta_i + \Delta_\star$. Then

$$V_i^{\mathrm{RB}} = V_i^{\mathrm{MC}} - D_i, \qquad S_i^{\mathrm{RB}} \geq S_i^{\mathrm{MC}}. \tag{45}$$

Define the pairwise misranking event

$$U_{i,E} := \{\overline{Q}^E_{i,N} \geq \overline{Q}^E_{\star,N}\}. \tag{46}$$

Then

$$\mathbb{P}(U_{i,E}) \leq \frac{V_i^E}{V_i^E + N\delta_i^2} = \frac{1}{1+S_i^E}. \tag{47}$$

If $\widehat{\nu}_E$ maximizes the empirical score and $\mathcal{R}_E := Q_\star - Q_{\widehat{\nu}_E}$ is its selection regret, then

$$\begin{aligned}\mathbb{E}[\mathcal{R}_E] &= \sum_{i\neq\star} \delta_i \mathbb{P}(\widehat{\nu}_E = \nu_i) \\ &\leq B_E(N) := \sum_{i\neq\star} \frac{\delta_i V_i^E}{V_i^E + N\delta_i^2} = \sum_{i\neq\star} \frac{\delta_i}{1+S_i^E}.\end{aligned} \tag{48}$$

Consequently, RB has the uniformly tighter variance-sensitive bound

$$B_{\mathrm{MC}}(N) - B_{\mathrm{RB}}(N) = \sum_{i\neq\star} \frac{N\delta_i^3 D_i}{(V_i^{\mathrm{MC}} + N\delta_i^2)(V_i^{\mathrm{MC}} - D_i + N\delta_i^2)} \geq 0. \tag{49}$$

Proof. Theorem 1 gives $\sigma^2_{j,\mathrm{RB}} = \sigma^2_{j,\mathrm{MC}} - \Delta_j$ for every candidate j, which proves Equation 45. Define $Z_{i,E} := \overline{Q}^E_{i,N} - \overline{Q}^E_{\star,N}$. Unbiasedness and independent evaluation give

$$\mathbb{E}[Z_{i,E}] = -\delta_i, \qquad \mathrm{Var}(Z_{i,E}) = \frac{V_i^E}{N}. \tag{50}$$

The event $U_{i,E}$ is equivalently $\{Z_{i,E} - \mathbb{E}[Z_{i,E}] \geq \delta_i\}$. Cantelli's inequality proves Equation 47. Selecting ν_i implies that it outranks $\nu_\star$, so $\mathbb{P}(\widehat{\nu}_E = \nu_i) \leq \mathbb{P}(U_{i,E})$. Substituting this inequality into the exact regret decomposition proves Equation 48. Finally, substituting $V_i^{\mathrm{RB}} = V_i^{\mathrm{MC}} - D_i$ into the two bounds and simplifying proves Equation 49. □

From pairwise SNR to candidate-set regret. The proposition predicts the empirical ordering in Figure 10(a): at the same rollout budget N, RB removes D_i from every pairwise comparison variance and therefore raises every S_i^E. The resulting reduction in mean regret need not be summarized by the single hardest comparison. The figures divide regret by $Q_\star$ to compare candidate sets across seeds; this positive within-set normalization does not change any ordering. In particular, $S^E_{\min} := \min_{i\neq\star} S_i^E$ ignores both the cost δ_i of selecting a candidate and the cumulative contribution of the remaining competitors. A low-SNR candidate with a very small value gap can be frequently misranked while contributing little regret. Conversely, many moderately difficult competitors can matter collectively. This explains why plotting regret against $S^E_{\min}$ does not align MC and RB in Figure 10(b).

The theoretically justified aggregation retains the full set of pairwise SNRs. Let

$$G := \sum_{i\neq\star} \delta_i, \qquad O_i := \frac{\delta_i}{G}, \qquad S^E_{\mathrm{eff}} := \left(\sum_{i\neq\star} \frac{O_i}{1+S_i^E}\right)^{-1} - 1. \tag{51}$$

Then the regret bound in Equation 48 becomes

$$B_E(N) = \frac{G}{1+S^E_{\mathrm{eff}}}, \qquad x_E(N) := \frac{B_E(N)}{Q_\star} = \sum_{i\neq\star} \frac{\delta_i/Q_\star}{1+S_i^E}. \tag{52}$$

Hence $x_E(N)$ is not separate from the SNR account: it is the normalized regret bound induced by an effective candidate-set decision SNR. Because RB increases every S_i^E, it also increases S_{eff}^E and decreases $x_E(N)$. Unlike a simple average SNR, $x_E(N)$ also retains the scale and geometry of the realized candidate set through the normalized gaps $\delta_i/Q_\star$.

For comparison, Figure 10(c) uses the simpler gap-weighted average

$$\overline{S}_w^E := \frac{\sum_{i\neq\star}\delta_i S_i^E}{\sum_{i\neq\star}\delta_i}. \tag{53}$$

Unlike $S_{\min}^E$, this statistic incorporates the whole candidate set and weights comparisons by their regret cost. The collapse in panel (c) shows that, for these tabular candidate sets, $\overline{S}_w^E$ is an effective one-dimensional empirical proxy for the distribution of the S_i^E. This collapse is an empirical diagnostic, not a consequence of Proposition 1; candidate sets having the same weighted average can in general have different regret.

Figure 10(d) uses the formally justified aggregate $x_E(N)$. All empirical regrets lie below their corresponding bound, as predicted by Proposition 1. More importantly for the mechanistic claim, MC and RB follow the same empirical regret–bound curve: conditioning on $x_E(N)$ largely removes the estimator-specific separation seen at a matched rollout budget. Cantelli's inequality guarantees only that the points lie below the diagonal; it does not guarantee this collapse. The collapse is therefore additional evidence that RB improves selection mainly by moving the planner to a higher-decision-SNR regime, rather than by changing the relationship between decision noise and regret.

Candidate-wise mechanism. Figure 11 resolves the candidate-set averages into individual comparisons. The quantity $\delta_i\mathbb{P}(U_{i,E})$ is the regret cost of candidate i multiplied by the probability that it outranks the optimum. It upper-bounds the corresponding contribution to expected selection regret because

$$\delta_i\mathbb{P}(\widehat{\nu}_E = \nu_i) \le \delta_i\mathbb{P}(U_{i,E}). \tag{54}$$

Panel (a) plots these pairwise upper-bound contributions, normalized by $Q_\star$ for comparability across seeds, against S_i^E. Exact collapse is not expected because the ordinate still contains the candidate-specific scale δ_i.

Dividing by the standard error of the pairwise comparison removes that scale:

$$\frac{\delta_i\mathbb{P}(U_{i,E})}{\sqrt{V_i^E/N}} = \sqrt{S_i^E}\,\mathbb{P}(U_{i,E}). \tag{55}$$

Cantelli's inequality supplies the distribution-free envelope

$$\sqrt{S_i^E}\,\mathbb{P}(U_{i,E}) \le \frac{\sqrt{S_i^E}}{1+S_i^E}. \tag{56}$$

If the sample-mean difference is approximately Gaussian, as suggested by the central limit theorem, then more specifically

$$\mathbb{P}(U_{i,E}) \approx \Phi(-\sqrt{S_i^E}), \qquad \sqrt{S_i^E}\,\mathbb{P}(U_{i,E}) \approx \sqrt{S_i^E}\,\Phi(-\sqrt{S_i^E}), \tag{57}$$

where Φ is the standard normal cumulative distribution function. Thus, after standardization, both estimators should follow the same curve if their difference is explained primarily by the variance removed by RB. The overlap in Figure 11(b) supports precisely this mechanism: RB moves comparisons to larger S_i^E, rather than changing the relationship between standardized SNR and ranking error.

Remark 6 (Variance bound versus exact regret). Variance reduction alone does not universally order the exact argmax regret for arbitrary estimator distributions, because selection also depends on tail shape and dependence across comparisons. Proposition 1 guarantees a tighter distribution-free upper bound under the stated independent-rollout evaluation protocol. Figures 10 and 11 provide the complementary empirical test: in the tabular experiment, candidate-set regret is organized by an aggregate of the pairwise SNRs, and the estimator-specific discrepancy nearly disappears after candidate-wise SNR standardization.

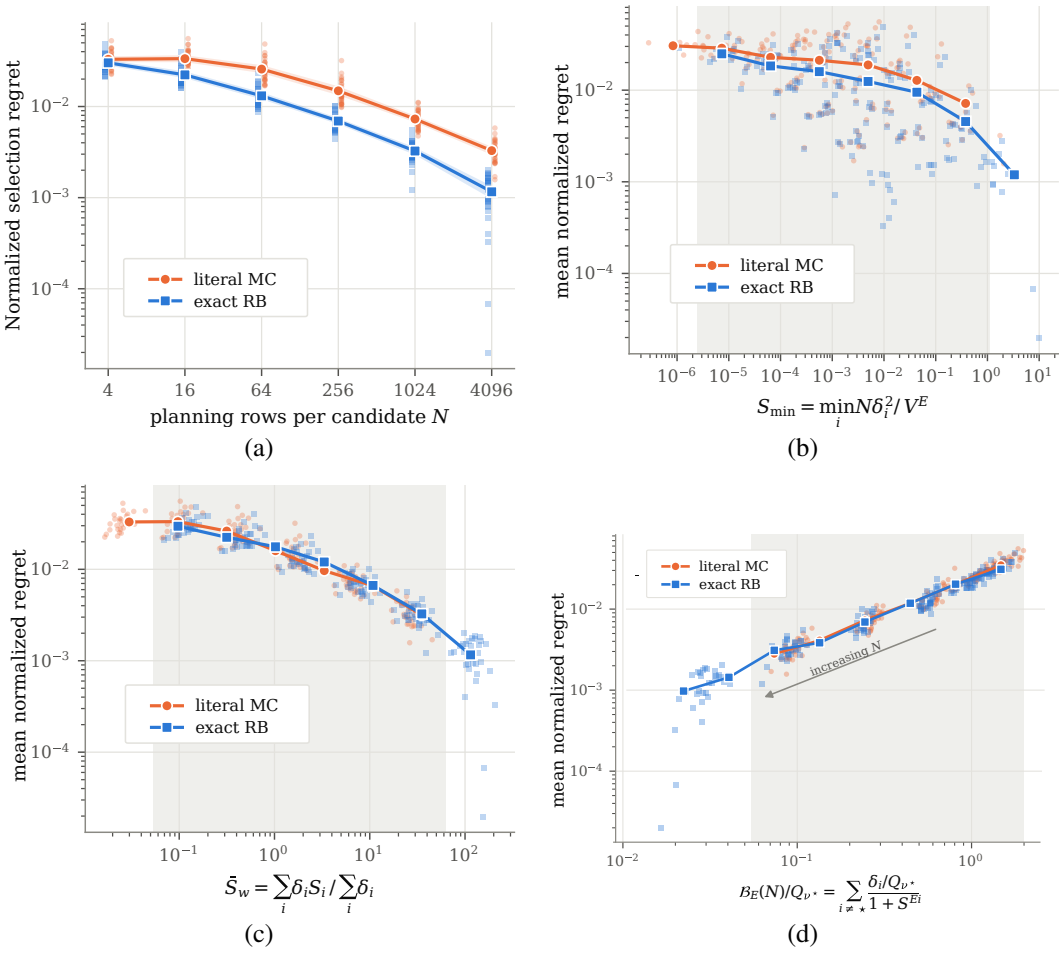
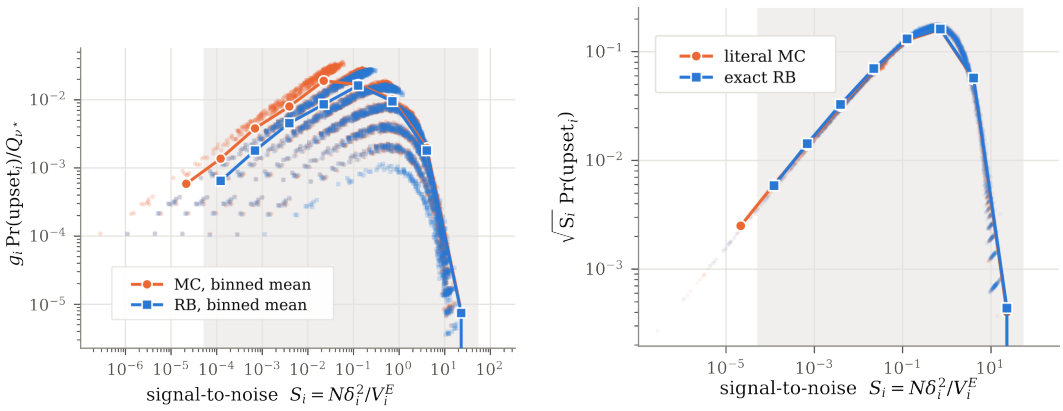


Figure 10: **Candidate-set regret is organized by decision SNR.** Each point is one of 30 tabular seeds at a given rollout budget. **(a)** At fixed N, exact RB consistently reduces normalized mean selection regret relative to literal MC. **(b)** The minimum pairwise SNR does not align the estimators because it retains only one competitor and ignores its regret cost. **(c)** Regret approximately collapses across estimators when plotted against the gap-weighted average SNR $\overline{S}_w^E$, which summarizes all competitors while emphasizing costly errors. **(d)** The theorem-aligned plot uses $x_E(N) = B_E(N)/Q_\star = (G/Q_\star)/(1 + S_{\text{eff}}^E)$. Empirical regret remains below this distribution-free upper bound and MC and RB follow a common curve. Thus the estimator advantage is explained primarily by RB's increase in aggregate decision SNR; $\overline{S}_w^E$ in panel (c) is a simpler empirical proxy for the same signal advantage.

Implication for continuous online RL. The central implication is improved reliability of the planning signal. At a fixed simulation budget, sparse MC scores can obscure differences between promising and poor candidates, leaving selection sensitive to random terminal hits or zero-score ties. Exact terminal conditioning preserves expected candidate values while reducing comparison noise, increasing planning SNR and tightening the selection-regret bound. The empirical collapse in Figure 10 supports this account: much of the estimator separation diminishes after accounting for planning SNR. Appendix F.5.1 examines this mechanism on frozen continuous-control models. Shared-reference comparisons isolate the scoring-noise advantage, while single-checkpoint and across-training curves measure how reliably MC and learned TC optimize their own expected scores. Their closer alignment in SNR-based coordinates provides a continuous counterpart to the tabular analysis, with the degree of alignment varying across checkpoints. Appendix F.5.2 provides more direct evidence: adding terminal noise while preserving expected TC scores increases ranking errors. These experiments support signal reliability as a mechanism behind OGPC-TC's planning advan-



(a) (b)

Figure 11: **Candidate-wise evidence for the variance mechanism. (a)** Normalized pairwise upper-bound contributions $\delta_i \mathbb{P}(U_{i,E})/Q_\star$ plotted against decision SNR. Residual separation is expected because the vertical axis retains the candidate-specific value scale δ_i. **(b)** Dividing by the pairwise standard error $\sqrt{V_i^E/N}$ yields the dimensionless contribution $\sqrt{S_i^E}\mathbb{P}(U_{i,E})$. The MC and RB curves nearly coincide, consistent with their regret difference being mediated by RB's increase in decision SNR.

tage; learned-head bias remains a separate limitation on which plans it prefers (Appendix D.5). More reliable selection can guide collection toward informative experience, providing a possible explanation for faster learning in Section 4.1. A secondary interpretation connects planning to gradients: for a stochastic choice between two fixed candidates, the gradient estimated by evaluating both scores is proportional to their difference and has SNR S_i^E. This highlights their shared dependence on reliable value comparisons, without identifying planning SNR with the deployed worker's gradient SNR.

D.5 Candidate selection with an approximate head

Define the learned-head row

$$\widehat{Q}^{\nu,\phi}_{\gamma,g^\star}(s,a) := \frac{1}{1-\gamma}\big(R_{<K} + w_K q_{\phi,g^\star}\big). \tag{58}$$

Lemma 7 (Argmax stability under uniform score error)**.** *Let $V_1,\dots,V_M$ be arbitrary values and let $\widehat{V}_1,\dots,\widehat{V}_M$ satisfy $\max_i |\widehat{V}_i - V_i| \le \eta$. If $i^\star \in \arg\max_i V_i$ and $\widehat{i} \in \arg\max_i \widehat{V}_i$, then*

$$V_{i^\star} - V_{\widehat{i}} \le 2\eta. \tag{59}$$

Proof. On the stated event,

$$V_{i^\star} - V_{\widehat{i}} \le (\widehat{V}_{i^\star} + \eta) - (\widehat{V}_{\widehat{i}} - \eta) \le 2\eta, \tag{60}$$

because $\widehat{V}_{\widehat{i}} \ge \widehat{V}_{i^\star}$. □

Theorem 8 (Candidate selection with an approximate head)**.** *Consider a realized set of M candidates evaluated with N fresh rows per candidate. Suppose the variance of every exact RB row is at most $\sigma^2_{\max}$ and*

$$|q_{\phi,g^\star} - q_{\widehat{m},g^\star}| \le \epsilon_q \tag{61}$$

on every terminal-prefix input scored in the planner call. Define

$$L := \log\frac{2M}{\delta}, \qquad \eta_N^{\mathrm{RB}} := \sqrt{\frac{2\sigma^2_{\max} L}{N}} + \frac{2L}{3(1-\gamma)N}. \tag{62}$$

With probability at least $1-\delta$, all learned-head sample means satisfy

$$\max_{1\le i\le M}\left|\overline{Q}^{\phi}_{i,N} - Q^{\nu_i,\widehat{m}}_{\gamma,g^\star}(s,a)\right| \le \eta_N^{\mathrm{RB}} + \frac{w_K}{1-\gamma}\epsilon_q. \tag{63}$$

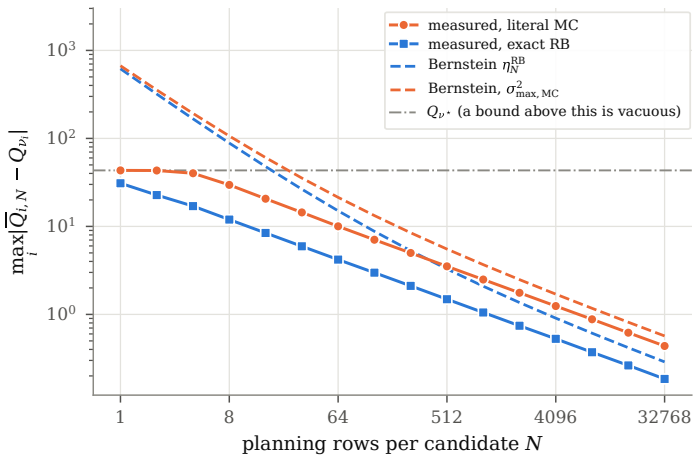


Figure 12: **Max regret bound:** Visualization of bounds from Theorem 8 against the empirical regret estimation in the tabular Snake Maze setting.

