IMAGINED ACTION SEQUENCE GUIDANCE FOR CONSTRAINED POLICY OPTIMIZATION IN GOAL-CONDITIONED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Goal-conditioned reinforcement learning trains a policy to guide an agent toward a desired goal.Sparse rewards make useful and blocked actions difficult to distinguish near obstacles.Continuation Guided Constrained Policy Optimization (CGCPO) is proposed to combine imagined sequence guidance with a separate comparison of current action values.Candidate and reference actions condition a generator learned from recorded trajectories, and a sequence critic estimates the returns of the generated action blocks.These blocks are evaluated without intermediate feedback, whereas the deployed policy selects a new action after every observation.An independently trained primitive critic therefore compares the actual candidate and reference actions followed by the same reference policy.The policy objective favors a larger sequence value gap and penalizes primitive value gaps below a chosen margin.The trained policy selects one action per observation during execution.Under fixed policies and a fixed generator, the associated Bellman evaluations converge to their unique fixed points. The candidate policy's optimality gap is bounded by the sequence and primitive estimation errors, the generated block discrepancy, the candidate advantage spread and the remaining improvement in the policy score.Experiments on Fetch and PointMaze show the highest late success among the compared methods on the larger maze layouts and gains over baselines on obstacle manipulation.

1 INTRODUCTION

Goal-conditioned reinforcement learning trains a policy to move an agent or object toward a desired goal (Schaul et al., 2015; Andrychowicz et al., 2017). These policies support robot manipulation and navigation across different object locations and destinations (Park et al., 2025a). The policy selects each action from the current state and desired goal and learns from the resulting experience. In the tasks considered here, useful reward feedback arrives only when the desired goal is reached. These sparse rewards reduce the need to design rewards for intermediate progress (Andrychowicz et al., 2017). To choose useful actions before success, the policy relies on estimates of discounted return after each action (Lillicrap et al., 2016). Bellman updates learn these action values by combining immediate rewards with estimated returns at later states. The quality of these later estimates therefore matters when immediate rewards give little help to the policy that must choose among them.

This difficulty is clear near obstacles, where several actions may be needed before useful and blocked choices receive different feedback. Lifting can prepare an object to pass above a barrier whereas direct pushing can leave it blocked. Both choices can receive the same unsuccessful reward despite their different later outcomes. Maze navigation can likewise require an initial detour that increases the distance from the desired goal. To distinguish such actions, Bellman updates must rely on their estimated later returns. Errors in these later estimates can propagate to earlier action values through repeated updates and weaken the preference for useful choices (Fujimoto et al., 2018; Zheng et al., 2026). Recorded trajectories retain the later outcomes of these actions, so the same experience can compare these choices over several steps.

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.

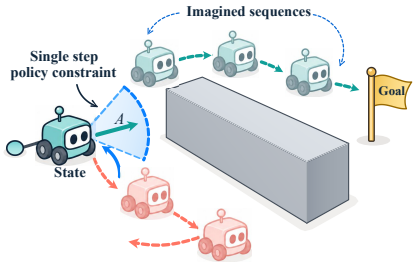


Figure 1: Imagined sequences distinguish a route that passes the barrier toward the goal from a route that turns back, and the single step constraint applies to the action at one state and one desired goal at a time.

Existing methods use this experience through hindsight goal relabeling, multistep value learning and action sequence evaluation. Hindsight experience replay (HER) reuses goals achieved in recorded trajectories as desired goals and recomputes rewards (Andrychowicz et al., 2017). This supports learning even when the original desired goal was missed. Multistep value learning updates goal-conditioned values using future states across several recorded transitions (Zheng et al., 2026). It passes information from later outcomes to earlier decisions over longer intervals. Action sequence evaluation compares predicted returns for sequences of consecutive actions (Seo & Abbeel, 2025; Li et al., 2025). Generative policies learn these sequences from recorded experience so that they can be compared before all their actions are executed (Chi et al., 2023; Li et al., 2026). However, a generated block is scored as an open loop, while the deployed policy chooses a new action after every observation. A preference between sequences can then reflect their different continuations rather than the actions that the deployed policy would take at the same state.

In this paper, we propose Continuation Guided Constrained Policy Optimization to combine imagined sequence guidance with a separate comparison of current action values. This method is called CGCPO. Figure 1 shows two such sequences at a barrier. Candidate and reference actions from the two policies condition a generator trained on recorded trajectories. A sequence critic estimates the return of each sequence, and the difference between mean candidate and reference scores guides the policy toward conditions with higher predicted returns. The later actions of a sequence form the continuation of its action condition, and the sequence critic scores that continuation in open loop while the deployed policy acts again after every observation. An independently trained primitive critic therefore compares the actual candidate and reference actions followed by the same reference policy. A separate Bellman target ties this comparison to a shared rule for later actions, and the update applies a finite penalty when the primitive value gap falls below a chosen margin. The trained policy selects one action from each new observation and desired goal. Our contributions are as follows. We identify an open-loop mismatch, because a generated block is scored as if it were executed without new observations. We propose CGCPO, which sets the update direction from sequence values and penalizes current actions that fall below the reference value under the same continuation. We bound the optimality gap of a fixed candidate policy by sequence, primitive, advantage spread and selection terms. On Fetch and PointMaze the method stays within the top three on all ten tasks, and the ablations show sequence gains on obstacle manipulation that four-step targets miss when the object must pass the barrier.

2 Related Work

Goal-conditioned learning improves values and policies through hindsight relabeling and temporal or quasimetric relationships between states and goals (Andrychowicz et al., 2017; Myers et al., 2024; 2025; Zheng et al., 2026). Action sequence methods learn values or policies over consecutive actions (Seo & Abbeel, 2025; Li et al., 2025). They therefore score a whole block of actions rather than a single action at the current state. Diffusion and flow models provide action or sequence proposals (Chi et al., 2023; Ren et al., 2025; Park et al., 2025b; Zhang et al., 2025). DQC distills optimistic full sequence values into a shorter sequence critic (Li et al., 2026). Chunk-Guided Q-Learning trains a single action critic with a primitive Bellman target and regularization toward sequence values (Song et al., 2026). Both methods compare sequences against primitive values inside one critic and they learn offline from fixed data, whereas CGCPO keeps two value rules in separate critics that train together on HER relabeled replay. Reference comparisons and regularization also guide policy updates in conservative policy iteration, trust region methods and offline learning (Kakade & Langford, 2002; Schulman et al., 2015; Chen et al., 2024; Fang et al., 2025; Ke et al., 2025). These methods adjust one update rule against a reference, whereas CGCPO compares two value rules that share the same continuation. The multiplier update of Appendix A.1 is projected dual ascent on the constraint in Equation (4) with a capped multiplier, as in Lagrangian methods for

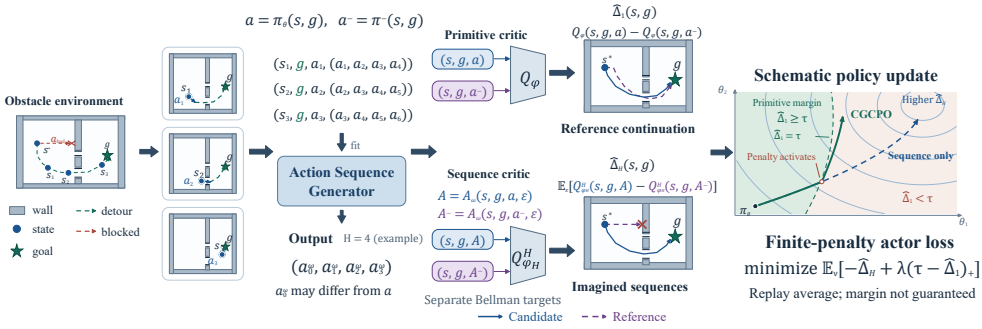


Figure 2: Imagined sequence values guide the update and margin shortfalls are penalized.

constrained reinforcement learning (Tessler et al., 2019; Stooke et al., 2020). Unlike constrained policy optimization (Achiam et al., 2017), the constraint here compares two value rules at the same state rather than bounding an auxiliary cost along a trajectory.