If $\widehat{\nu}_\phi$ maximizes the empirical learned-head score and $\nu^\star$ maximizes the current-GHM value in the same candidate set, then

$$Q^{\nu^\star,\widehat{m}}_{\gamma,g^\star}(s,a) - Q^{\widehat{\nu}_\phi,\widehat{m}}_{\gamma,g^\star}(s,a) \le 2\left(\eta_N^{\mathrm{RB}} + \frac{w_K}{1-\gamma}\epsilon_q\right). \tag{64}$$

If, separately,

$$\epsilon_{\mathrm{GHM}} := \max_{\nu \text{ in } \mathcal{C}} \left| Q^{\nu,\widehat{m}}_{\gamma,g^\star}(s,a) - Q^{\nu,\mathrm{env}}_{\gamma,g^\star}(s,a) \right|, \tag{65}$$

the environment-level selection regret is at most

$$2\left(\eta_N^{\mathrm{RB}} + \frac{w_K}{1-\gamma}\epsilon_q + \epsilon_{\mathrm{GHM}}\right). \tag{66}$$

Proof. For candidate i, conditional Bernstein's inequality for exact RB rows in $[0,(1-\gamma)^{-1}]$, followed by a union bound over the M fixed or sequentially proposed candidates, gives

$$\mathbb{P}\left(\max_i |\overline{Q}^{\mathrm{RB}}_{i,N} - Q_i| \le \eta_N^{\mathrm{RB}}\right) \ge 1-\delta. \tag{67}$$

Couple the learned-head and exact RB rows using the same simulated prefix. Equation 61 gives

$$\left|\widehat{Q}^{\nu,\phi}_{\gamma,g^\star} - \widehat{Q}^{\nu,\mathrm{RB}}_{\gamma,g^\star}\right| = \frac{w_K}{1-\gamma}|q_{\phi,g^\star} - q_{\widehat{m},g^\star}| \le \frac{w_K}{1-\gamma}\epsilon_q. \tag{68}$$

The same bound holds for the sample means. Combining it with Equation 67 proves Equation 63; applying Lemma 7 proves Equation 64. Finally, applying Equation 65 once at the environment-best candidate and once at the selected candidate adds $2\epsilon_{\mathrm{GHM}}$, proving Equation 66. □

Distribution-free alternative. Without a variance bound, conditional Hoeffding and the same union-bound argument give

$$\eta_N^{\mathrm{H}} := \frac{1}{1-\gamma}\sqrt{\frac{1}{2N}\log\frac{2M}{\delta}}. \tag{69}$$

The conclusions of Theorem 8 remain valid after replacing η_N^{RB} by η_N^{H}. The quantity ϵ_q measures approximation of the current-GHM conditional occupancy; it does not include GHM error relative to the environment. Moreover, the learned head is not automatically unbiased or variance-reducing unless it equals the exact conditional expectation.

D.6 CONDITIONAL RESULT DURING ONLINE TRAINING

Corollary 9 (Online conditional Rao–Blackwellization)**.** *Let $\mathcal{H}_t$ contain all data, replay contents, parameters, and optimization randomness available before planner call t. Conditional on $\mathcal{H}_t$, freeze the current GHM and policy while the call is evaluated. Fix a candidate measurable with respect to $\mathcal{H}_t$, or condition additionally on the within-call proposal history that selected it. Let $\mathcal{F}_{t,K-1}$ be the simulated-prefix sigma-field and define*

$$q_{t,g^\star} := \mathbb{E}[B_{t,K} \mid \mathcal{H}_t \vee \mathcal{F}_{t,K-1}]. \tag{70}$$

Then, almost surely,

$$\widehat{Q}^{\nu,\mathrm{RB}}_{\gamma,g^\star,t} = \mathbb{E}\Big[\widehat{Q}^{\nu,\mathrm{MC}}_{\gamma,g^\star,t} \mid \mathcal{H}_t \vee \mathcal{F}_{t,K-1}\Big], \tag{71}$$

$$\mathbb{E}[\widehat{Q}^{\nu,\mathrm{RB}}_{\gamma,g^\star,t} \mid \mathcal{H}_t] = \mathbb{E}[\widehat{Q}^{\nu,\mathrm{MC}}_{\gamma,g^\star,t} \mid \mathcal{H}_t], \tag{72}$$

$$\mathrm{Var}(\widehat{Q}^{\nu,\mathrm{MC}}_{\gamma,g^\star,t} \mid \mathcal{H}_t) - \mathrm{Var}(\widehat{Q}^{\nu,\mathrm{RB}}_{\gamma,g^\star,t} \mid \mathcal{H}_t) = \frac{w_K^2}{(1-\gamma)^2}\mathbb{E}[q_{t,g^\star}(1-q_{t,g^\star}) \mid \mathcal{H}_t] \geq 0. \tag{73}$$

If Assumption 3 holds conditionally on $\mathcal{H}_t$, the N-sample and candidate selection results also hold conditionally at call t.

Proof. Fix a realization of $\mathcal{H}_t$. The current policy and GHM are then fixed, leaving only planner-simulation randomness. Because $R_{t,<K}$ is measurable with respect to $\mathcal{H}_t \vee \mathcal{F}_{t,K-1}$,

$$\mathbb{E}\Big[\widehat{Q}^{\nu,\mathrm{MC}}_{\gamma,g^\star,t} \mid \mathcal{H}_t \vee \mathcal{F}_{t,K-1}\Big] = \frac{1}{1-\gamma}\left(R_{t,<K} + w_K\mathbb{E}[B_{t,K} \mid \mathcal{H}_t \vee \mathcal{F}_{t,K-1}]\right) \tag{74}$$

$$= \frac{1}{1-\gamma}(R_{t,<K} + w_K q_{t,g^\star}) = \widehat{Q}^{\nu,\mathrm{RB}}_{\gamma,g^\star,t}. \tag{75}$$

The conditional tower property proves Equation 72. Applying the conditional law of total variance and using the Bernoulli variance $q_{t,g^\star}(1-q_{t,g^\star})$ proves Equation 73. Conditional independence of fresh rows permits the sample-mean and concentration arguments to be repeated under the conditional law given $\mathcal{H}_t$. Since the argument is pointwise in $\mathcal{H}_t$, the learned components may change between planner calls. □

D.7 THEORETICAL BOUNDS FOR COMPOSITIONAL PLANNING

Appendices D.4–D.5 analyse how scoring noise and approximation errors affect candidate selection. Here we examine the execution that follows: with the policy and models fixed, does compositional planning increase Time at Goal relative to directly commanding the final goal? We first bound execution gain by the value available from composition minus accumulated local shortfalls. We then derive a finite-sample lower bound from execution measurements and evaluate it as the number of planned subgoals varies, with terminal-conditioned scoring held fixed. This complements the evaluation-time study in Appendix F.3, Figure 23.

Values and execution. For an episode of H steps, let state s include elapsed time $t(s)$ and the fixed task goal $g^\star$. Rewards $r_{g^\star} \in [0,1]$ and continuation values are zero after termination. Define the environment value of a composition ν after first action a by

$$Q^\nu(s,a) = \frac{1}{H}\mathbb{E}_{P,\nu}\left[\sum_{j=0}^{H-t(s)-1} r_{g^\star}(S_{j+1}) \mid S_0 = s, A_0 = a\right].$$

All values are normalized by the full episode length H. This keeps values at different times on the same scale. Write $\pi_0 = \pi_{g^\star}$ and $V_0(s) = \int Q^{\pi_0}(s,a)\pi_0(da \mid s)$ for direct-goal execution. Let $\mathfrak{P}_K$ be nested families of geometric compositions with at most K subgoal-conditioned policies, including π_0 and the remaining portions of each composition (Thakoor et al., 2022). We use $\bar{\beta}$ for the execution discount, which equals the shared nonfinal-hop discount $\beta_1 = \cdots = \beta_{K-1}$ when $K > 1$. Within each GSP, nonfinal policies switch with probability $1 - \bar{\beta}/\gamma$, where $0 \leq \bar{\beta} < \gamma < 1$. The final policy continues until the episode ends. For these fixed goal-conditioned policies, the best value

over the composition family and first action is $W_K(s) = \sup_{\nu \in \mathfrak{P}_K,\, a \in \mathcal{A}} Q^\nu(s,a)$, which we assume measurable. Nesting gives $W_{K+1} \geq W_K \geq V_0$. These execution values measure undiscounted Time at Goal; the planner selects candidates using the discounted learned scores analysed above.

Execution follows Algorithm 2, with the policy and models held fixed. Let $J_K(s)$ denote OGPC's expected episode reward divided by H, starting from s with K planned subgoals and empty planner memory.

Local progress and the episode bound. At planning call n, denote its time by t_n, the selected candidate by c_n, and the keep_seed indicator from Algorithm 1 by I_n. This indicator equals one when the original direct-goal seed c^0 is retained, and zero otherwise. Choose a measurable value function $F : \mathcal{S} \to [0,1]$, zero at termination, to assess each commitment. For example, $F = V_0$ compares with direct execution and $F = W_K$ with the best available composition. Let τ be the independently sampled commitment length, truncated at episode end. Define the commitment value and its Bellman residual by

$$T_{c,i}F(s) = \mathbb{E}_{c,i}\left[\frac{1}{H}\sum_{j<\tau} r_{g^\star}(S_{j+1}) + F(S_\tau) \mid S_0 = s\right], \qquad a_F(s,c,i) = T_{c,i}F(s) - F(s).$$

Thus a_F is reward earned plus expected value remaining, minus value before the commitment. Positive values indicate progress relative to F; negative values indicate a shortfall. Write $\mathcal{E}_F(s) = \mathbb{E}_s \sum_{t_n<H}[-a_F(S_{t_n}, c_n, I_n)]_+$ for the expected accumulated shortfalls when planning over K subgoals, with $[u]_+ = \max\{u, 0\}$.

Theorem 10 (Execution improvement)**.** *For every such F,*

$$J_K(s) - V_0(s) = F(s) - V_0(s) + \mathbb{E}_s \sum_{t_n<H} a_F(S_{t_n}, c_n, I_n). \tag{76}$$

In particular,

$$\boxed{J_K(s) - V_0(s) \geq W_K(s) - V_0(s) - \mathcal{E}_{W_K}(s).} \tag{77}$$

Proof. Condition on each call's history and output. Summing $a_F = T_{c,i}F - F$ and taking expectations cancels intermediate F-values, leaving $\mathbb{E}_s \sum_n a_F = J_K(s) - F(s)$. Subtract $V_0(s)$ to obtain Equation 76. Set $F = W_K$ and use $a_F \geq -[-a_F]_+$ for Equation 77. At most H calls and zero terminal values justify this performance-difference argument (Sutton et al., 1999; Kakade & Langford, 2002). □

D.7.1 Calculation and proof of the numerical lower bound

Equation 77 guarantees improvement when the available composition gain exceeds the shortfalls. To evaluate execution, we bound $J_K - V_0$ directly, averaging over a distribution μ of initial states and task goals: $J_K = \mathbb{E}_\mu J_K(S_0)$ and $V_0 = \mathbb{E}_\mu V_0(S_0)$. Taking $F = 0$ in Equation 76 expresses J_K as the sum of expected commitment rewards. We use this identity to construct an unbiased measurement from short segments, then bound its sampling error. This measures the combined execution outcome without separately estimating W_K or $\mathcal{E}_{W_K}$.

A commitment measurement with mean J_K. To recover Time at Goal, commitments must be weighted by the steps they contribute. Let $p_u = (1-\bar\beta)\bar\beta^{u-1}$ for $1 \leq u < \tau_{\max}$ and $p_{\tau_{\max}} = \bar\beta^{\tau_{\max}-1}$ be the nominal duration probabilities. At a call at time t, set $d_t = \min\{\tau_{\max}, H-t\}$ and $\bar\tau_t = \sum_{j=0}^{d_t-1} \bar\beta^j = \mathbb{E}[\tau \mid t]$. After running the planner, select this call for measurement with probability $\bar\tau_t/(H-t)$; otherwise execute its command for u steps and continue, with joint probability $p_u(H-t-u)_+/(H-t)$ for each u. These probabilities sum to one. Planner and cache updates retain their ordinary conditional rules.

At the selected call, independently simulate a policy trajectory of length d_t using the selected command and first-action rule, and record

$$Y_K = \frac{1}{\bar\tau_t}\sum_{j=0}^{d_t-1} \bar\beta^j r_{g^\star}(S_{j+1}) \in [0,1]. \tag{78}$$

The weight $\bar{\beta}^j$ is the probability that the commitment lasts beyond step j. Summing with these weights averages out duration randomness by conditional expectation (Blackwell, 1947).

Proposition 2 (Unbiased execution measurement)**.** The sampling rule above satisfies $\mathbb{E}Y_K = J_K$.

Proof. Along a history ending at t_n, continuation probabilities multiply its ordinary execution probability by factors $(H - t_{j+1})/(H - t_j)$, whose product is $(H - t_n)/H$. Selecting that call multiplies by $\bar{\tau}_{t_n}/(H - t_n)$, leaving weight $\bar{\tau}_{t_n}/H$. Its expected measured contribution is therefore $(\bar{\tau}_{t_n}/H)\mathbb{E}[Y_K \mid s, c, i]$, exactly its expected commitment reward divided by H. Summing over histories recovers J_K, including search randomness and retained candidates. □

From segment rewards to measured gain. In each trial b, draw a task from μ and select a call by the rule above. Simulate n_{branch} conditionally independent policy trajectories from its state with its selected command. Each *branch* ℓ gives $Y_{b,K,\ell}$ through Equation 78. An independent direct-policy episode on the same task gives reward fraction $R_b^0 \in [0, 1]$. Then

$$\widehat{Y}_{b,K} = \frac{1}{n_{\text{branch}}} \sum_{\ell=1}^{n_{\text{branch}}} Y_{b,K,\ell}, \qquad X_{b,K} = \widehat{Y}_{b,K} - R_b^0 \in [-1, 1], \qquad \mathbb{E}X_{b,K} = J_K - V_0.$$

The last equality follows from Proposition 2. Across $n_{\text{trial}} \geq 2$ independent trials, compute

$$\bar{X}_K = \frac{1}{n_{\text{trial}}} \sum_{b=1}^{n_{\text{trial}}} X_{b,K}, \qquad s_{X,K}^2 = \frac{1}{n_{\text{trial}} - 1} \sum_{b=1}^{n_{\text{trial}}} (X_{b,K} - \bar{X}_K)^2.$$

Thus $s_{X,K}^2$ measures variation between trial gains after averaging their branches; the independent sample count is n_{trial}. Choose a failure probability $\delta \in (0, 1)$ before inspecting the measurements. It sets the confidence level of the execution bound.

Theorem 11 (Finite-sample execution bound)**.** *For i.i.d. trials as above, with probability at least* $1 - \delta$*,*

$$\boxed{J_K - V_0 \geq \bar{X}_K - \underbrace{\sqrt{\frac{2s_{X,K}^2 \log(2/\delta)}{n_{\text{trial}}}}}_{\textit{variance term}} - \underbrace{\frac{14 \log(2/\delta)}{3(n_{\text{trial}} - 1)}}_{\textit{range term}}.} \tag{79}$$

Proof. Set $Z_b = (1 - X_{b,K})/2 \in [0, 1]$. The empirical Bernstein inequality (Maurer & Pontil, 2009, Theorem 4) gives $\mathbb{E}Z - \bar{Z} \leq \sqrt{2s_Z^2 \log(2/\delta)/n_{\text{trial}}} + 7\log(2/\delta)/(3(n_{\text{trial}} - 1))$ with the stated probability. Use $s_Z^2 = s_{X,K}^2/4$ and $J_K - V_0 = 1 - 2\mathbb{E}Z$ to rearrange this into Equation 79. □

The two subtracted terms account for measurement uncertainty. Changing K changes the selected commands, hence $\bar{X}_K$ and $s_{X,K}^2$; at fixed n_{trial}, δ, the range term is unchanged. To compare two numbers of planned subgoals, replace R_b^0 by $\widehat{Y}_{b,j}$ on the same initial task to bound $J_K - J_j$. Theorem 11 uses unbiased execution measurements; the selection-error bounds in Appendices D.4–D.5 describe the earlier scoring stage.

D.7.2 Empirical verification of the Time at Goal bound

We evaluate Equation 79 in the JAX Pointax Medium environment, with position–velocity state and two-dimensional actions. Starts and goals are sampled from the free cells in Figure 13(a). Each step within distance 0.45 of the fixed goal earns one reward over $H = 1000$ steps, so Time at Goal measures both reaching the region and remaining there.

Evaluation protocol. We freeze one CRL policy trained for 4.19M transitions, its GHM and terminal head. All OGPC conditions share these components and use terminal-conditioned (TC) scoring; we vary the number of planned subgoals K. Direct execution uses the same policy and always commands the final goal. OGPC uses $K \in \{1, 2, 4, 6\}$, 12 candidates, four iCEM rounds and $N = 64$ scoring rows per candidate, with $\bar{\beta} = .95$, $\tau_{\max} = 59$ and $\gamma = .99$. We use $n_{\text{trial}} = 8192$ commitment trials, with $n_{\text{branch}} = 16$ trajectories per selected planning call. We also evaluate 4096 full

episodes per condition. Initial states and task goals are paired across conditions. The commitment measurements and full-episode evaluations use separate samples. For six comparisons, we allocate $\delta = .025/6$ to each commitment bound and $.025/12$ to each full-episode tail. A union bound gives at least 95% simultaneous coverage for all bounds and intervals.

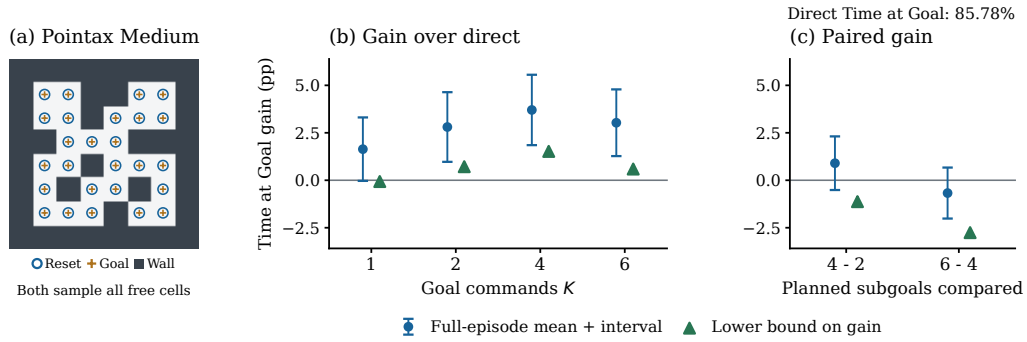


Figure 13: **Execution improvement with fixed policy and models.** (a) Medium layout. (b) Gain over direct execution. (c) Gain from increasing the number of planned subgoals. Blue points and intervals use independent full episodes. Green triangles use commitment measurements in Equation 79. Positive lower bounds establish improvement. Gains are percentage points of Time at Goal, with at least 95% simultaneous coverage.

Effect of the number of planned subgoals. Taken together, the theoretical decomposition and Pointax evaluation provide a way to assess the execution benefit of compositional planning with the learned policy and models held fixed. Under TC scoring, the numerical lower bounds establish improved Time at Goal over direct-goal execution at $K = 2, 4, 6$. At $K = 4$, the certified gain is at least 1.52 percentage points, corresponding to approximately 15 additional steps at the goal per episode, while independent full episodes measure a mean gain of approximately 37 steps. The largest observed mean occurs at $K = 4$, although the paired comparisons between depths remain inconclusive. This distinction is reflected in the theoretical bound: increasing K enlarges the available composition family, while the performance of the executed planner also depends on the commitments it selects and their accumulated shortfalls. The results therefore support the benefit of compositional execution at this checkpoint without identifying a uniquely preferable number of subgoals. They complement the candidate-selection analysis by quantifying the environment reward obtained after planning and replanning. During online training, increased goal occupancy would provide more reward-bearing transitions over the same environment-step budget, offering a concrete mechanism through which planning can change the experience available to the learner. The training experiments assess how planner-collected experience subsequently affects policy improvement while the present analysis establishes the execution-level gain that can support that process.

E EXPERIMENTAL DETAILS

All methods in Section 4 share one online learning loop, one goal-conditioned policy, and one evaluation protocol. They differ in how collection actions are chosen and in the auxiliary models this requires. Planning, retrieval, and subgoal selection are used only to collect training data, and every reported curve evaluates the learned policy alone. This section states the shared protocol once (Appendices E.1 and E.2) and then lists, for OGPC (Appendix E.3) and for each baseline (Appendices E.4–E.6), only what the method adds or changes. The tables give the default settings.