3 Background

Goal-conditioned tasks are modeled as Markov decision processes with state $s \in \mathcal{S}$, action $a \in \mathcal{A}$ and desired goal $g \in \mathcal{G}$. The desired goal $g \sim p_g$ stays fixed throughout an episode while deterministic policy $a_t = \pi_\theta(s_t, g)$ selects each action. The transition $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ gives the next state and achieved goal $\psi(s_{t+1})$ such as the resulting object position. The reward $r_t = -\mathbf{1}[\|\psi(s_{t+1}) - g\| > \epsilon_g]$ is zero within success threshold ϵ_g and minus one otherwise. The policy value $V^\pi(s, g) = \mathbb{E}_\pi[\sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, g]$ measures expected discounted return with discount $\gamma \in (0, 1)$. True termination under the evaluated goal excludes later rewards and values in tasks ending upon arrival. Time limit truncation ends an episode without true termination, so the estimated value beyond the final observed transition is retained. Policy learning seeks to maximize $J(\pi) = \mathbb{E}_{(s_0, g) \sim \mu} V^\pi(s_0, g)$ over the joint starting state and desired goal distribution μ with goal marginal p_g.

4 Method

4.1 Framework

Figure 2 shows how recorded trajectories train a sequence generator and two critics to guide the policy update. Trajectories $\zeta = (s_0, a_0, r_0, d_0, \ldots, s_T)$ collected by behavior policy π_b are stored in replay buffer $\mathcal{D}$ where d_t marks true termination. Complete recorded blocks of H actions train the generator and sequence critic while their first transitions train the separate primitive critic. The current policy π_θ is called the actor and proposes candidate action $a = \pi_\theta(s, g)$. The reference policy π^- slowly tracks the actor parameters through an exponential moving average (EMA) and supplies reference action $a^- = \pi^-(s, g)$. The generator $A_\omega(s, g, a, \varepsilon)$ uses each action condition with state s, desired goal g and noise ε to produce imagined sequences without a learned dynamics model. The later actions of a sequence form the continuation of the conditioning action. The sequence critic $Q^H_{\phi_H}(s, g, A)$ scores the sequences generated under these two action conditions. The primitive critic $Q_\phi(s, g, a)$ is trained separately to estimate the return of a current action followed by the reference policy at every later observation, so the two critics judge one state and desired goal. Section 4.3 compares the two critics through these action conditions and turns that comparison into the policy update.

4.2 Imagined Action Sequences and Value Learning

Complete recorded blocks $A_t = (a_t, \ldots, a_{t+H-1})$ supply generator targets and conditioning inputs (s_t, g, a_t) from replay. HER reuses recorded achieved goals as desired goals and recomputes rewards. The generator is trained by flow matching to turn random noise into sequences resembling recorded blocks (Lipman et al., 2023), and Appendix A.1 gives the flow construction together with its loss.

Flow matching is one implementation of this generator, and the core idea of CGCPO does not depend on this choice. Other suitable sequence models or a multilayer perceptron can implement the same interface. The action condition a guides generation but need not equal the first generated action a_0^ω, so the scored block need not start with the conditioning action of the generator. The sequence critic fits $Q^H_{\phi_H}(s_t, g, A_t)$ to the target in Equation (1), which combines recorded block rewards with estimated return after the block.

$$y_t^H = R_t^{(H)} + \gamma^H m_{t,H} \bar{Q}^H\big(s_{t+H}, g, A_\omega(s_{t+H}, g, \pi^-(s_{t+H}, g), \varepsilon')\big) \tag{1}$$

where $R_t^{(H)} = \sum_{h=0}^{H-1} \gamma^h m_{t,h} r_{t+h}$ sums the discounted block rewards. The mask $m_{t,h} = \prod_{j=0}^{h-1}(1 - d_{t+j})$ excludes rewards after stored termination with initial mask $m_{t,0} = 1$. The target sequence critic $\bar{Q}^H$ scores a new sequence generated at the block boundary from the reference action and fresh noise ε'. The primitive critic is trained to estimate return from one action followed by the reference policy after each new observation and under the original task reward. The primitive prediction $Q_\phi(s_t, g, a_t)$ is fitted from the first transition of the same complete block using Equation (2).

$$y_t = r_t + \gamma(1 - d_t)\bar{Q}\big(s_{t+1}, g, \pi^-(s_{t+1}, g)\big) \tag{2}$$

where r_t is the task reward at the first transition of the block, d_t marks termination at that transition and $\bar{Q}$ is the target primitive critic. The target discounts one step by γ and hands the later part of the return to the reference policy, while the sequence target in Equation (1) discounts a whole block by γ^H. Both targets therefore share the same reference continuation and differ in the length of the recorded part. Each target critic slowly tracks its corresponding learned critic and stays fixed during target computation.

4.3 Constrained Policy Optimization

The two critics score the same state and desired goal under their own rules, and this section compares them through two gaps. For candidate action $a = \pi_\theta(s, g)$ and reference action $a^- = \pi^-(s, g)$, Equation (3) defines a sequence gap and a primitive gap. The sequence gap averages the sequence critic over generator noise under each action condition, and the primitive gap compares the two actual actions followed by the same reference policy. A positive gap therefore favors the candidate condition for the sequence rule and the candidate action for the primitive rule.

$$\widehat{\Delta}_H(s, g) = \widehat{q}_H(s, g, a) - \widehat{q}_H(s, g, a^-) \qquad \widehat{\Delta}_1(s, g) = Q_\phi(s, g, a) - Q_\phi(s, g, a^-) \tag{3}$$

where $\widehat{q}_H(s, g, a) = \mathbb{E}_\varepsilon Q^H_{\phi_H}(s, g, A_\omega(s, g, a, \varepsilon))$ averages the sequence critic over generator noise under the action condition a. The same sampled noise is used for both generated sequences in each comparison. The two gaps can have opposite signs when generated sequences and the reference continuation favor different current choices at the same state and desired goal. Because the reference policy tracks the actor slowly and the default margin is zero, the constraint asks the candidate action to score at least as well as the action of the previous policy under the primitive critic (Kakade & Langford, 2002). This monotone improvement condition is of the kind used in conservative policy iteration and in the trust region methods that bound the size of a policy step against a reference. The ideal objective in Equation (4) increases the mean sequence gap while requiring a primitive margin at each evaluated state and desired goal.

$$\max_{\pi_\theta} \mathbb{E}_{(s,g)\sim\nu} \widehat{\Delta}_H(s, g) \quad \text{s.t.} \quad \widehat{\Delta}_1(s, g) \geq \tau \quad \forall (s, g) \in \mathcal{E}_{\text{eval}} \tag{4}$$

where ν is the replay distribution over state and desired goal pairs and $\mathcal{E}_{\text{eval}} \subseteq \mathcal{S} \times \mathcal{G}$ contains the evaluated pairs. The margin τ sets the required primitive gap, with $\tau = 0$ requiring the candidate value to match or exceed the reference value at every pair that the evaluation domain contains. The first term raises the mean sequence gap and the second term charges any shortfall below the margin, so the actor receives one signal from each critic. The sampled policy update uses Equation (5) to combine sequence guidance with a finite penalty on primitive margin shortfalls.

$$\mathcal{L}_{\text{actor}}(\theta) = \mathbb{E}_{(s,g)\sim\nu}\Big[-\widehat{\Delta}_H(s, g) + \lambda\big(\tau - \widehat{\Delta}_1(s, g)\big)_+\Big] \tag{5}$$

where $\lambda \geq 0$ controls penalty strength and $(z)_+ = \max\{z, 0\}$ keeps only positive values. The penalty is active only when $\widehat{\Delta}_1 < \tau$, and a finite penalty keeps the sequence term active below the margin, so the update still follows sequence values while it corrects the primitive shortfall. The

generator and both critics remain fixed together with the reference policy during each policy update. Sequence gradients pass through the generator to favor action conditions with higher sequence critic values. Primitive gradients contribute below the margin when $\lambda > 0$ to reduce the predicted shortfall under the reference continuation. Penalty strength increases with accumulated mean shortfalls up to a cap, and Appendix C.4 gives the derivatives of both gradient paths together with the clipping rules that apply at the action limits of the decoder.

4.4 Theoretical Analysis

CGCPO uses generated sequence returns to guide policy updates and penalizes primitive value gaps below a chosen margin. The analysis establishes convergence of the associated Bellman evaluations and identifies their limiting values. A second result bounds the difference between the value of the learned policy and the optimal value. The generator, learned score functions, reference policy π^- and candidate policy π^+ stay fixed during these evaluations. Both results therefore describe fixed rules. The state and desired goal are written as $x = (s, g)$ with g fixed along each trajectory. The optimal state and action values are denoted by V^* and Q^*, and true termination removes all later rewards and values unlike a time limit that truncates an episode.