E.1 BENCHMARK AND EVALUATION PROTOCOL

We use nine JaxGCRL environments (Bortkiewicz et al., 2025) in two groups (Table 7). Table 8 lists the state, action, and goal dimensions, the state coordinates that define the goal, the goal radius, and the box within which the planner may place subgoals. The goal g of an episode is drawn at reset and held fixed. Maze goals are uniform over the goal cells of the layout.

Table 7: Training protocol for the tasks in Figures 4 and 5.

Setting	Figure 4	Figure 5
Environment steps	50M	100M
Seeds per method	10	5
Policy residual depth	8	64
Training / evaluation environments	1,024 / 128	512 / 128
Evaluation frequency	Every ≈ 0.83M steps	Every ≈ 0.98M steps
Methods compared	All seven methods	CRL, OGPC-TC, OGPC-MC, DCP, SF^2

Table 8: Environment dimensions and goal definitions. The dimensions (d_s, d_a, d_g) are for states, actions, and goals. Success requires the Euclidean distance between the listed goal coordinates and g to be less than the goal radius. Subgoal bounds apply coordinatewise and can extend beyond the support of sampled task goals.

Task	(d_s, d_a, d_g)	Goal coordinates	Goal radius	Subgoal bounds	
				Lower	Upper
Ant U-Maze	$(29, 8, 2)$	torso xy	0.5	$(0, 0)$	$(16, 16)$
Ant Big Maze	$(29, 8, 2)$	torso xy	0.5	$(0, 0)$	$(28, 28)$
Ant Hardest Maze	$(29, 8, 2)$	torso xy	0.5	$(0, 0)$	$(32, 44)$
Ant Push	$(31, 8, 2)$	torso xy	0.5	$(-1, -1)$	$(19, 19)$
Pusher Easy	$(20, 7, 3)$	object xyz	0.1	$(-1, -1, -0.4)$	$(1, 1, -0.2)$
Arm BinPick Easy	$(18, 5, 3)$	object xyz	0.1	$(-0.24, 0.51, 0.02)$	$(0.26, 0.69, 0.08)$
Humanoid	$(268, 17, 3)$	torso xyz	0.5	$(-5, -5, 1.25)$	$(5, 5, 1.25)$
Humanoid U-Maze	$(268, 17, 3)$	torso xyz	0.5	$(0, 0, 1.25)$	$(8, 8, 1.25)$
Humanoid Big Maze	$(268, 17, 3)$	torso xyz	0.5	$(0, 0, 1.25)$	$(14, 14, 1.25)$

Metrics Evaluation runs one 1,000-step episode in each of 128 evaluation environments using the deterministic goal-conditioned policy alone, without planning or subgoal selection. *Success Rate* is the fraction of episodes that reach the goal region at least once. *Time at Goal* is the mean number of steps spent in the goal region per episode. Learning curves show unsmoothed evaluations. Final performance summaries average the last five scheduled evaluations within each seed before aggregating across seeds. Unless a caption states otherwise, intervals are 95% bootstrap intervals with the training seed as the unit, paired across methods where seeds are shared.

E.2 Shared online loop and goal-conditioned policy

Every method collects data in 62-step blocks from all training environments in parallel and performs its learning updates after each block. The replay buffer holds $1{,}000$ transitions per parallel training environment, and updates begin once each environment has contributed at least $1{,}000$ transitions. OGPC-TC, OGPC-MC, OGPC-SS, and GC-TD-MPC2 begin planning after $2{,}000$ steps per environment. Until then, the stochastic goal-conditioned policy collects data alone. DCP begins retrieval after replay prefill (Appendix E.4), whereas CRL and SF^2 collect with the policy throughout training.

After each collection block, we update the policy first and then each method's auxiliary models. The next collection block uses these updated models.

Matrix multiplications use bfloat16 compute precision, while the parameters and optimizer states are stored in float32. We use the same optimizer Adam (Kingma & Ba, 2015) for all methods, with its default moment decay rates.

The goal-conditioned policy is the residual CRL actor and contrastive critic of Wang et al. (2025),[6] trained with hindsight-relabeled goals (Table 9). The networks use LayerNorm (Ba et al., 2016) and SiLU activations (Hendrycks & Gimpel, 2016; Elfwing et al., 2018).

For the Humanoid tasks, we use 512 parallel training environments and increase the actor and critic depth to 64. Policy training uses minibatches of 512 and 800 updates per collection block. The OGPC's GHM and terminal head also use minibatches of 512. For the Humanoid OGPC-TC runs on RTX 5090s, we distribute the learning minibatches over two GPUs with replicated parameters to fit in memory. Runs on an RTX PRO 6000 Blackwell use one GPU, as do the Humanoid CRL, DCP, and SF^2 runs on RTX 5090s. Table 18 lists the hardware used for each method and task.

Table 9: Goal-conditioned policy (CRL) and shared update schedule.

Hyperparameter	Value
Actor and critic architecture	Residual, width 256, depth 8 (Table 7)
Residual block structure	Four dense layers per block
Normalization / activation	LayerNorm / SiLU
Contrastive representation dimension	64
Contrastive objective / energy	Forward InfoNCE / negative Euclidean distance
Log-sum-exp penalty coefficient	0.1
Actor distribution	Tanh-squashed Gaussian
Target entropy	$-0.5 \dim(\mathcal{A})$
Discount and hindsight discount	0.99
Actor / critic / entropy learning rate	3×10^{-4}
Minibatch size	256
Update-to-data ratio	1:16
Collection / evaluation policy	Stochastic / deterministic

E.3 OGPC

OGPC trains a GHM and, for OGPC-TC, a terminal conditional occupancy head from the shared replay buffer. Its planner (Algorithm 3) uses iCEM over action–subgoal sequences. Appendix C.2 credits the source implementations and describes the optimizer conventions used here.

E.3.1 Geometric horizon model

The GHM is the discount-conditioned TD-Flow of Appendix B.3, trained with the TD-HC objective equation 27 (Table 10). Half of each minibatch relabels the command g with a replay goal so that the GHM covers the commands the planner issues. We draw β per example. For a randomly selected 15% of each minibatch, we draw a shorter local horizon $\omega < \beta$ to train the horizon-consistency

[6] https://github.com/wang-kevin3290/scaling-crl

objective. The remaining examples use $\omega = \beta$, which recovers the original objective of the TD-Flow. A separate, non-overlapping 10% of the minibatch has its action and command inputs masked and uses the recorded next action from replay for bootstrapping, rather than an action sampled from π_g. These examples train the GHM used for subgoal proposals (Appendix C). The GHM uses FiLM conditioning (Perez et al., 2018) and Mish activations (Misra, 2020).

Table 10: GHM architecture and optimization hyperparameters.

Hyperparameter	Value
Model	Conditional velocity field $v_u^\vartheta(x \mid s, a, g, \beta)$
Flow-time embedding	256 (sinusoidal), MLP $(1024, 1024)$
Discount embedding	256 (sinusoidal) on $(\beta,\ 1-\beta,\ -\log(1-\beta))$, MLP $(1024, 1024)$
Condition encoder MLP	$(1024, 1024, 1024)$
Processing blocks	Three FiLM residual blocks of width 1024
Hidden activation / normalization	Mish / encoder LayerNorm and pre-norm blocks
Output initialization	Zero
Discount sampling	$\beta \sim \mathcal{U}[0.90, 0.996]$
Local-horizon sampling (TD-HC examples)	$\omega \sim \mathcal{U}[0.90, \beta]$
Local horizon (remaining examples)	$\omega = \beta$
Horizon-consistency / masked-input fraction	0.15 / 0.10 (non-overlapping)
Target-network Euler steps	10
Target-network EMA update rate	5×10^{-4}
Optimizer / learning rate	Adam / 10^{-4}
Minibatch size / update-to-data ratio	256 / 1:16
Command relabeling fraction	0.5
State normalizer	Frozen after 500 warm-up updates
Gradient clipping	None

E.3.2 Terminal conditional occupancy head

We implement the head q_ϕ as an ensemble of $n_{\text{head}} = 5$ MLPs with separate parameters and random initializations. The planner averages their predicted probabilities as follows:

$$q_\phi(s, a, g_K, g^\star, \gamma) := \frac{1}{n_{\text{head}}} \sum_{e=1}^{n_{\text{head}}} q_{\phi_e}(s, a, g_K, g^\star, \gamma). \tag{80}$$

State-action pairs (s, a) are sampled from replay. The policy command g_K is either the executed command or a hindsight-achieved goal, using the same relabeling rule as the GHM. The task goal $g^\star$ is sampled independently. For each tuple, we draw L endpoints $\widetilde{S}_\ell^+ \sim \widehat{m}_\gamma^{\pi_{g_K}}(\cdot \mid s, a)$ from the current GHM and form the soft target

$$\widehat{y} := \frac{1}{L} \sum_{\ell=1}^{L} r_{g^\star}(\widetilde{S}_\ell^+). \tag{81}$$

All five networks are trained online by binary cross-entropy against the same target (Algorithm 7). Training uses the task discount γ, matching the planner's terminal query. Table 11 gives the architecture and training settings.

E.3.3 Planner and commitment

We provide the settings for the search procedure used in Algorithm 3. The execution commitment is $\tau_1 \sim \min\{\text{Geom}(1 - 0.95), \tau_{\max}\}$, with $\tau_{\max} = 59$ environment steps. This distribution stays fixed when we vary K. At $K = 1$, its discount differs from the sole GHM discount $\beta_1 = \gamma = 0.99$. For the default $K = 4$ configuration, the first-hop and execution discounts both equal 0.95.

Table 11: Terminal conditional occupancy head hyperparameters (OGPC-TC only).

Hyperparameter	Value
Inputs	$(s, a, g_K, g^\star, \gamma, -\log(1-\gamma))$
Ensemble size n_{head} / hidden layers per member	5 / $(256, 256)$
Normalization / activation / output	LayerNorm / SiLU / sigmoid
Training target	Goal-hit frequency over $L = 16$ GHM samples
Training discount	$\gamma = 0.99$
Command relabeling fraction	0.5
Optimizer / learning rate	Adam / 3×10^{-4}
Minibatch size / updates per block	256 / 62
Global gradient-norm clipping	10
Planner readout	Ensemble mean

Table 12: Default collection-time planner and commitment hyperparameters.

Hyperparameter	Value
Number of planned subgoals K	4
Optimized variables	$(a, g_{1:K})$
iCEM population / rounds / elites	12 / 4 / 3
Candidate evaluations per boundary plan	48
Simulation rows per candidate	64
Planner GHM Euler steps	5
Discount schedule $\beta_{1:K}$	$(0.95, 0.95, 0.95, 0.99)$
Composition weights $w_{1:K}$	$(0.200, 0.160, 0.128, 0.512)$
GHM proposal distribution	Unconditional (Algorithm 4)
Structured first-round candidates	Direct-goal plan and two GHM chains
Colored-noise exponent	2
iCEM momentum / population decay	0.1 / 1.0
Elite reuse fraction	1.0
Initial normalized variance / variance floor	0.25 / 0
Action bounds	$[-1, 1]$ in each dimension
Subgoal bounds	Table 8
Commitment distribution	$\min\{\text{Geom}(1-0.95), \tau_{\max}\}$
Maximum commitment $\tau_{\max}$	59 environment steps
Expected commitment length	19 primitive action steps
Replanning condition	Commitment boundary or episode reset

E.3.4 VARIANTS

OGPC-MC shares the GHM, planner, budget, and commitment schedule of OGPC-TC and differs in one line of the scorer (Algorithm 6). OGPC-MC samples the terminal GHM hop and evaluates the literal goal indicator instead of querying the learned terminal head, so it requires one additional GHM hop per simulation row.

OGPC-SS sets $\alpha_1 = \ldots = \alpha_{K-1} = 1$, so the planner optimizes a sequence of $H_{\text{SS}} = 16$ primitive actions through a separately trained one-step model (Table 13). The one-step model is a conditional flow $\widehat{P}(\cdot \mid s, a)$ with the GHM's architecture minus the goal and discount branches, trained by conditional flow matching on executed transitions. Starting from $\widetilde{S}_0 = s$, a candidate $a_{1:H_{\text{SS}}}$ is scored using one model trajectory, with $\widetilde{S}_h \sim \widehat{P}(\cdot \mid \widetilde{S}_{h-1}, a_h)$:

$$\widehat{Q}^{\text{SS}}(a_{1:H_{\text{SS}}}) = (1-\gamma) \sum_{h=1}^{H_{\text{SS}}-1} \gamma^{h-1} r_{g^\star}(\widetilde{S}_h) + \gamma^{H_{\text{SS}}-1} q_\phi(\widetilde{S}_{H_{\text{SS}}-1}, a_{H_{\text{SS}}}, g^\star, g^\star, \gamma), \tag{82}$$

so the prefix uses the literal sparse reward and the horizon is closed by the same head as OGPC-TC. The planner uses the iCEM budget of Table 12, replans at every step, and executes only a_1.

Table 13: OGPC-SS settings that differ from OGPC-TC.

Hyperparameter	Value
Optimized variables	Primitive actions $a_{1:16}$
Model / trajectories per candidate	One-step conditional flow / 1
One-step model architecture	GHM architecture without goal and discount branches
One-step model objective	Conditional flow matching on (s, a, s')
Optimizer / learning rate	Adam / 10^{-4}
Target-network EMA update rate	5×10^{-4}
Minibatch size / update-to-data ratio	256 / 1:16
Prefix reward / weights	Literal task success / $(1-\gamma)\gamma^{h-1}$, $h = 1, \ldots, 15$
Terminal term	$\gamma^{15}\, q_\phi$ with command $g_K = g^\star$ and $\gamma = 0.99$
Euler steps per predicted state	5
Replanning / executed actions	Every step / a_1

E.4 DCP

DCP (Swaminathan et al., 2026) conditions its actor on the state and on the normalized direction and distance to a target in the critic's goal-embedding space. During collection, the target is retrieved from replay. We retain the authors' direction features and retrieval rule, whose definitions and default settings are given in their source implementation.[7]

We adapt DCP to the shared CRL architecture and training protocol so that its backbone and base learning objective match those of the other methods (Appendix E.2). More specifically, we use residual networks with LayerNorm and forward InfoNCE, rather than the shallow networks and backward InfoNCE used by the authors' experiment scripts. Table 14 lists the implementation differences. Actor training uses the shared hindsight-relabeled goals and does not backpropagate through the goal encoder. Evaluation uses the deterministic actor conditioned on the task goal, without retrieval. After replay prefill, we rebuild the retrieval pool before each collection block. Until the first valid pool is available, the actor pursues the task goal.

Table 14: DCP adaptations to the shared CRL training protocol. Unchanged direction features and retrieval settings follow the authors' implementation.

Component	Our adaptation
Actor and critic networks	Shared residual architecture (Table 9)
Contrastive objective	Forward InfoNCE instead of backward InfoNCE
Critic energy	Unsmoothed negative Euclidean distance, as in CRL
Environment and update schedule	Shared protocol (Appendix E.2)
Before the first valid pool	Task-goal actions instead of retrieval from a zero-initialized pool

E.5 SF^2

We adapt the successor-flow auxiliary objective of SF^2 (Shi et al., 2026) to CRL, using the authors' released implementation as the reference.[8] The auxiliary shares CRL's existing 64-dimensional state–action representation, rather than the 512-dimensional representation in the original implementation. We keep the base contrastive objective and add the auxiliary loss with coefficient 1.0 on the same critic minibatches. Its gradients update the state–action encoder and auxiliary flow network, but do not directly update the actor or goal encoder. Bootstrap actions are sampled from the goal-conditioned policy using the minibatch's hindsight-relabeled goal. CRL's actor objective, goal relabeling, and planner-free collection and evaluation remain unchanged.

The same auxiliary settings are used with the depth-8 and depth-64 residual networks (Table 15). We mask invalid successors at episode boundaries and truncations by zeroing their losses before averag-

[7] https://anonymous.4open.science/r/dcp-supplement-anon-CF41/README.md

[8] https://github.com/Shiien/successor-flow-representation-implementation

ing over the full minibatch. Only valid successors contribute to the running normalization statistics. Target generation uses one midpoint integration step, and we omit the original implementation's additional zero-length iteration.

Table 15: SF^2 integration with CRL and auxiliary settings.

Setting	Value
Successor representation	Shared CRL state–action embedding, dimension 64
Auxiliary flow network	Two hidden layers of width 512, output $d_s \times 64$
Auxiliary activation / normalization	ReLU / LayerNorm
Flow-time embedding	64-dimensional sinusoidal
Auxiliary coefficient	1.0
Successor discount	0.99
Bootstrap action	$a' \sim \pi(\cdot \mid s', g)$, relabeled goal of the minibatch
Target networks	State–action encoder and auxiliary flow network
Target-network EMA update rate	0.005
Target flow integration	One midpoint step (two velocity evaluations)
Successor normalization	Running statistics over valid successor rows
Invalid successors	Episode boundaries and truncations masked
Auxiliary / shared critic learning rate	10^{-4} / 3×10^{-4}
Update frequency	One auxiliary update per critic minibatch

E.6 GC-TD-MPC2

E.6.1 Implementation

GC-TD-MPC2 plans over primitive actions using a learned latent dynamics model. We adapt ZILOT's goal-conditioned TD-MPC2 (Rupf et al., 2025) to the shared online training loop of Appendix E.2. During collection, the planner executes the first planned action and replans at every environment step. Evaluation uses the learned CRL policy without planning. Table 16 summarizes the adaptations.

Model and training We follow the networks and latent-model training objectives in the ZILOT implementation,[9] using the JAX TD-MPC2 implementation as a reference.[10] State and goal encoders map observations into latent space, where a dynamics model predicts subsequent states. A detached state decoder maps these predictions back to observation coordinates. Bootstrap actions come from the shared CRL policy rather than a separate latent actor. The Q ensemble supplies targets for the goal-conditioned value function V, which estimates the negative remaining time to reach the goal. The reward head contributes an auxiliary training loss but does not enter the planner's score. We retain the network widths and depths used in ZILOT's implementation, with SimNorm latent normalization (Hansen et al., 2024). Table 17 gives the model and training settings.

Plan scoring The planner searches over $H = 16$ primitive actions with the shared iCEM settings in Table 12. Each simulated step incurs a unit cost until a decoded state satisfies the task's success criterion. If the goal has not been reached within the planning horizon, the terminal value accounts for the remaining time. Writing z_h for the latent state at step h, let m_h equal one if none of the decoded states from z_0 through z_h satisfies the success criterion, and zero otherwise. The planner maximizes

$$J_{\text{GC-TD-MPC2}}(a_{1:H}) = -\sum_{h=0}^{H-1} m_h + m_H\, V(z_H, z_g), \tag{83}$$

where z_g is the encoded task goal.

[9] https://github.com/martius-lab/zilot

[10] https://github.com/ShaneFlandermeyer/tdmpc2-jax

Table 16: **GC-TD-MPC2 adaptations.** Both implementations use iCEM with four rounds and a planning horizon of 16 primitive actions.

	GC-TD-MPC2 (Rupf et al., 2025)	This paper
Training data	Fixed offline dataset	Online collection
Bootstrap actions	Goal-conditioned latent actor	Shared CRL policy
Predicted goal arrival	Threshold on $-V(z_h, z_g)$	Task success test on decoded states
Candidates / elites per round	512 / 6	12 / 3
Evaluation	Planner acts at every step	CRL policy without planning

Table 17: **GC-TD-MPC2 settings.** Shared iCEM hyperparameters are given in Table 12.