Assumption 1 (Evaluation domain and regularity)**.** *Rewards satisfy $|r| \le R_{\max}$ with discount $0 < \gamma < 1$, and all Bellman values use the original task reward. The evaluation domain contains every successor under every admissible action and all required generated blocks. The policies π^- and π^+ are deterministic measurable admissible policies, and generated actions are admissible along their paths. The kernels are measurable and all Bellman maps preserve bounded measurability. The optimal map admits a measurable uniformly δ maximizing action selector for every bounded measurable state value function and every $\delta > 0$. An absorbing terminal input with zero value is included when taking suprema and infima over the evaluation domain.*

The generated path law $\mathbb{K}_{\omega,H}(\cdot \mid x, a)$ executes the full block $A_\omega(x, a, \varepsilon)$ in the environment. The next block is generated from the reference action at its boundary. This rule defines the sequence Bellman map below, together with the ordinary action value map for a fixed policy σ.

$$\begin{aligned}(\mathcal{B}^-_{\omega,H} f)(x,a) &= \mathbb{E}_{\mathbb{K}_{\omega,H}(\cdot|x,a)}\left[R_H + \beta m_H f(x_H, \pi^-(x_H))\right], \\ (\mathcal{T}^\sigma q)(x,a) &= \mathbb{E}\left[r_0 + \gamma(1-d_0) q(x_1, \sigma(x_1)) \mid x, a\right].\end{aligned} \tag{6}$$

where $\beta = \gamma^H$ with positive integer H. The masks $m_h = \prod_{j=0}^{h-1}(1 - d_j)$ use true termination and $m_0 = 1$. The block return is $R_H = \sum_{h=0}^{H-1} \gamma^h m_h r_h$. Both maps are zero at terminal starts.

Proposition 1 (Convergence of the two evaluation operators)**.** *Under the conditions above, $\mathcal{B}^-_{\omega,H}$ preserves order and contracts by at most β. The map $\mathcal{T}^\sigma$ preserves order and contracts by at most γ for each fixed $\sigma \in \{\pi^-, \pi^+\}$. Their unique fixed points are denoted by $q^-_{\omega,H}$ and Q^σ. Iteration from any bounded measurable initialization approaches the fixed points at the rates β^n and γ^n stated in Appendix C.1. The value $q^-_{\omega,H}(x, a)$ is the expected return of the generated block rule and $Q^\sigma(x, a)$ is the expected return of executing a and then following σ. Exact evaluation of π^+ converges to its state value $V^{\pi^+}(x) = Q^{\pi^+}(x, \pi^+(x))$ at every state and desired goal.*

Proposition 1 separates the value of a generated block from the value of the executed policy, and the two differ by the discrepancy D_H defined below. The choice $H = 1$ with $A_\omega(x, a, \varepsilon) = (a)$ recovers ordinary reference policy evaluation of the same fixed policy. The learned sequence score and primitive estimate are denoted by $\widehat{q}_H$ and $\widehat{Q}$. Their true Bellman residuals are $\epsilon_H = \|\widehat{q}_H - \mathcal{B}^-_{\omega,H}\widehat{q}_H\|_\infty$ and $\epsilon_1 = \|\widehat{Q} - \mathcal{T}^{\pi^-}\widehat{Q}\|_\infty$, and the generated block discrepancy is $D_H = \|\mathcal{B}^-_{\omega,H} Q^{\pi^-} - Q^{\pi^-}\|_\infty$. The resulting value error bounds are $e_S = (\epsilon_H + D_H)/(1-\beta)$ and $e_1 = \epsilon_1/(1-\gamma)$. The discrepancy D_H includes differences in the generated first action and in all later actions. The errors are defined over the full evaluation domain. The policy score compares an admissible action with the reference action $a^- = \pi^-(x)$.

$$\begin{aligned}F_\lambda(x,a) &= \widehat{q}_H(x,a) - \widehat{q}_H(x,a^-) - \lambda\left(\tau - [\widehat{Q}(x,a) - \widehat{Q}(x,a^-)]\right)_+, \\ \eta &= \sup_{x \text{ nonterminal}}\left[\sup_{a \in \mathcal{A}(x)} F_\lambda(x,a) - F_\lambda(x, \pi^+(x))\right].\end{aligned} \tag{7}$$