Setting	Value
Latent / encoder / model hidden width	512 / 256 / 512
Encoder depth / latent normalization	Two layers / SimNorm, groups of 8
Dynamics, reward, Q, value MLPs	Two 512-unit hidden layers, LayerNorm, Mish
Q ensemble / dropout	5 heads / 0.01 on the first hidden layer
Distributional support	101 two-hot bins on symlog $[-10, 10]$
Training rollout / horizon weights	3 transitions / 0.5^h
Consistency / reward / Q coefficients	20 / 0.1 / 0.1
Hindsight relabeling	60% future ($\gamma = 0.99$), 20% next state, 20% random
Hitting-time target range	$[-1000, 0]$, minimum over a random pair of target heads
Learning rates	9×10^{-5} (encoders), 3×10^{-4} (all other)
Gradient-norm clipping / target Polyak	20 / 0.01
Minibatch size / update-to-data ratio	256 / 1:16
Planning horizon	16 primitive actions
Replanning frequency	Every environment step

E.6.2 Policy-prior configurations

The additional comparisons in Appendix F.8.1 use two configurations: *12 candidates (1 prior)* and *128 candidates (6 priors)*. A prior candidate follows the shared CRL policy through the latent model. Its first action uses the observed state, and later actions use decoded latent states. The smaller configuration includes one deterministic policy rollout. The larger configuration includes this rollout and five stochastic policy rollouts. The priors occupy slots within the stated population in the first search round, with all other search mechanics unchanged. Appendix F.8.2 also evaluates these priors on frozen models. Appendix F.8.3 reports their computational cost.

E.7 Compute

Table 18 reports the mean wall-clock time per run for every method and environment of Section 4, taken from the training process's runtime record or from the Slurm job duration.

Beyond CRL, OGPC-TC adds the GHM and head updates and one planner call per commitment boundary. With the default $K = 4$ configuration (Table 12), every call scores 48 candidates with 64 simulation rows each. OGPC-TC makes 3,072 head queries and samples 9,216 GHM prefix hops. OGPC-MC also samples the terminal hop, giving 12,288 hops in total. At five Euler steps per hop, scoring costs 46,080 (TC) or 61,440 (MC) GHM velocity-field evaluations per call. The two GHM-chain proposals add 40 more evaluations.

Table 18: Mean wall-clock hours per run for the main experiments. Runs use 50M transitions on the Ant and manipulation tasks and 100M on the Humanoid tasks. Seed counts and GPU configurations are listed for each method and task.

Environment	Method	Seeds	GPU	GPUs	Wall-clock (h)
Ant U-Maze	CRL	10	RTX 5090	1	0.8
	OGPC-TC	10	RTX 5090	1	11.7
	OGPC-MC	10	RTX 5090	1	14.5
	OGPC-SS	10	RTX 3090	1	37.5
	DCP	10	RTX 3090	1	3.3
	SF^2	10	RTX 3090	1	2.5
	GC-TD-MPC2	10	RTX 3090	1	10.7
Ant Big Maze	CRL	10	RTX 5090	1	0.8
	OGPC-TC	10	RTX 5090	1	11.7
	OGPC-MC	10	RTX 5090	1	14.6
	OGPC-SS	10	RTX 3090	1	37.8
	DCP	10	RTX 3090	1	3.5
	SF^2	10	RTX 3090	1	2.7
	GC-TD-MPC2	10	RTX 3090	1	10.9
Ant Hardest Maze	CRL	10	RTX 5090	1	0.9
	OGPC-TC	10	RTX 5090	1	11.8
	OGPC-MC	10	RTX 5090	1	14.6
	OGPC-SS	10	RTX 3090	1	38.0
	DCP	10	RTX 3090	1	3.7
	SF^2	10	RTX 3090	1	3.0
	GC-TD-MPC2	10	RTX 3090	1	11.2
Pusher Easy	CRL	10	RTX 3090	1	2.0
	OGPC-TC	10	RTX 5090	1	11.7
	OGPC-MC	10	RTX 5090	1	14.5
	OGPC-SS	10	RTX 5090	1	12.1
	DCP	10	RTX 3090	1	3.6
	SF^2	10	RTX 3090	1	2.7
	GC-TD-MPC2	10	RTX 5090	1	4.6
Ant Push	CRL	10	RTX 5090	1	0.9
	OGPC-TC	10	RTX 5090	1	11.8
	OGPC-MC	10	RTX 5090	1	14.7
	OGPC-SS	10	RTX 5090	1	12.3
	DCP	10	RTX 3090	1	3.7
	SF^2	10	RTX 3090	1	2.7
	GC-TD-MPC2	10	RTX 5090	1	4.7
Arm BinPick Easy	CRL	10	RTX 5090	1	3.1
	OGPC-TC	10	RTX 5090	1	14.1
	OGPC-MC	10	RTX 5090	1	16.8
	OGPC-SS	10	RTX 5090	1	14.5
	DCP	10	RTX 3090	1	12.3
	SF^2	10	RTX 3090	1	10.0
	GC-TD-MPC2	10	RTX 5090	1	7.0
Humanoid	CRL	5	RTX 5090	1	5.2
	OGPC-TC	2	RTX 5090	2	80.9
	OGPC-TC	3	RTX PRO 6000 Blackwell	1	54.0
	OGPC-MC	5	RTX PRO 6000 Blackwell	1	48.2
	DCP	5	RTX 5090	1	5.6
	SF^2	5	RTX 5090	1	7.0
Humanoid U-Maze	CRL	5	RTX 5090	1	5.9
	OGPC-TC	5	RTX 5090	2	81.1
	OGPC-MC	5	RTX PRO 6000 Blackwell	1	48.5
	DCP	5	RTX 5090	1	5.8
	SF^2	5	RTX 5090	1	7.3
Humanoid Big Maze	CRL	5	RTX 5090	1	6.3
	OGPC-TC	5	RTX 5090	2	83.8
	OGPC-MC	5	RTX PRO 6000 Blackwell	1	51.7
	DCP	5	RTX 5090	1	6.3
	SF^2	5	RTX 5090	1	8.1

F ADDITIONAL RESULTS

This appendix supplements Section 4. Unless stated otherwise, we report 95% seed-bootstrap confidence intervals using 10,000 resamples.

F.1 HEADLINE RESULTS: SUCCESS RATE AND FINAL PERFORMANCE

Figure 14 complements Figure 4 with Success Rate curves. Figures 15 and 16 show final performance across seeds on the six 50M-step tasks, summarized in Table 19. Figures 17–19 and Table 20 provide the corresponding Humanoid results at 100M steps. OGPC-TC has the highest mean final Success Rate and Time at Goal on five of the six 50M-step tasks. On Arm BinPick Easy, SF^2 has the highest means for both metrics, with confidence intervals overlapping those of OGPC-TC.

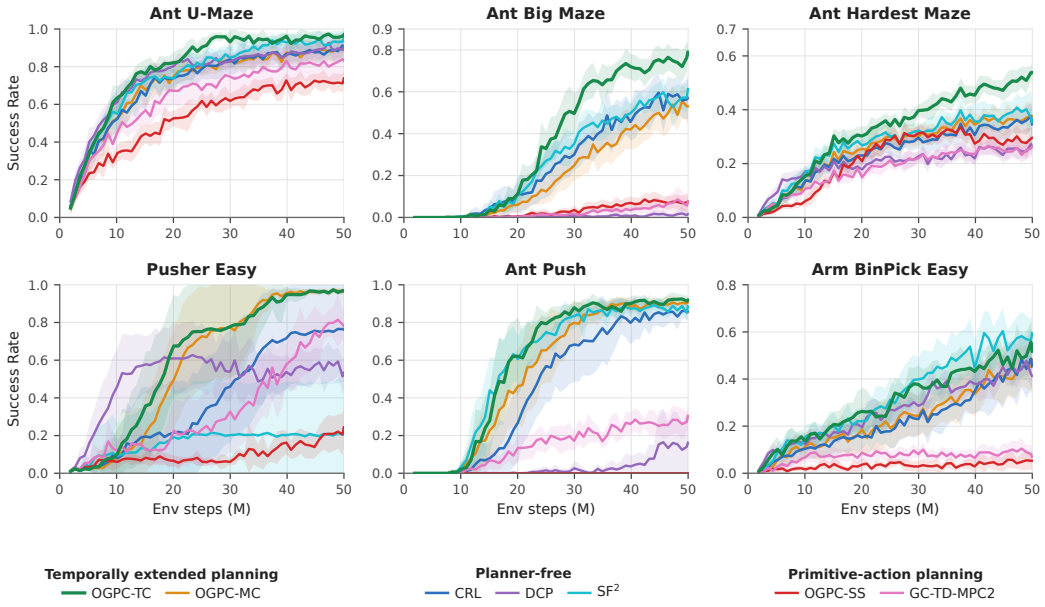


Figure 14: **Planner-free Success Rate during training on six JaxGCRL tasks.** Same runs as Figure 4. Unsmoothed means and pointwise 95% seed-t confidence intervals over 10 seeds.

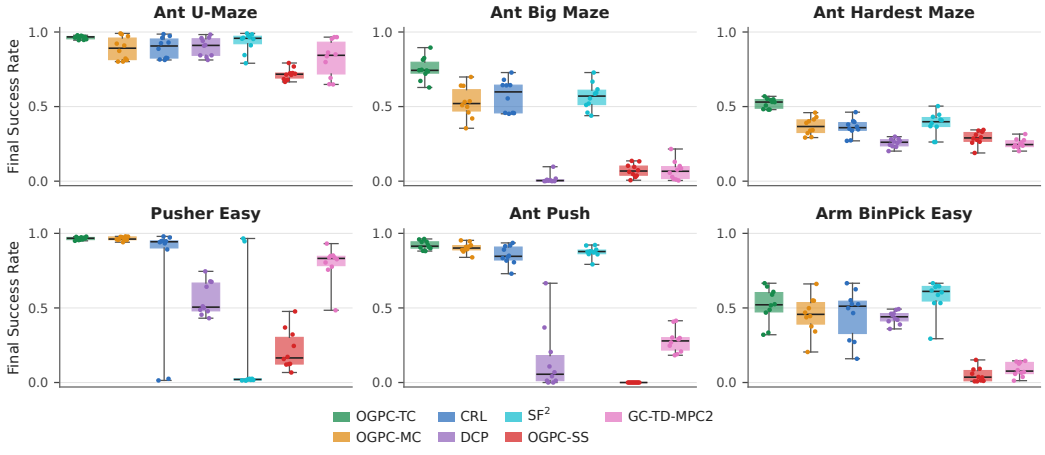


Figure 15: **Final Success Rate across seeds.** Mean of each seed's last five planner-free evaluations at 50M steps, with 10 seeds per method. Points show individual seeds, boxes show the interquartile range, horizontal lines mark medians, and whiskers span the full range.

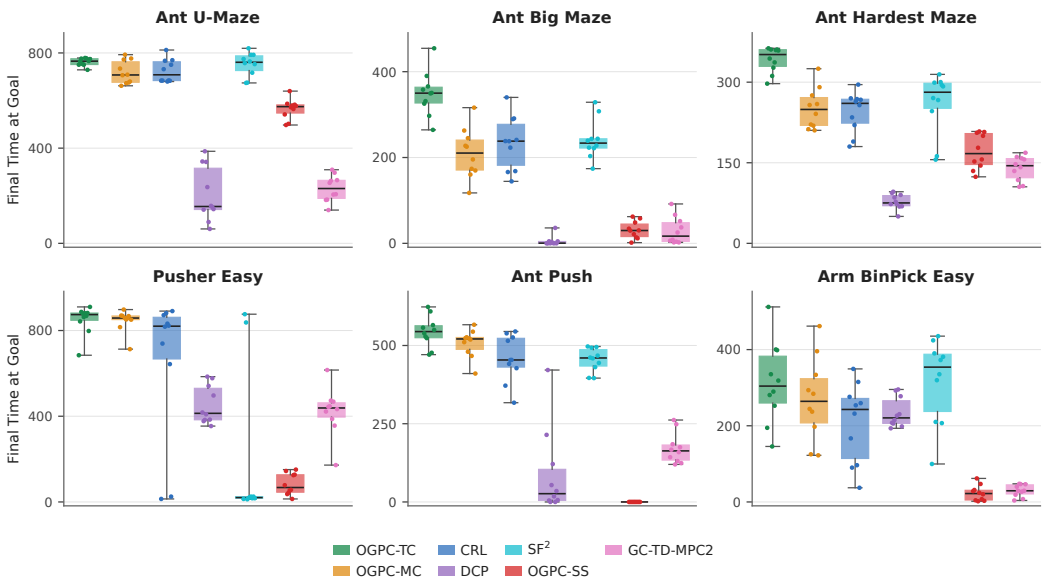


Figure 16: **Final Time at Goal across seeds.** Same runs and box-plot conventions as Figure 15. Time at Goal counts steps in the goal region during a 1,000-step episode.

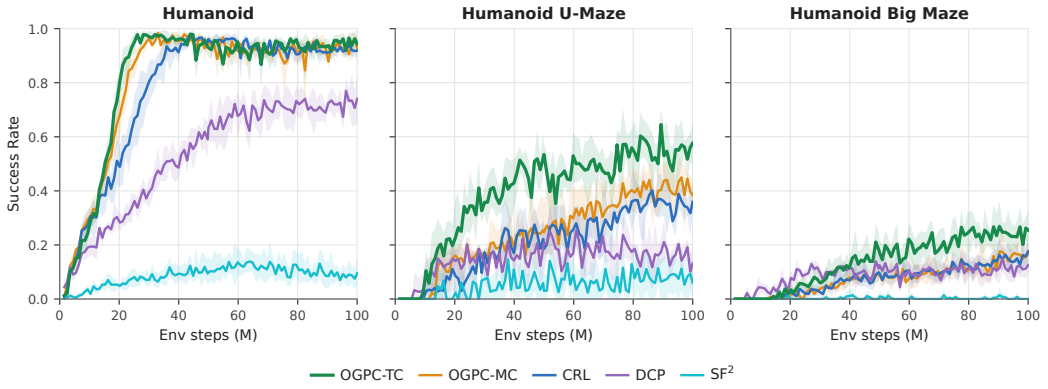


Figure 17: **Planner-free Success Rate on the Humanoid tasks.** Same runs as Figure 5. Unsmoothed mean and pointwise 95% seed-bootstrap confidence interval over 5 seeds per method.

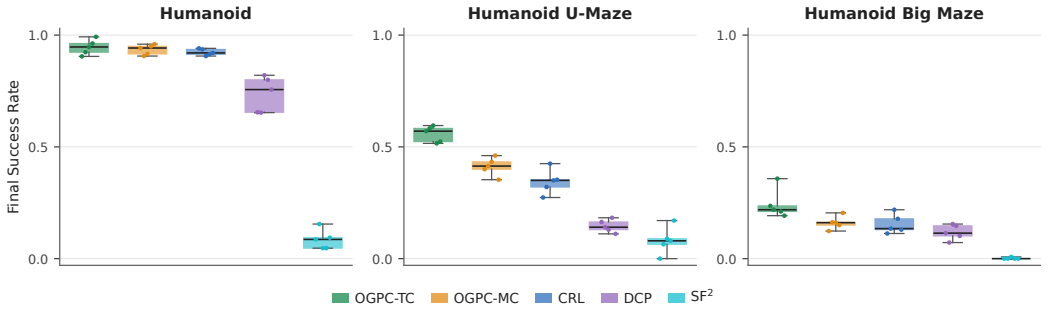


Figure 18: **Final Success Rate across seeds on the Humanoid tasks.** Mean of each seed's last five planner-free evaluations at 100M steps, with five seeds per method. Same runs as Figure 17, using the box-plot conventions of Figure 15.

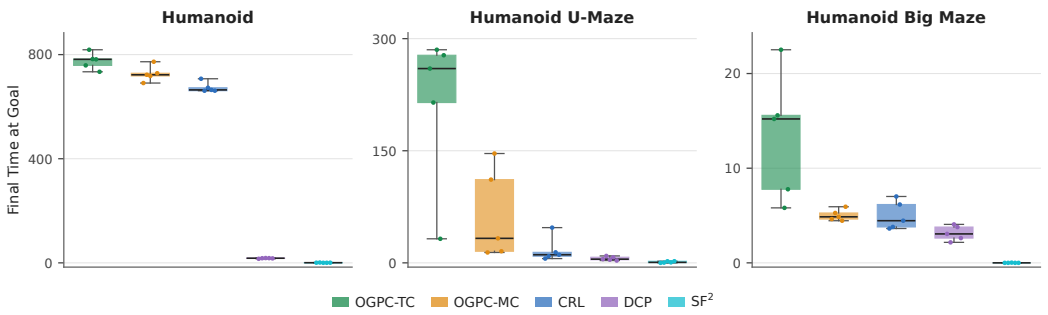


Figure 19: **Final Time at Goal across seeds on the Humanoid tasks.** Same 100M-step evaluations and box-plot conventions as Figure 18. Vertical scales differ between tasks.

Table 19: **Final Success Rate and Time at Goal on the six 50M-step tasks.** Mean over 10 seeds of each seed's last five planner-free evaluations. Brackets are 95% seed-bootstrap intervals. Time at Goal counts steps in the goal region during a 1,000-step episode. Bold marks the highest mean for each task and metric, before rounding.

	Ant U-Maze	Ant Big Maze	Ant Hardest Maze	Pusher Easy	Ant Push	Arm BinPick Easy
Final Success Rate						
OGPC-TC	**0.96** [0.96, 0.97]	**0.75** [0.71, 0.80]	**0.52** [0.50, 0.54]	**0.97** [0.96, 0.97]	**0.92** [0.90, 0.94]	0.52 [0.44, 0.58]
OGPC-MC	0.89 [0.84, 0.93]	0.53 [0.47, 0.59]	0.37 [0.33, 0.40]	0.96 [0.96, 0.97]	0.90 [0.88, 0.92]	0.45 [0.38, 0.53]
OGPC-SS	0.72 [0.70, 0.74]	0.07 [0.05, 0.10]	0.29 [0.26, 0.31]	0.22 [0.15, 0.30]	0.00 [0.00, 0.00]	0.05 [0.02, 0.08]
CRL	0.89 [0.85, 0.94]	0.57 [0.51, 0.64]	0.36 [0.32, 0.39]	0.76 [0.49, 0.95]	0.85 [0.82, 0.89]	0.46 [0.36, 0.55]
DCP	0.90 [0.86, 0.94]	0.01 [0.00, 0.03]	0.26 [0.24, 0.27]	0.56 [0.50, 0.63]	0.15 [0.04, 0.29]	0.44 [0.41, 0.46]
SF^2	0.93 [0.89, 0.97]	0.57 [0.52, 0.62]	0.38 [0.34, 0.43]	0.21 [0.02, 0.49]	0.88 [0.85, 0.89]	**0.58** [0.50, 0.63]
GC-TD-MPC2	0.82 [0.75, 0.89]	0.07 [0.04, 0.11]	0.25 [0.23, 0.27]	0.80 [0.72, 0.85]	0.28 [0.23, 0.33]	0.09 [0.06, 0.12]
Final Time at Goal (steps)						
OGPC-TC	**762** [752, 771]	**349** [319, 380]	**342** [327, 355]	**850** [806, 881]	**544** [516, 573]	313 [252, 376]
OGPC-MC	718 [690, 747]	209 [176, 245]	251 [230, 275]	845 [812, 869]	507 [478, 531]	269 [208, 334]
OGPC-SS	564 [539, 588]	31 [20, 43]	171 [151, 190]	83 [54, 112]	0 [0, 0]	23 [12, 35]
CRL	725 [697, 754]	234 [199, 271]	245 [222, 266]	654 [434, 833]	459 [413, 502]	208 [145, 266]
DCP	205 [140, 277]	5 [1, 12]	78 [69, 86]	453 [402, 508]	87 [20, 179]	234 [213, 258]
SF^2	752 [722, 779]	241 [216, 270]	260 [225, 290]	187 [18, 436]	454 [431, 475]	**317** [250, 377]
GC-TD-MPC2	231 [198, 263]	30 [12, 49]	140 [126, 153]	422 [353, 482]	171 [145, 202]	30 [20, 39]

Table 20: **Final Success Rate and Time at Goal on the Humanoid tasks.** Mean over five seeds of each seed's last five planner-free evaluations ending at 100M environment steps. Same evaluations as Figures 18 and 19. Brackets are 95% seed-bootstrap intervals. Bold marks the highest mean for each task and metric, before rounding.

	Humanoid	Humanoid U-Maze	Humanoid Big Maze
Final Success Rate			
OGPC-TC	**0.95** [0.92, 0.97]	**0.56** [0.53, 0.59]	**0.24** [0.20, 0.30]
OGPC-MC	0.93 [0.92, 0.95]	0.41 [0.38, 0.44]	0.16 [0.14, 0.18]
CRL	0.92 [0.91, 0.93]	0.34 [0.30, 0.39]	0.15 [0.12, 0.19]
DCP	0.74 [0.67, 0.80]	0.15 [0.12, 0.17]	0.12 [0.09, 0.14]
SF^2	0.09 [0.05, 0.12]	0.08 [0.03, 0.13]	0.00 [0.00, 0.00]
Final Time at Goal (steps)			
OGPC-TC	**775** [753, 799]	**214** [123, 277]	**13** [8, 18]
OGPC-MC	726 [705, 752]	64 [18, 113]	5 [5, 6]
CRL	673 [662, 691]	17 [8, 33]	5 [4, 6]
DCP	17 [17, 18]	6 [4, 8]	3 [3, 4]
SF^2	1 [0, 1]	1 [0, 2]	0 [0, 0]

F.2 GOAL-DISTANCE ANALYSIS ON ALL NAVIGATION TASKS

Figures 20 and 21 show the performance of the learned CRL and OGPC-TC policies at evaluation on shared start–goal queries, grouped by the distance to the goal, for Time at Goal and Success Rate respectively. We evaluate the frozen planner-free policies on shared goal banks at 50M steps (Ant, 10 paired seeds) and 100M steps (Humanoid tasks, 5 paired seeds). We compute the maze distance by counting the shortest-path grid edges (4 m spacing for Ant, 2 m for Humanoid). For the 'open-arena' Humanoid task, we use the initial horizontal distance to the goal in 1 m bins. Time at Goal intervals are pointwise 95% training-seed bootstrap intervals.

On Ant Hardest Maze, OGPC-TC yields greater Time at Goal over intermediate distances. We observe that the gap is small for nearby goals, and the most distant goals remain difficult for both policies. On Ant Big Maze it roughly doubles Time at Goal at distances 8–10, and on Ant U-Maze the difference is concentrated at the farthest goals. The Humanoid mazes show the same pattern at 100M steps. These results show that the OGPC-TC's advantage over the planner-free policies is primarily observed on challenging but attainable goals.

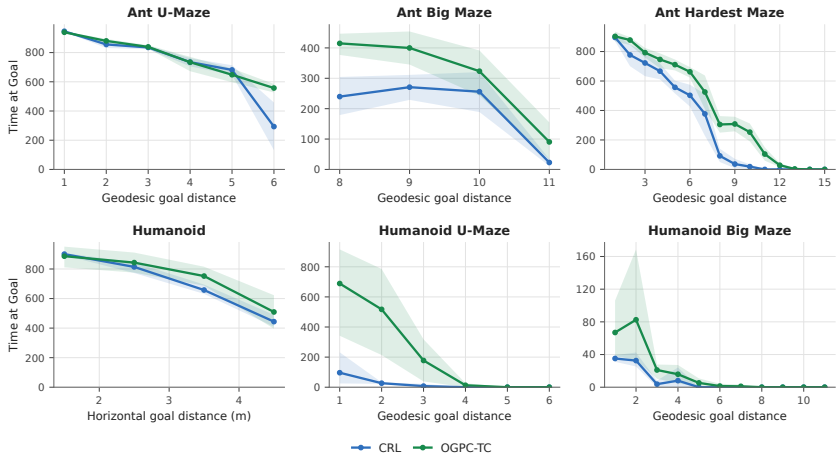


Figure 20: **Time at Goal by initial goal distance on all navigation tasks.** Planner-free evaluation of CRL and OGPC-TC by initial goal distance, for the Ant and Humanoid embodiments. Vertical scales differ between panels.

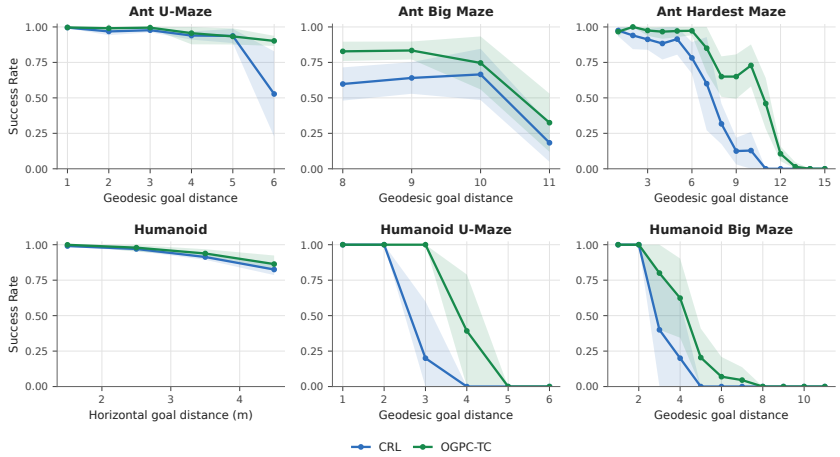


Figure 21: **Success Rate by initial goal distance on all navigation tasks.** An episode is successful if the goal is reached at least once. Time at Goal was measured in a separate evaluation.

F.3 Number of planned subgoals during learning and evaluation

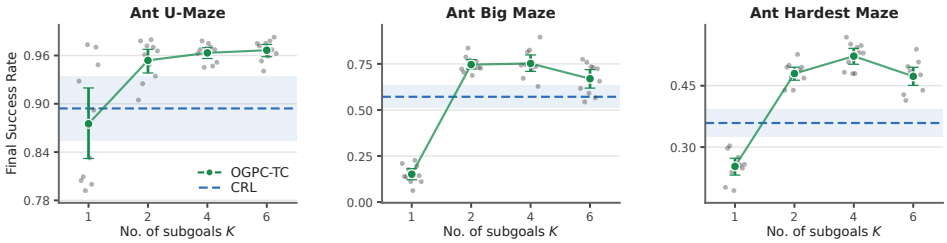


Figure 22: **Success Rate by number of subgoals used during training.** Final Success Rate for the runs in Figure 7, evaluated without planning. Markers show means with 95% seed-bootstrap intervals, and translucent dots show ten seeds. The dashed line and band show the CRL mean and interval.

Table 21: **Final performance by number of subgoals used during training.** Results underlying Figure 7, with Success Rate also reported. The $K = 4$ and CRL runs also appear in Section 4.1. We use the same code and seeds for all values of K. Values are means over 10 seeds, using each seed's last five planner-free evaluations ending at 50M steps. Brackets are 95% seed-bootstrap intervals. Bold marks the highest mean for each task and metric, before rounding.

Task	CRL	$K = 1$	$K = 2$	$K = 4$	$K = 6$
Final Time at Goal (steps)					
Ant U-Maze	725 [698, 754]	711 [686, 737]	753 [738, 768]	762 [752, 771]	**783** [774, 790]
Ant Big Maze	234 [198, 270]	48 [37, 58]	337 [312, 362]	**349** [320, 381]	310 [281, 337]
Ant Hardest Maze	245 [222, 266]	149 [135, 162]	324 [309, 336]	**342** [327, 355]	315 [302, 325]
Final Success Rate					
Ant U-Maze	0.89 [0.85, 0.93]	0.88 [0.83, 0.92]	0.95 [0.94, 0.97]	0.96 [0.96, 0.97]	**0.97** [0.96, 0.97]
Ant Big Maze	0.57 [0.51, 0.63]	0.15 [0.12, 0.18]	0.75 [0.72, 0.77]	**0.75** [0.71, 0.80]	0.67 [0.62, 0.72]
Ant Hardest Maze	0.36 [0.32, 0.39]	0.25 [0.23, 0.27]	0.48 [0.46, 0.50]	**0.52** [0.50, 0.54]	0.47 [0.45, 0.50]

Evaluation-time comparison. We freeze the policies and models trained with $K = 4$ at 50M steps and vary K only at evaluation (Figure 23). Increasing K from 4 to 6 improves Success Rate on Big Maze and Ant Push but not on Hardest Maze, and Pusher Easy and Arm BinPick Easy change little as K varies.

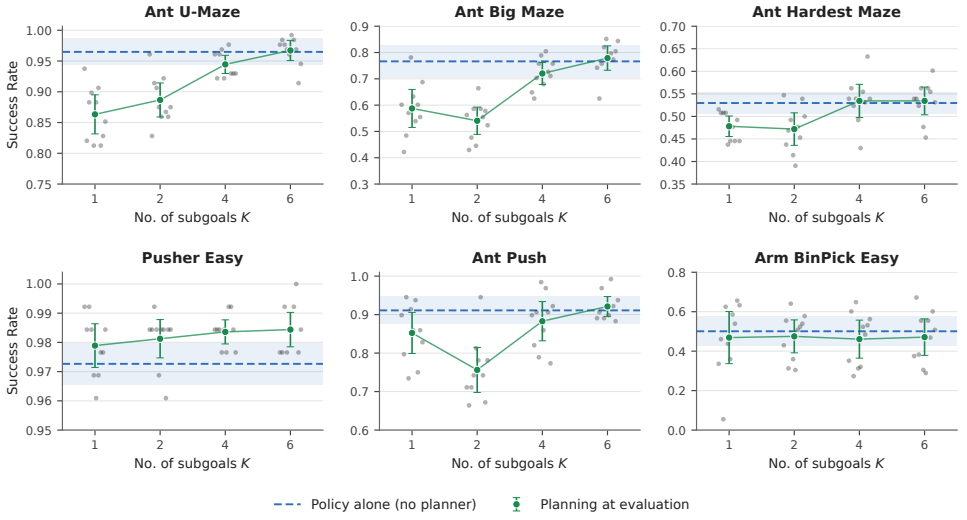


Figure 23: **Number of planned subgoals at evaluation.** Frozen-checkpoint Success Rate as K varies. Markers show means with 95% seed-t intervals, and translucent dots show ten seeds. The dashed line and band show the same policies evaluated without planning.

F.4 SUBGOAL-SELECTION ABLATION

We test whether randomly sampled GHM subgoals suffice to obtain OGPC-TC's learning gains without scored search. At each replanning boundary after warm-up, we randomly sample one subgoal sequence using OGPC's GHM proposal mechanism (Algorithm 4). The agent follows its policy conditioned on the first sampled subgoal g_1, using the same execution rule as OGPC. We do not score or refine the sequence, or compare it with the direct-goal plan. We keep the policy and model learning settings unchanged.

Figure 24 shows substantially lower final performance with unscored proposals on both tasks. Mean Success Rate falls from 78.4% to 13.3% on Ant Big Maze and from 96.5% to 20.7% on Pusher Easy. Three of five unscored Big Maze seeds finish at zero success. On Pusher, four remain below 2.1%, while one reaches 96.9%. GHM-generated subgoals and temporal commitment alone therefore do not recover OGPC-TC's gains. This comparison tests scored search and selective intervention together.

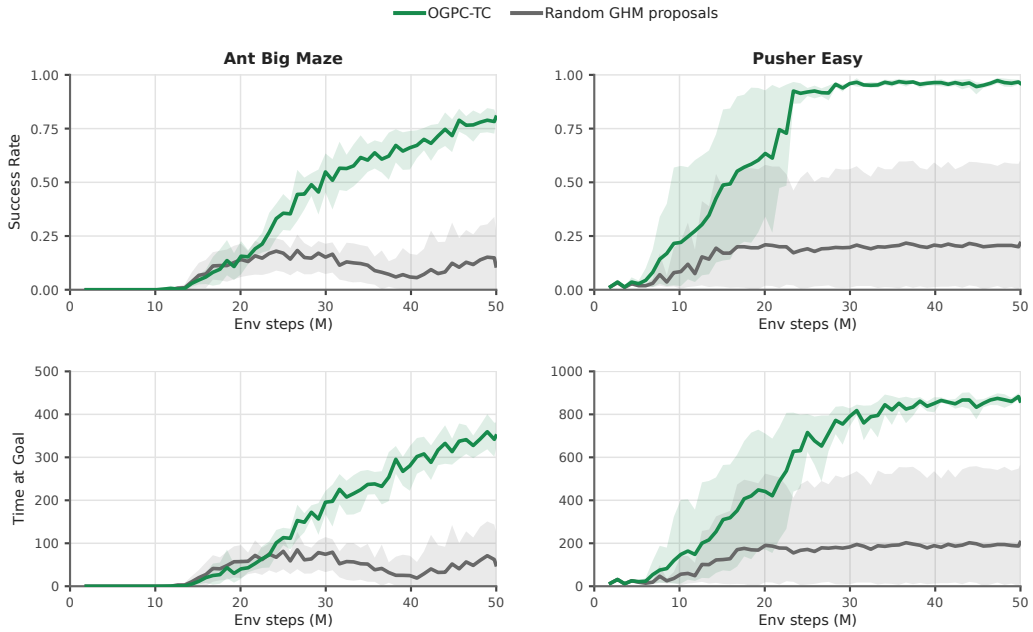


Figure 24: **Subgoal selection during training.** Planner-free Success Rate (top) and Time at Goal (bottom) throughout training, comparing OGPC-TC with random GHM proposals. Lines show unsmoothed means over five seeds, with pointwise 95% seed-bootstrap confidence bands. The OGPC-TC comparator is the five-seed longitudinal cohort. Final values in the text average the last five evaluations.

F.5 TERMINAL CONDITIONING: SCORING NOISE AND PLANNER DECISIONS

Theorem 1 predicts that exact terminal conditioning reduces scoring noise without changing current-GHM values. On frozen continuous-control models, we compare MC and TC using regret and SNR relative to each estimator's expected scores, then add terminal noise while preserving TC's expected scores. TC reaches lower own-objective regret at matched sampling budgets in these comparisons. Restoring noise increases errors in recovering the independently estimated TC score ordering. These diagnostics examine the relationship between scoring noise and candidate selection in continuous domains.

F.5.1 REGRET AND SNR RELATIVE TO EACH SCORING OBJECTIVE

We examine how reliably MC and TC optimize their own expected scores as the simulation budget increases. We freeze MC-trained GHM checkpoints on Ant Big Maze and compare literal scoring

with a transferred TC head on the same backbone. This controls the backbone at each checkpoint while retaining the learned head's approximation error. The comparison is the continuous counterpart of the regret plots in Figure 10. For estimator $E \in \{\mathrm{MC}, \mathrm{TC}\}$, write

$$\mu_i^E := \mathbb{E}[\widehat{Q}_i^E], \qquad i_E^\star \in \arg\max_i \mu_i^E, \qquad \delta_i^E := \mu_{i_E^\star}^E - \mu_i^E. \tag{84}$$

With $\hat{i}_{E,N}$ selected using N simulations per candidate, define

$$r_E(N) := \frac{\mathbb{E}[\mu_{i_E^\star}^E - \mu_{\hat{i}_{E,N}}^E]}{\mu_{i_E^\star}^E}. \tag{85}$$

The normalization requires a positive reference optimum. This regret measures finite-budget optimization of each scorer's own objective; MC and learned TC need not share the same optimum or value gaps. Their SNRs incorporate both their own gaps and their own comparison variances.

For a unique optimum and independent candidate simulations, set $V_i^E := \sigma_{i,E}^2 + \sigma_{i_E^\star,E}^2$ and $S_i^E := N(\delta_i^E)^2/V_i^E$. We report

$$\bar{S}_w^E := \frac{\sum_{i \neq i_E^\star} \delta_i^E S_i^E}{\sum_{i \neq i_E^\star} \delta_i^E}, \qquad x_E(N) := \sum_{i \neq i_E^\star} \frac{\delta_i^E/\mu_{i_E^\star}^E}{1 + S_i^E}. \tag{86}$$

Larger $\bar{S}_w^E$ indicates stronger planning SNR; smaller x_E indicates a tighter normalized regret bound. The Cantelli argument in Proposition 1 gives $r_E(N) \leq x_E(N)$ for the population quantities. This applies to either estimator relative to its own mean, without assuming that the TC head is unbiased for the GHM value. The arithmetic summary $\bar{S}_w^E$ is an empirical diagnostic, whereas x_E retains the nonlinear contributions entering the bound.

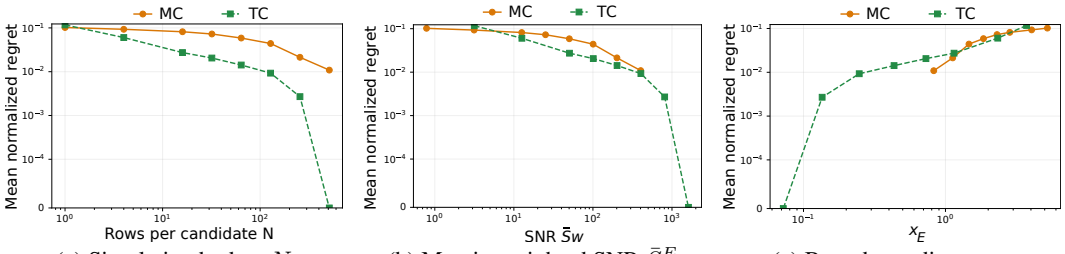
In the empirical evaluation, separate reference simulations estimate candidate means and the reference-best candidate; the planning samples used for selection are separate. Independent reference halves select and evaluate the reference anchor, with an exploratory split-agreement screen. Finite references approximate Equations 84– 86: they do not establish the population optimum or turn an estimated x_E into a confidence bound. Zero measured comparison variance is treated as unresolved, rather than as evidence of infinite SNR.

Single-checkpoint comparison with the tabular result. Figure 25 shows seed 4 at 15M steps on 14 matched query populations. Panels (a), (b), and (c) correspond to Figure 10(a), (c), and (d), respectively: the vertical quantity is normalized regret, and the horizontal coordinate changes from simulation budget to margin-weighted SNR and then to the regret-bound coordinate. TC has substantially lower own-objective regret at matched budgets. The curves move into closer alignment over their shared SNR range, with further alignment visible against x_E. This approximate collapse supports planning SNR as an explanation of ranking reliability within the checkpoint. The lower-regret range reached only by TC has no MC counterpart at the tested budgets, so it cannot be used to assess collapse.

Checkpoint sweeps and paired comparisons. Figure 26 repeats the three coordinates across training checkpoints. Each curve is a budget sweep at a frozen checkpoint, not a learning trajectory. TC generally reduces own-objective regret more rapidly with N and reaches higher margin-weighted SNR. The relationship between regret and SNR varies across checkpoints, as both the model and the candidate-score distributions change. Accordingly, a global collapse of all checkpoints is not required by the within-checkpoint explanation. The x_E coordinate organizes the curves more closely, but neither this alignment nor the SNR summary determines regret exactly.

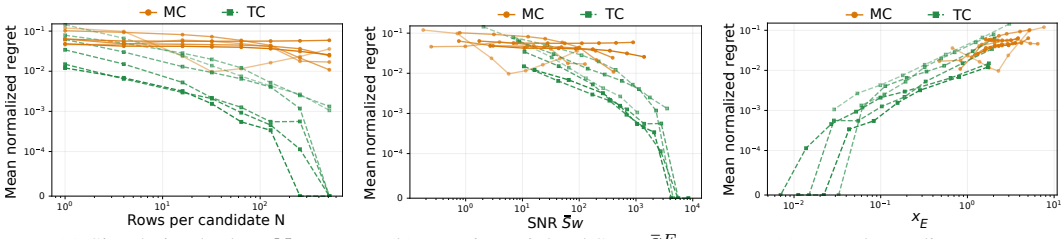
Figure 27 presents the paired version, matching the query/candidate populations used to compare MC and TC within each checkpoint. The qualitative pattern persists under this matching: TC reaches lower own-objective regret as the budget increases, and the SNR-based coordinates reduce some of the separation seen against N. Matching populations controls the comparison support, but it does not make the estimators' expected objectives identical. Residual differences can reflect head bias, the distribution of value gaps, and score tails.

These comparisons show that TC can resolve its own candidate preferences with fewer simulations, consistent with its higher planning SNR. Appendix F.5.2 isolates the effect of terminal sampling noise while preserving expected TC scores.



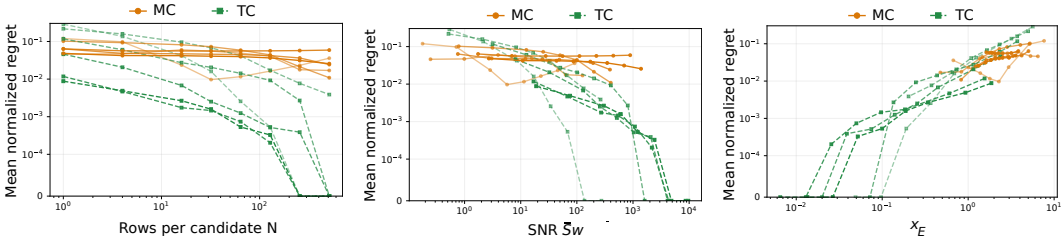
(a) Simulation budget N (b) Margin-weighted SNR $\bar{S}_w^E$ (c) Bound coordinate x_E

Figure 25: **Single-checkpoint counterpart of the tabular SNR analysis.** Ant Big Maze, MC-trained backbone with transferred TC head, seed 4 at 15M steps; 14 matched query populations. Left to right: normalized own-objective regret against N, $\bar{S}_w^E$, and $x_E(N)$, corresponding to Figure 10(a,c,d). Accounting for SNR brings MC and TC closer together over overlapping coordinates. Each estimator uses its own independently estimated reference objective; this tests ranking reliability, not which scorer selects higher environment returns.



(a) Simulation budget N (b) Margin-weighted SNR $\bar{S}_w^E$ (c) Bound coordinate x_E

Figure 26: **Own-objective ranking reliability across MC-trained checkpoints.** Each curve sweeps the simulation budget at one frozen Ant Big Maze checkpoint. Orange circles denote MC; green squares denote learned TC. The same regret measurements are shown against (a) N, (b) $\bar{S}_w^E$, and (c) x_E. Variation across curves reflects different checkpoints; alignment between estimators should be assessed within checkpoints and over overlapping horizontal ranges. The reference objective is specific to each scorer, as in Equation 85.



(a) Simulation budget N (b) Margin-weighted SNR $\bar{S}_w^E$ (c) Bound coordinate x_E

Figure 27: **Paired checkpoint comparison.** The three coordinates from Figure 26 with matched query/candidate populations for MC and TC within each checkpoint. The qualitative advantage in finite-budget optimization persists under pairing. Reference values and optima remain estimator-specific; paired populations do not imply a common scoring objective.

F.5.2 Ranking reliability at fixed expected scores

We test whether increasing terminal sampling noise causes more pairwise ranking errors while holding expected TC scores fixed.

We replace each normalized TC row by

$$Y_\lambda = R_{<K} + w_K\big[(1-\lambda)q_\phi + \lambda\widetilde{B}\big], \qquad \widetilde{B} \mid \mathcal{F}_{K-1} \sim \text{Bernoulli}(q_\phi), \tag{87}$$

with $\lambda \in \{0, \frac{1}{2}, 1\}$ for zero, partial, and full terminal noise. The conditional mean is unchanged, and the added row variance is $\lambda^2 w_K^2 \mathbb{E}[q_\phi(1-q_\phi)]$, so decision SNR decreases without changing the expected score gap.

We use 380 first-round candidate populations from frozen OGPC-TC checkpoints on Ant Big Maze and Pusher Easy, with five seeds per task at 10, 20, 30, 40, and 50M steps. Each population contains twelve candidates. We split the stored simulations into 4,096 reference and 4,096 evaluation rows. The reference estimates expected TC scores and selects a candidate for comparison. We retain comparisons whose ordering agrees between the reference halves and whose estimated gap exceeds a Bonferroni-adjusted normal-approximation threshold (5% level over 66 pairs). On the evaluation rows, we measure how often the lower-expected-score candidate is ranked ahead, using $N = 64$. Each 64-row block receives eight sets of Bernoulli draws. All noise levels use identical retained comparisons.

Mean pairwise ranking error increases from 0.04% without added noise to 0.18% with partial noise and 8.01% with full noise on Ant Big Maze (Figure 28). The corresponding values on Pusher Easy are 0.25%, 0.67%, and 11.56%. Both noise levels increase average error in every seed on both tasks. Partial noise increases errors without producing all-zero populations, showing that the effect extends beyond direct-goal fallback.

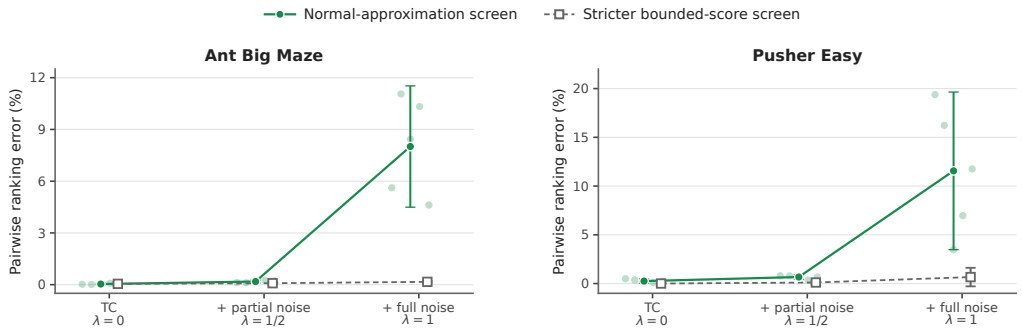


Figure 28: **Terminal noise increases pairwise ranking error.** Five-seed means and 95% seed-t intervals at $N = 64$, averaging 10–50M within seed. Filled circles and translucent dots show aggregate and individual-seed errors for the normal-approximation screen. Hollow squares additionally require non-overlapping bounded-score reference intervals. The two screens retain 2,199 and 1,516 of 2,200 comparisons on Big Maze, and 1,933 and 1,459 of 1,980 on Pusher, respectively.

Together with Appendix F.5.1, these results show that lower scoring noise enables more reliable comparisons for a fixed expected objective.

F.5.3 Online planning decisions

Figure 29 extends the planning-decision histories in Figure 6 to Pusher Easy. Both variants use 64 samples per candidate. On Pusher Easy, the initially low all-zero rate rises sharply, then falls in the successful MC runs but persists in the run that fails to learn. TC commands intermediate subgoals frequently early in training on both tasks, and less often as the policy improves.

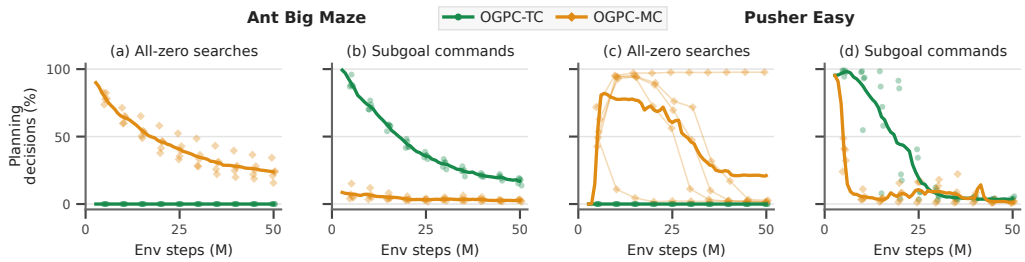


Figure 29: **Planner decisions on Ant Big Maze and Pusher Easy.** Percentage of planning decisions with (a,c) all-zero candidate scores or (b,d) a first subgoal more than one success radius from the task goal. Opaque lines show five-seed means. Markers show individual seed results. Faint lines in (c) connect each MC seed's markers.

F.6 Proposal distribution: goal-conditioned versus unconditional GHM proposals

The planner seeds its first search round with two subgoal chains sampled from the GHM (Algorithm 3). The default configuration draws them with Algorithm 4, the *unconditional* proposal distribution, which chains the GHM with masked action and command so that the chain follows the replay behavior from the current state and ignores the commanded goal. The alternative *goal-conditioned* proposal distribution is defined in Algorithm 5, At every hop, it draws the action from the policy conditioned on the task goal and conditions the GHM on that action, such that the chain follows what the policy would do when heading for $g^\star$. Appendix C.2 discusses the two proposal distributions. Appendix G compares them in Figure 36 and shows rollouts planned with goal-conditioned proposals.

Training-time comparison Figure 30 and Table 22 compare OGPC-TC trained with the goal-conditioned proposal distribution against OGPC-TC trained with the unconditional proposal distribution, with everything else fixed. We report final Success Rate and Time at Goal, averaging the last five evaluations within each seed. Goal-conditioned proposals have negative impact on both metrics for Arm BinPick Easy, with final Time at Goal lower by 93 steps. On Pusher Easy, final Success Rate is lower, while the Time at Goal interval includes zero. We therefore opted to use the unconditional proposal distribution for our default configuration.

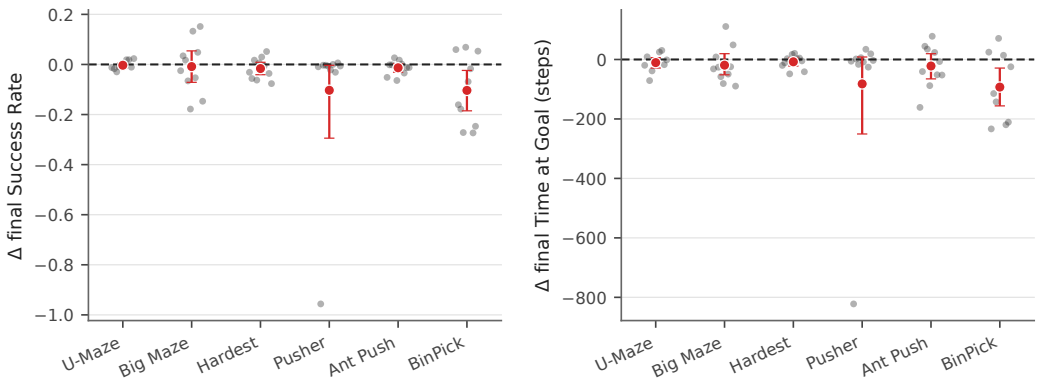


Figure 30: **Proposal-distribution ablation: goal-conditioned minus unconditional GHM proposals.** Paired differences over 10 seeds in final Success Rate (left) and final Time at Goal (right). Colored markers show means with 95% seed-bootstrap intervals. Translucent dots show individual seed-paired differences. Both arms are OGPC-TC and differ only in the proposal distribution used to seed the planner (Algorithms 5 and 4).

Table 22: Proposal-distribution ablation, goal-conditioned (Algorithm 5) minus unconditional (Algorithm 4) proposals: paired differences over 10 seeds with 95% seed-bootstrap intervals. Final values average the last five evaluations within each seed.

Task	Δ final Success Rate	Δ final Time at Goal
Ant U-Maze	$-0.003\,[-0.013, +0.007]$	$-10\,[-29, +6]$
Ant Big Maze	$-0.008\,[-0.071, +0.054]$	$-19\,[-52, +20]$
Ant Hardest Maze	$-0.016\,[-0.041, +0.009]$	$-8\,[-22, +5]$
Pusher Easy	$-0.103\,[-0.294, -0.003]$	$-82\,[-251, +8]$
Ant Push	$-0.013\,[-0.031, +0.003]$	$-22\,[-65, +19]$
Arm BinPick Easy	$-0.104\,[-0.185, -0.024]$	$-93\,[-156, -29]$

F.7 Alternative subgoal selection during collection

We compare learned subgoal policies and replay-graph search as alternatives for guiding collection. Each subgoal policy, referred to here as a *manager*, selects goals for the shared goal-conditioned CRL policy to pursue. All the baselines use the same depth-8 CRL policy and default CRL learning objective, with 50M training interactions and 1,000-step episodes. To keep the evaluation protocol consistent, we do not make use of the manager or graph at evaluation time. The evaluated policy acts directly toward the task goal.

Learned subgoal managers Our implementation of HRAC-CRL retains HRAC's ten-action relative-subgoal manager, twin critics, off-policy goal correction, and adjacency penalty (Zhang et al., 2020). HIGL-CRL adds coverage- and novelty-based landmarks that guide manager training (Kim et al., 2021). We adapt the authors' released HRAC and HIGL implementations. Table 23 summarizes the adaptations. We report one configuration per method, including episode-terminal manager replay, valid-transition sampling, and adjacency training across the commitment window. Our SAC-manager baseline uses Soft Actor-Critic (SAC) to train the subgoal policy (Haarnoja et al., 2018). It selects a new subgoal every 20 environment steps.

Figure 31 shows that neither adaptation improves the policy on Ant U-Maze. Over three seeds, HRAC-CRL and HIGL-CRL attain 45.4% and 46.1% final Success Rate respectively, while CRL achieves 90.6%. Final Time at Goal is 164 and 211 steps, versus 737 for CRL. Simpler goal-space SAC managers, with and without hindsight action relabeling, attain 84.5% and 83.4% Success Rate over ten seeds respectively, but only 150 and 154 steps at the goal compared to 732 for CRL. Reaching the goal therefore does not translate into sustained occupancy for these policies.

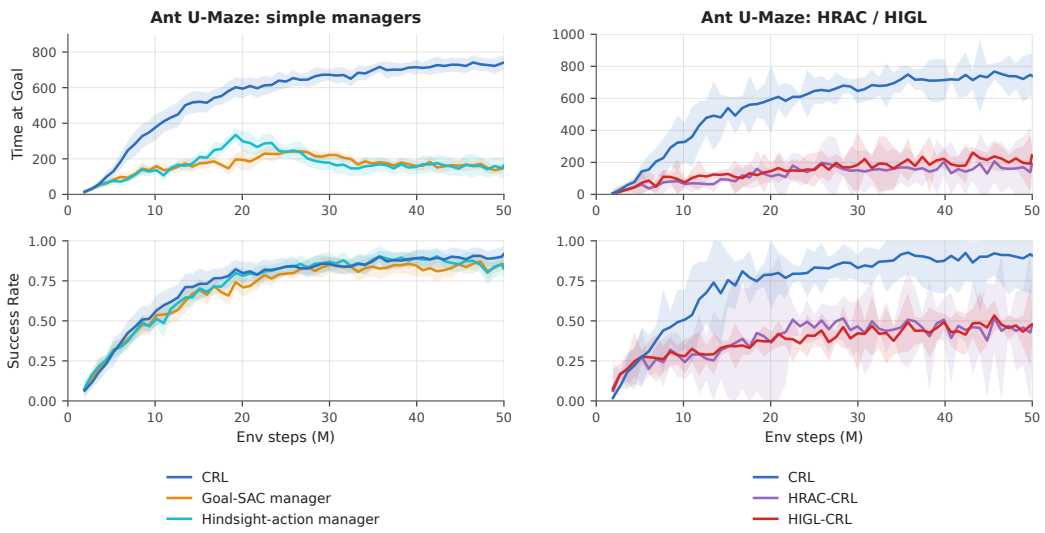


Figure 31: **Learning with subgoal managers on Ant U-Maze.** Top: Time at Goal (steps per 1,000-step episode); bottom: Success Rate. Left: a goal-space SAC manager selects a subgoal every 20 actions. The hindsight-action variant relabels the manager's replay action to the achieved endpoint. Right: HRAC-CRL and HIGL-CRL. All evaluations use the policy alone. Lines and shaded bands are means and pointwise 95% seed-t intervals, without smoothing. Ten seeds per method on the left panels and three on the right panels. Each panel's blue CRL curve uses the same seeds. Final values in the text average the last five evaluations per seed.

Table 23: **Adapting HRAC and HIGL to CRL-guided collection.** Principal differences from the released implementations; the comparison preserves CRL learning and evaluates the policy without its manager.

Component	Released implementations	CRL adaptations
Goal-conditioned policy	TD3 (Fujimoto et al., 2018) with intrinsic subgoal rewards	Depth-8 stochastic CRL without intrinsic-reward loss
Subgoal execution	Relative displacement, updated as the agent moves	Bounded absolute goal passed to CRL
Goal correction	Recorded-action agreement under the controller	Same criterion, using deterministic CRL actions
HIGL landmark search	Controller-value edges and soft value iteration	Adjacency-embedding edges and shortest paths for manager training
Collection and updates	Serial interaction with episode-based updates	1,024 environments, blockwise auxiliary updates scaled by environment count
Evaluation	Manager and goal-conditioned policy act together	CRL policy acts directly toward the task goal

Direct replay-landmark graph collection To test whether graph-based subgoal search over replay could provide similar advantage as planning with learned GHM, we evaluate a SoRB-inspired (Eysenbach et al., 2019) replay-graph collection baseline. This collector selects subgoals directly from a directed graph of 128 replay landmarks. We train a separate categorical distance model which scores connections under the current CRL policy. The graph is refreshed at every block, with 20-action waypoint commitments and direct-goal fallback. We leave the underlying CRL training objective and model architecture unchanged.

We find that neither metric demonstrates a clear final gain over CRL (Figure 32). After activation, graph guidance accounts for 20.2% of transitions on U-Maze, but only 1.3% on Big Maze and 0.8% on Hardest Maze. Therefore, on larger maze environments, the search mostly defaults to direct-goal fallback and fails to provide sustained graph-based guidance.

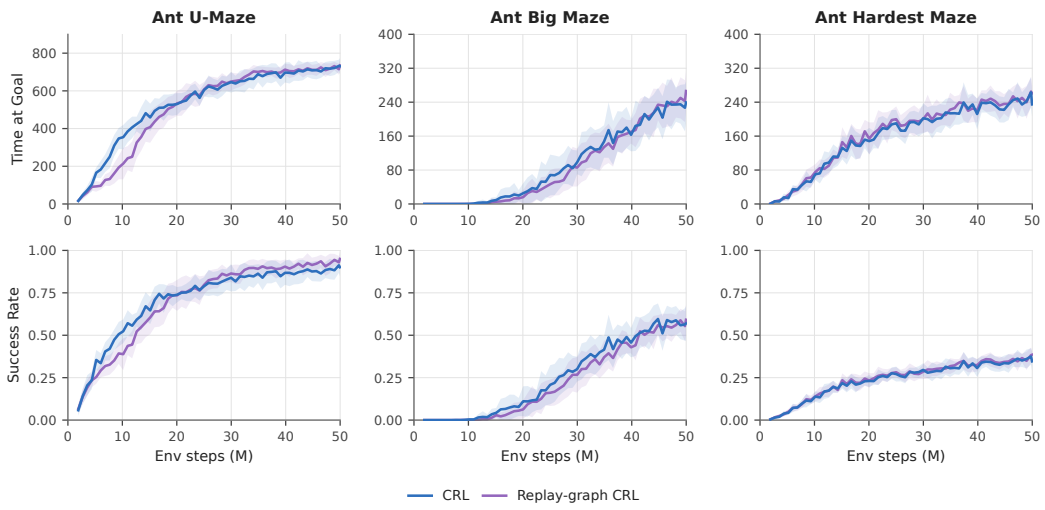


Figure 32: **Learning with replay-graph collection across the Ant mazes.** Time at Goal and Success Rate, evaluated with graph search removed. Lines and bands are means and pointwise 95% seed-t intervals across ten seeds (0–9).

F.8 GC-TD-MPC2: POLICY PRIORS AND SEARCH BUDGET

F.8.1 POLICY PRIORS AND SEARCH BUDGET DURING LEARNING

We test whether a policy prior and a larger candidate population improve the policy learned with GC-TD-MPC2. We train two additional configurations: *12 candidates (1 prior)* and *128 candidates (6 priors)*. Figure 33 shows their learning curves on all six tasks. Neither configuration closes the gap to planner-free CRL in final Time at Goal. The prior candidates follow the shared CRL policy through the latent model, as described in Appendix E.6.2. We train each configuration to 50M environment steps with five seeds and evaluate without planning. Comparisons use the same five seeds of CRL and default GC-TD-MPC2.

Final performance Figure 34 compares the mean of the last five evaluations, using paired seed-t 95% intervals. Both configurations perform below CRL on every task. Neither shows a statistically significant improvement over default GC-TD-MPC2. The clearest deterioration occurs on Pusher Easy, where both lose approximately 380 steps of Time at Goal. The smaller configuration also performs worse on Ant Hardest Maze. The larger configuration improves over the smaller counterpart by 49 steps (95% interval $[23, 75]$), but not over the default configuration.

The smaller configuration has higher final Success Rate on Ant Push, and both have higher final Success Rate on Arm BinPick Easy and higher training-average Time at Goal on Ant Push. However, these improvements do not close the gap to CRL.

F.8.2 SEARCH BUDGET ON FROZEN MODELS

We separately evaluate whether a larger search budget improves performance after the dynamics model and policy have been learned. This isolates changes to planning at evaluation time from changes to the training data. Figure 35 shows that wider search generally reduces Time at Goal on these models, whereas a policy prior helps on some tasks.

We use the default GC-TD-MPC2 checkpoints at 50M environment steps, with five seeds per task. The planner acts at every evaluation step with 12, 48, 128, or 512 candidates per round. All configurations use four rounds. The first three retain three elites, while the 512-candidate configuration uses ZILOT's six. We also test one prior candidate at populations of 12 and 128, and six prior candidates at 128. Each seed is evaluated on two repeats of 128 episodes with shared reset states across

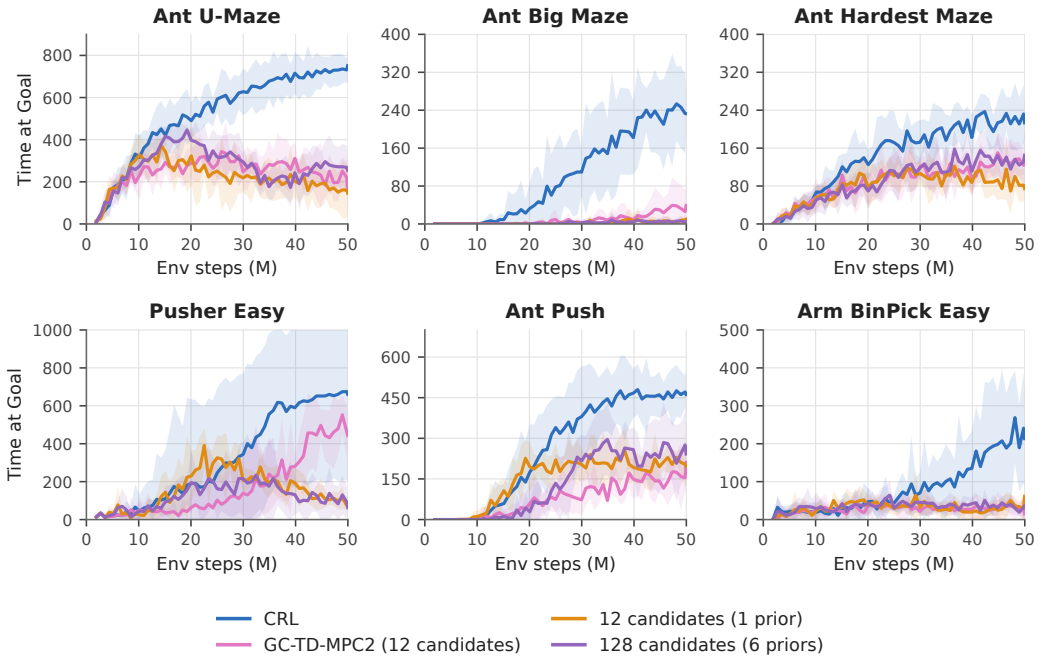


Figure 33: **Policy priors and a larger search budget during learning.** Time at Goal of the learned policies, evaluated without planning. Curves show unsmoothed means over five seeds, with pointwise 95% seed-t intervals clipped to the metric's range. All search configurations use four rounds and three elites.

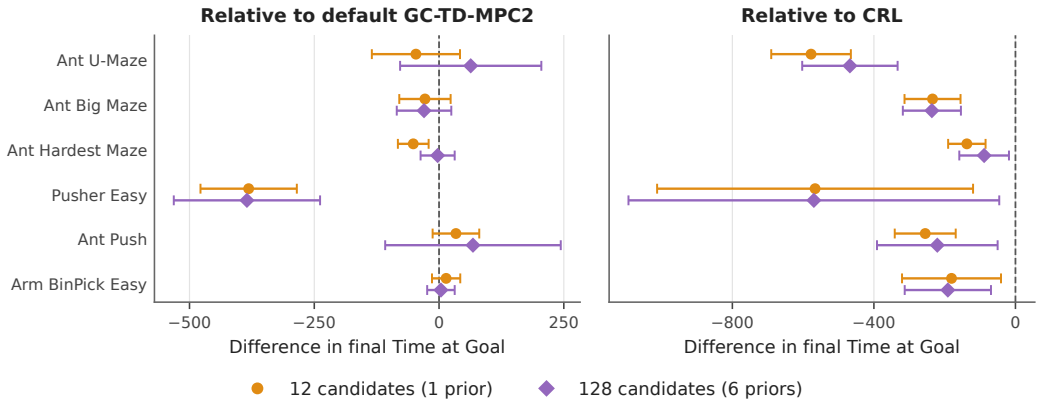


Figure 34: **Paired changes in final Time at Goal.** Each point compares one prior configuration against default GC-TD-MPC2 (left) or CRL (right). Positive values favor the prior configuration. Error bars are 95% seed-t intervals over five matched seed differences, without adjustment for multiple comparisons. Final performance averages the last five planner-free evaluations.

configurations. The planner-free reference is the policy from the *same checkpoint*, not the separately trained CRL baseline in Figure 33.

Without a policy prior, increasing the population from 12 to 48 or 128 reduces Time at Goal on Ant U-Maze, Ant Hardest Maze, Pusher Easy, and Ant Push. The same trend holds for the 512-candidate configuration on these tasks. Ant Big Maze remains near zero, while 48 candidates improve on 12 on Arm BinPick Easy. We find that adding one prior candidate at population 12 improves Ant Hardest

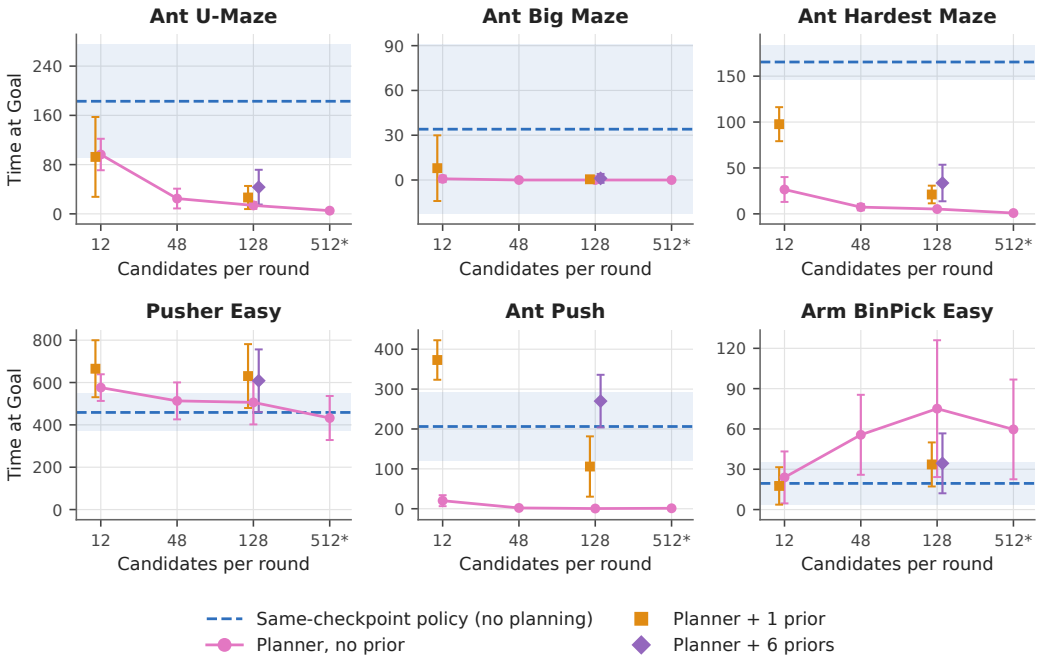


Figure 35: **Search-budget sensitivity with frozen models.** Time at Goal when planning from the same default GC-TD-MPC2 checkpoints. Markers show five-seed means with 95% seed-t intervals. The dashed line and shaded band show the same checkpoints' policies evaluated without planning. Prior markers are slightly offset horizontally for improved visibility. *The 512-candidate configuration uses six elites rather than three.

Maze and Ant Push. At population 128, six prior candidates improve over one on Ant U-Maze and Ant Push. We select six priors for the larger online configuration using this comparison.

F.8.3 Computational cost

Table 24 reports the realized wall-clock time per run of both prior configurations. Seeds 0–1 ran on an RTX 3090 and seeds 2–4 on an RTX 5090, with each seed pair on the same GPU model. Before training, we measured step times on Ant Big Maze and Pusher Easy using an RTX 5090. The larger configuration approximately matches OGPC-TC's training cost on those two tasks, taking 0.91–0.96× as long as the OGPC-TC runs on the same GPU. The 60 completed runs consumed 836 GPU-hours in total. Table 18 reports the costs of default GC-TD-MPC2 and the other methods.

Table 24: **Wall-clock time per run of the two GC-TD-MPC2 prior configurations.** Mean elapsed hours to 50M transitions over the listed seeds on each GPU type.

	12 candidates (1 prior)		128 candidates (6 priors)	
Environment	RTX 3090 (2 seeds)	RTX 5090 (3 seeds)	RTX 3090 (2 seeds)	RTX 5090 (3 seeds)
Ant U-Maze	11.1	4.6	31.1	10.6
Ant Big Maze	11.2	4.7	31.3	10.7
Ant Hardest Maze	11.6	4.8	31.6	10.9
Pusher Easy	11.5	4.8	31.6	10.9
Ant Push	11.4	4.7	31.4	10.8
Arm BinPick Easy	18.8	7.1	38.9	13.2

G Qualitative Planner Rollouts: Subgoal Plans and GHM Proposals

Figures 37–42 visualize the selected plans, GHM proposals, and executed motion during one successful episode in each of six environments. We use seed-0 50M OGPC-TC checkpoints for the Ant tasks and 100M checkpoints for the Humanoid tasks. These rollouts use goal-conditioned proposals (Algorithm 5), rather than the unconditional proposal distribution used during training (Algorithm 4). Each figure pairs six snapshots with synchronized MuJoCo views. Maps show the plan active at each snapshot, and the bottom row shows four-hop GHM proposals from a selected replanning step.

G.1 Ant U-Maze

Figure 36 compares the two proposal distributions from the same state and GHM. In this example, unconditional proposals spread more broadly, while goal-conditioned proposals follow the corridor toward the task goal.

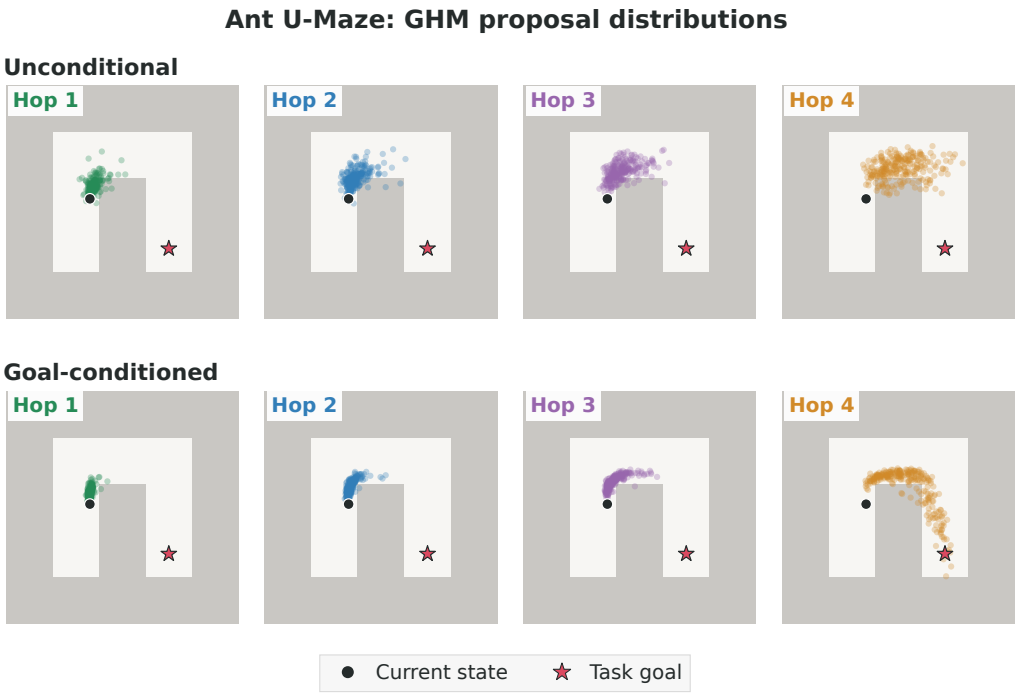


Figure 36: **Unconditional and goal-conditioned GHM proposals on Ant U-Maze.** Both rows use the same GHM and recorded state, shown at step 235 in Figure 37(d), with 256 four-hop chains per row. Columns show the sampled subgoal coordinates at successive hops. The unconditional proposals mask the GHM's action and goal inputs (Algorithm 4). The goal-conditioned proposals use the task goal and an action sampled from the goal-conditioned policy at each hop (Algorithm 5).

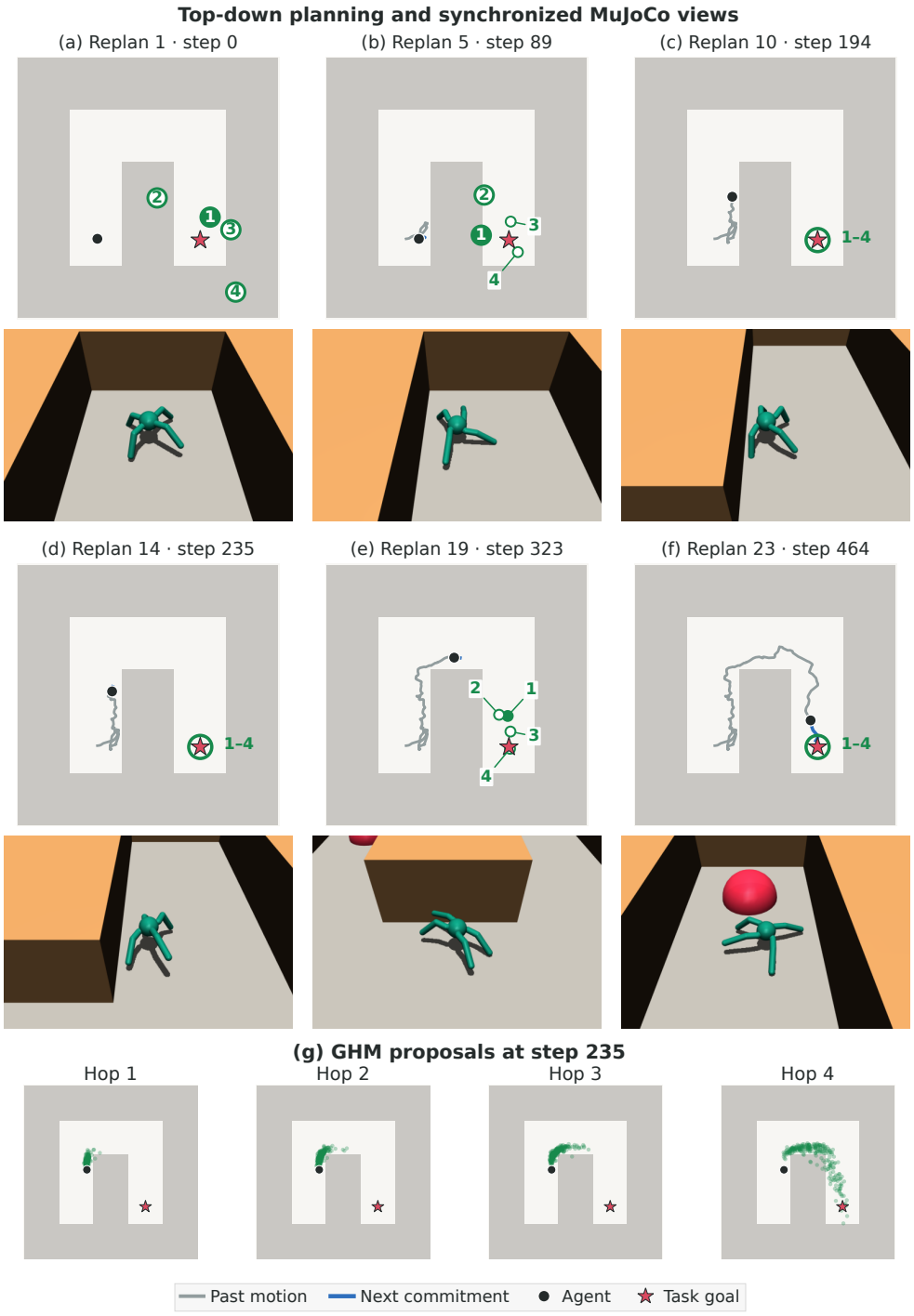


Figure 37: **Planning and physical execution on Ant U-Maze.** Six replanning boundaries are shown from left to right, then top to bottom. Each map and the camera view below it show the same recorded state. Numbers identify the selected subgoals, with the first subgoal filled. Only the first subgoal is pursued before replanning. The subgoals that coincide with the commanded goal (red star) are represented as an enclosing circle around it. The bottom row shows goal-conditioned GHM proposals from the state in (d).

G.2 Ant Big Maze

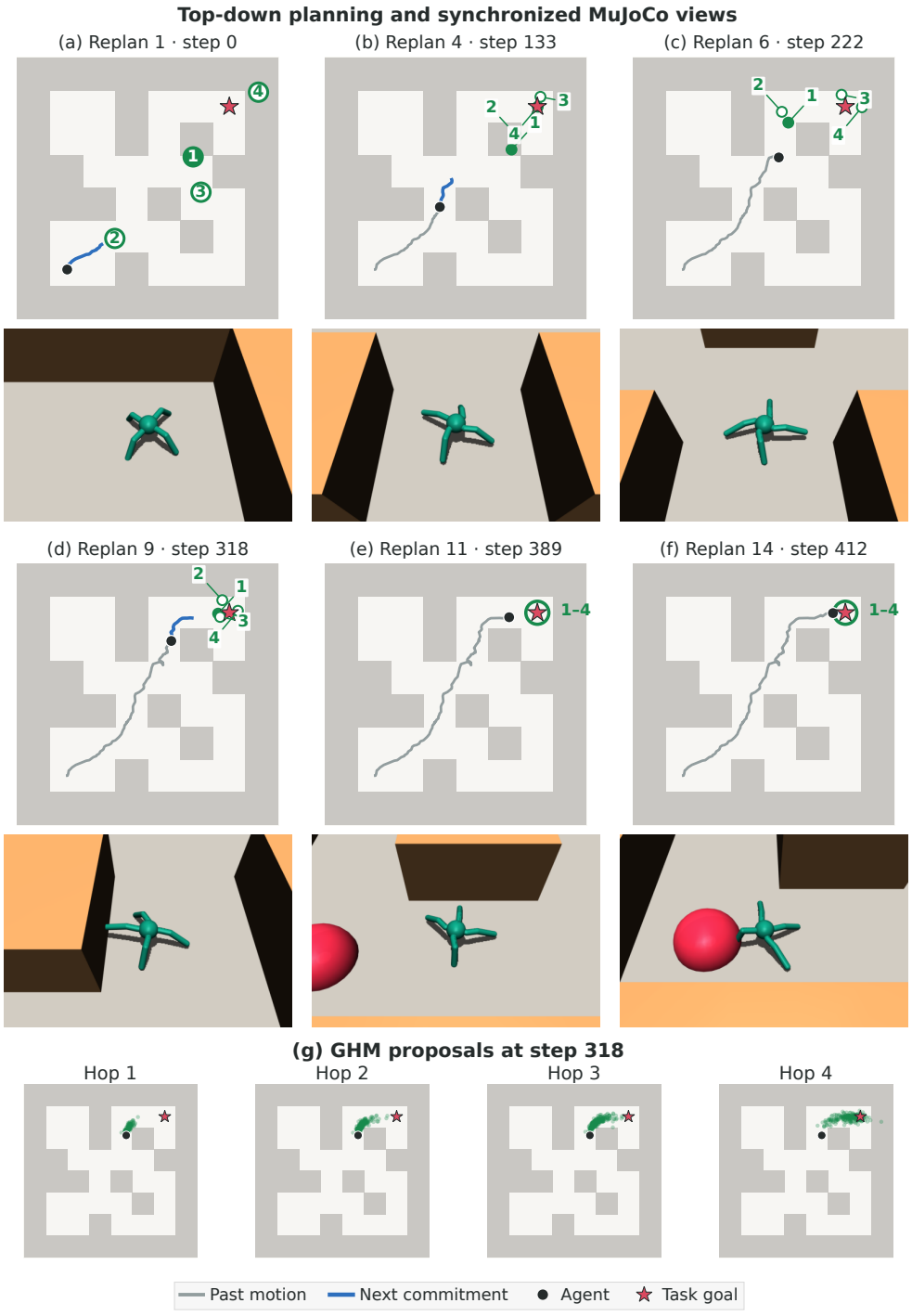


Figure 38: **Planning and physical execution on Ant Big Maze.** The task goal is in the upper-right corner, ten grid edges from the starting cell. Six replanning boundaries are shown using the same layout and notation as Figure 37. The bottom row shows goal-conditioned GHM proposals from the state in (d).

G.3 Ant Hardest Maze

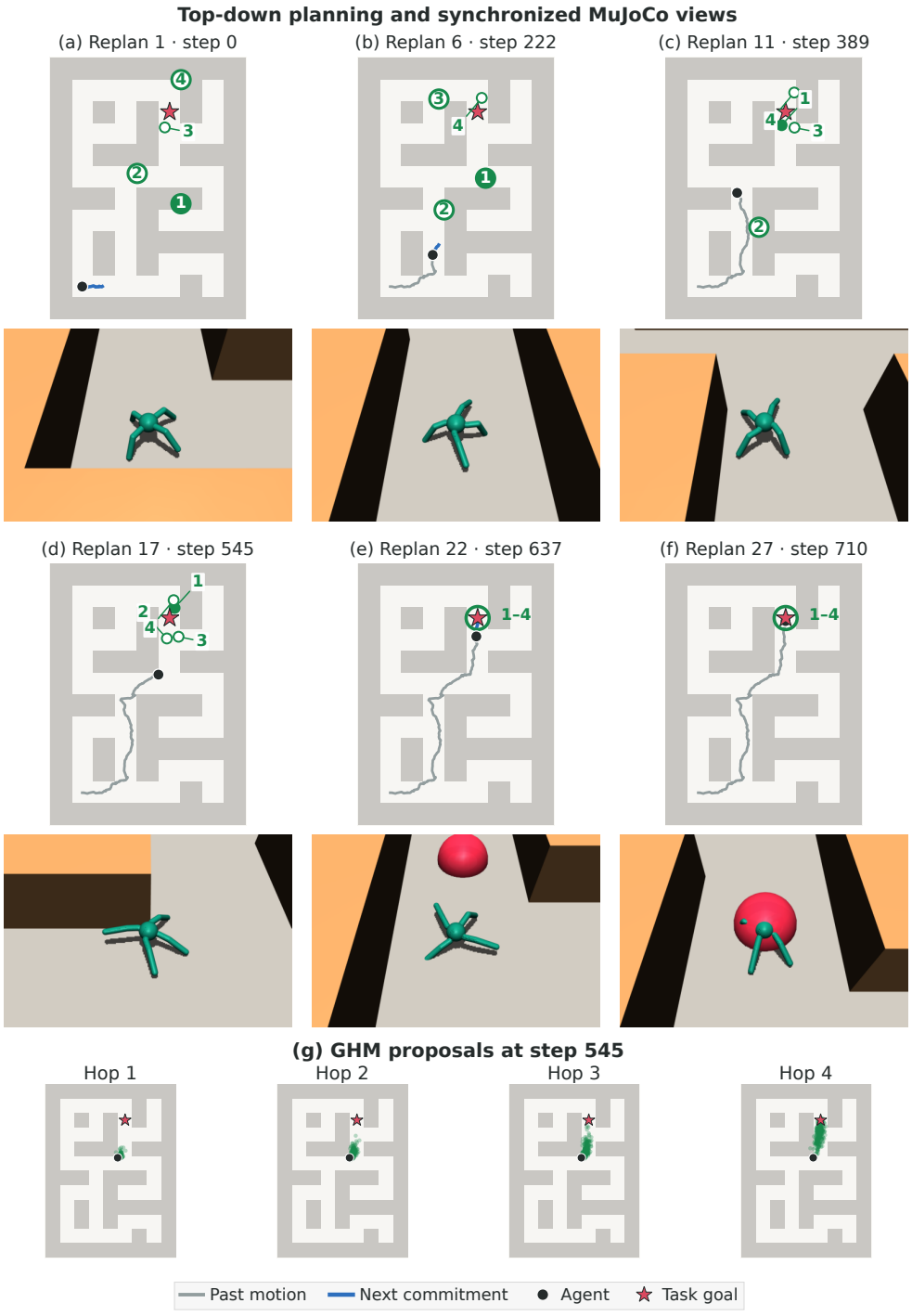


Figure 39: **Planning and physical execution on Ant Hardest Maze.** The task goal lies in the upper part of the maze, twelve grid edges from the starting cell. Six replanning boundaries are shown using the same layout and notation as Figure 37. The bottom row shows goal-conditioned GHM proposals from the state in (d).

G.4 ANT PUSH

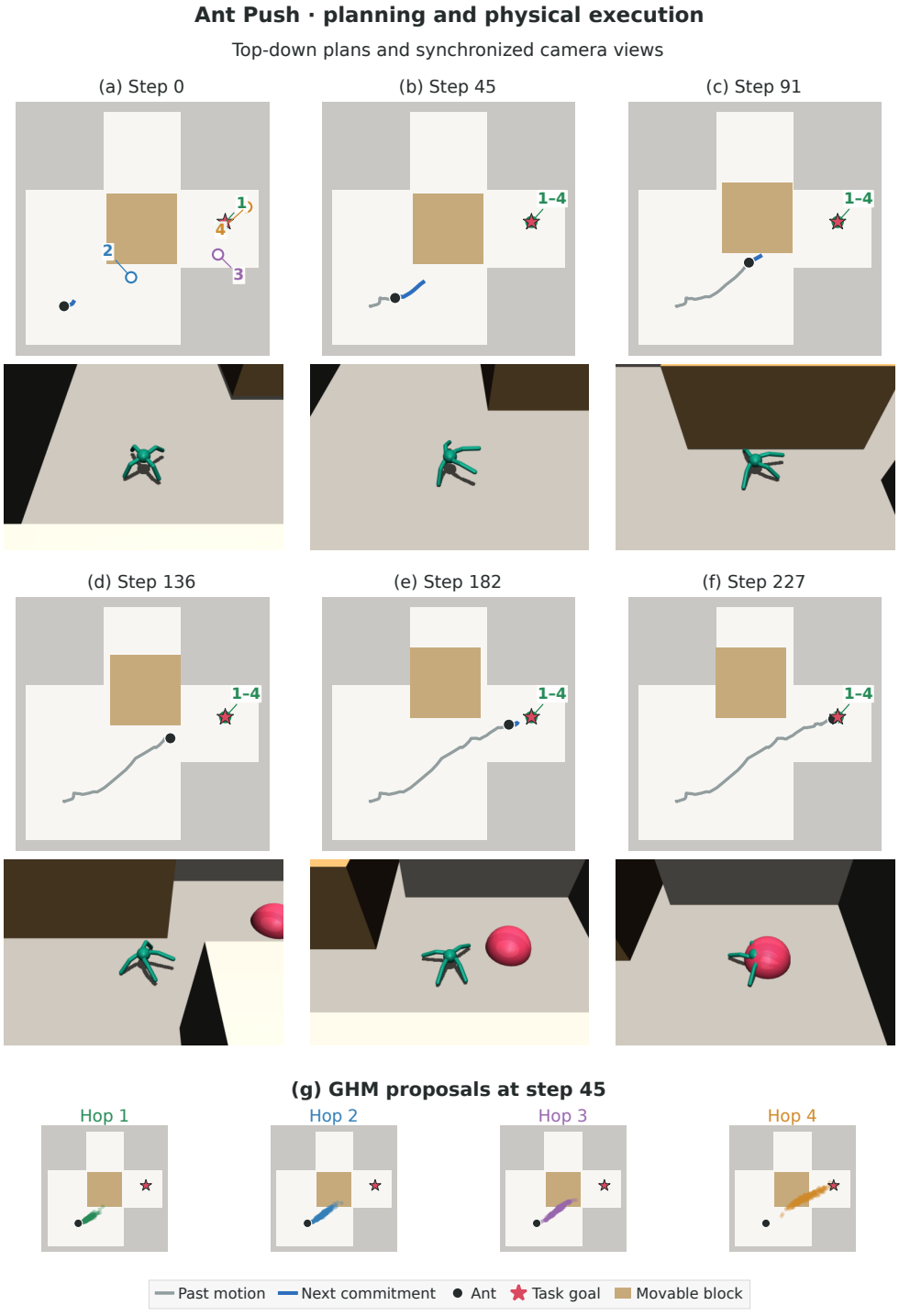


Figure 40: **Planning and physical execution on Ant Push.** Six snapshots show the Ant moving the block before reaching the task goal. The brown square marks the movable block, while the task goal specifies the Ant's position. The bottom row shows goal-conditioned GHM proposals from the state in (b), at step 45.

G.5 HUMANOID

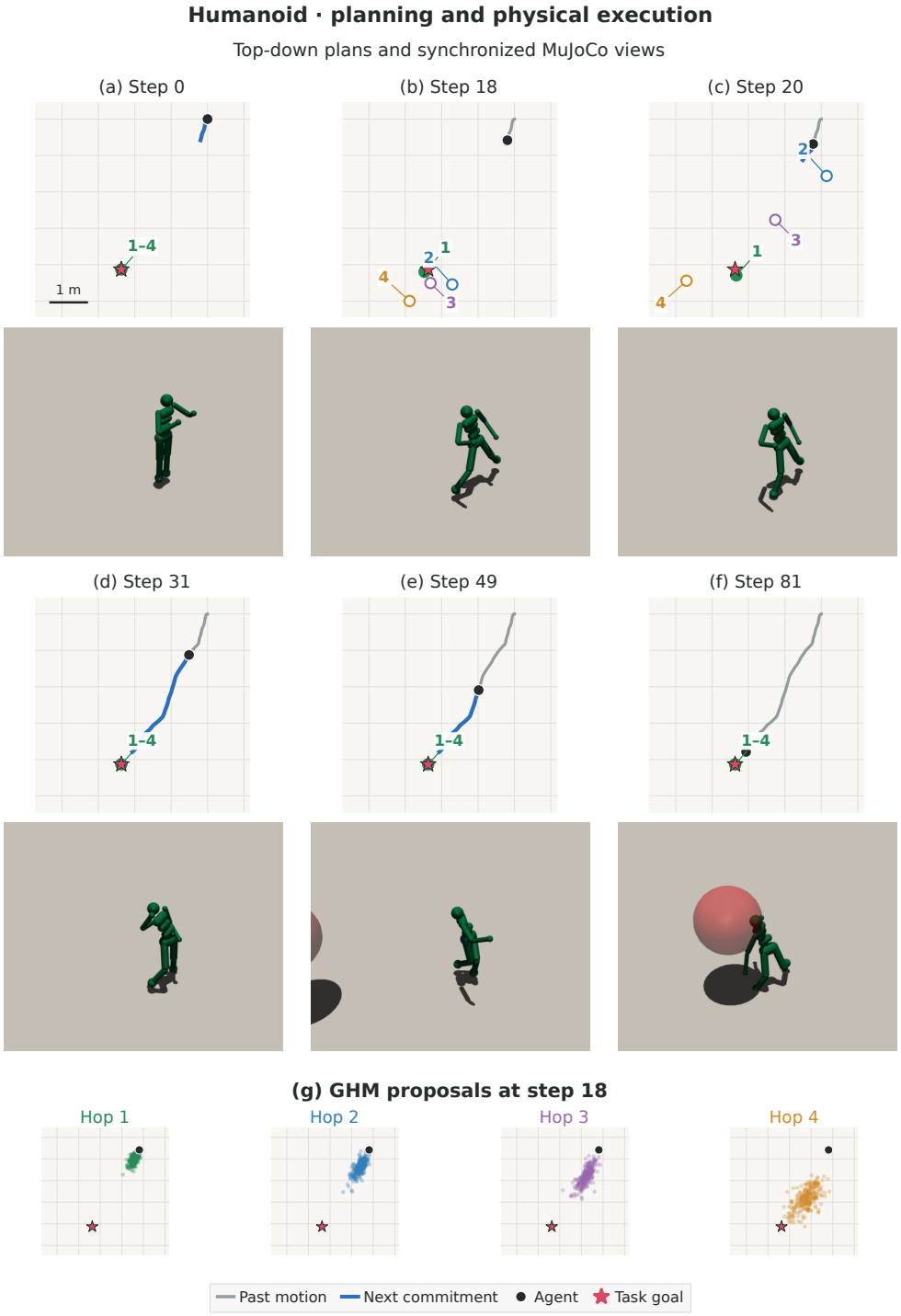


Figure 41: **Planning and physical execution on Humanoid.** Six snapshots from the seed-4 100M checkpoint show the selected plans and the Humanoid approaching its goal. The maps project positions onto the horizontal plane. The bottom row shows goal-conditioned GHM proposals from the state in (b), at step 18.

G.6 HUMANOID U-MAZE

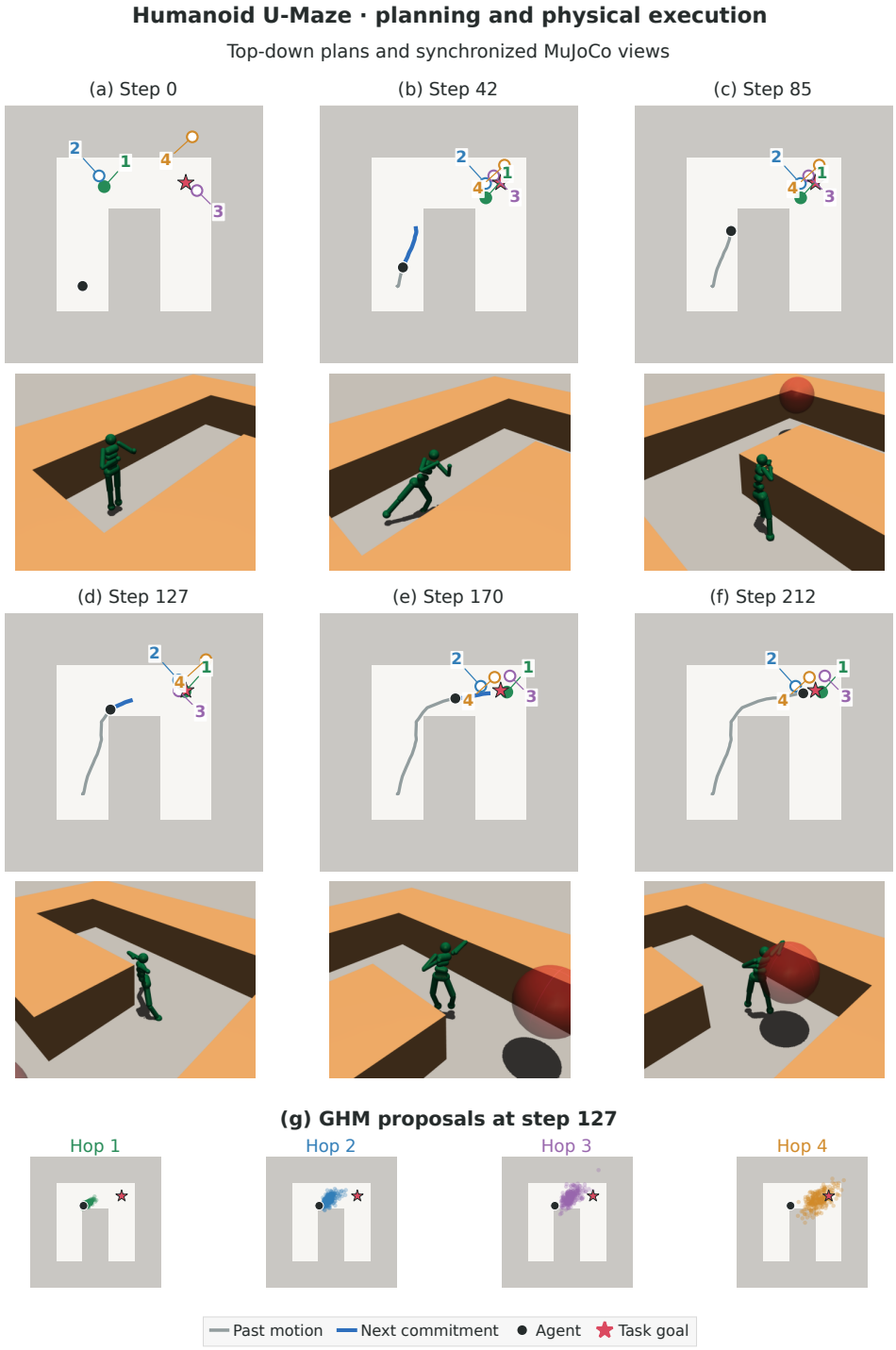


Figure 42: **Planning and physical execution on Humanoid U-Maze.** Six snapshots from the seed-2 100M checkpoint show the Humanoid navigating the corridor to the task goal. The maps use the same horizontal projection as Figure 41. The bottom row shows goal-conditioned GHM proposals from the state in (d), at step 127.