where $0 \leq \lambda < \infty$ and $\tau \geq 0$ are fixed. The quantity η measures the largest remaining score improvement over all admissible actions. It includes optimization error and restrictions of the policy class. The residuals and η are full domain quantities for the fixed functions and policy and are distinct from the mean sampled training loss. The true reference advantage of the candidate is $\mathrm{Adv}^-(x) = Q^{\pi^-}(x, \pi^+(x)) - V^{\pi^-}(x)$. Its spread is $\rho_{\mathrm{adv}} = \mathrm{sp}(\mathrm{Adv}^-)$ where $\mathrm{sp}(u) = \sup_x u(x) - \inf_x u(x)$. The predicted primitive gap is $\widehat{\Delta}_1(x) = \widehat{Q}(x, \pi^+(x)) - \widehat{Q}(x, \pi^-(x))$. Both advantages are assigned zero at terminal inputs, because no later reward or value remains after termination.

Theorem 1 (Optimality gap of the learned policy)**.** *The conditions of Proposition 1 hold and the learned estimates are bounded and measurable. Define* $c = \eta + 2e_S$ *and* $b = c + \frac{\lambda}{1+\lambda}(2e_1 - c)_+$. *The state value of the candidate policy satisfies*

$$\|V^* - V^{\pi^+}\|_\infty \leq \frac{b + \gamma\rho_{\mathrm{adv}}}{1-\gamma} \leq \frac{b + \gamma[\mathrm{sp}(\widehat{\Delta}_1) + 4e_1]}{1-\gamma}. \tag{8}$$

Its action value satisfies

$$\|Q^* - Q^{\pi^+}\|_\infty \leq \frac{\gamma(b + \gamma\rho_{\mathrm{adv}})}{1-\gamma}. \tag{9}$$

If $\pi^+ = \pi^-$*, then* $\rho_{\mathrm{adv}} = 0$*. If also* $\epsilon_H = D_H = \epsilon_1 = \eta = 0$*, then* $V^{\pi^+} = V^*$ *and* $Q^{\pi^+} = Q^*$.

The bound connects sequence evaluation and the primitive penalty to the value of the learned policy. The sequence residual and generated block discrepancy control e_S. The primitive error limits changes in action ordering caused by the penalty. The advantage spread accounts for the difference between the candidate and reference policies. The common margin enters through the selection defect η without an additional margin term. The comparison uses Bellman value relations for fixed policies (Kakade & Langford, 2002).

5 Experiments

Three questions organize the experiments. The first asks whether CGCPO reaches the goal more often than the compared methods, the second asks whether the primitive penalty changes action preferences near obstacles, and the third asks where CGCPO and $\mathrm{CGCPO}_{\mathrm{Prim}}$ differ in reach across the maze. CGCPO uses DDPG with HER on four Fetch and six PointMaze tasks, with five independent random seeds (Lillicrap et al., 2016; Andrychowicz et al., 2017).

5.1 Evaluation Results

We compare CGCPO with DDPG+HER, EBP, TMD and the online HER adaptation CQN AS* on Fetch and the first four maze tasks (Zhao & Tresp, 2018; Myers et al., 2025; Seo & Abbeel, 2025; Ghosh et al., 2021). As illustrated in Figure 3, each panel plots success rate against training epoch, with the four Fetch tasks in panels a through d and the six PointMaze tasks in panels e through j. The widest separation appears in the PointMaze panels, where CGCPO reaches the highest late success on Medium, CustomLarge and both AntMaze-Large tasks. In panel h, CGCPO stays clearly above EBP and DDPG+HER through late training, and in panels i and j it stays above GCSL and DDPG+HER. On the Fetch tasks CGCPO improves over DDPG+HER on both obstacle manipulation tasks. Across the ten tasks CGCPO stays within the top three on every one of them, and no other method comes close to the leading value on every task family. The lowest late success of CGCPO on any task is higher than that of every other method. The larger mazes have longer episodes and more internal walls, so useful actions receive reward after more steps. Those tasks also reward detours that first increase the distance to the goal, so immediate reward cannot separate a detour from a blocked action. Later estimates therefore matter most there, which is the setting that the sequence comparison is designed for. These gains motivate the following probes of action preferences near obstacles and of reach across the maze.

5.2 Action Preferences near Manipulation Obstacles

We vary the primitive penalty to test how it relates to lifting around the Pick&Place obstacle. CGCPO and $\mathrm{CGCPO}_{\mathrm{Cap20}}$ use penalty caps of two and twenty, while $\mathrm{CGCPO}_{\mathrm{NoPenalty}}$ removes the penalty.

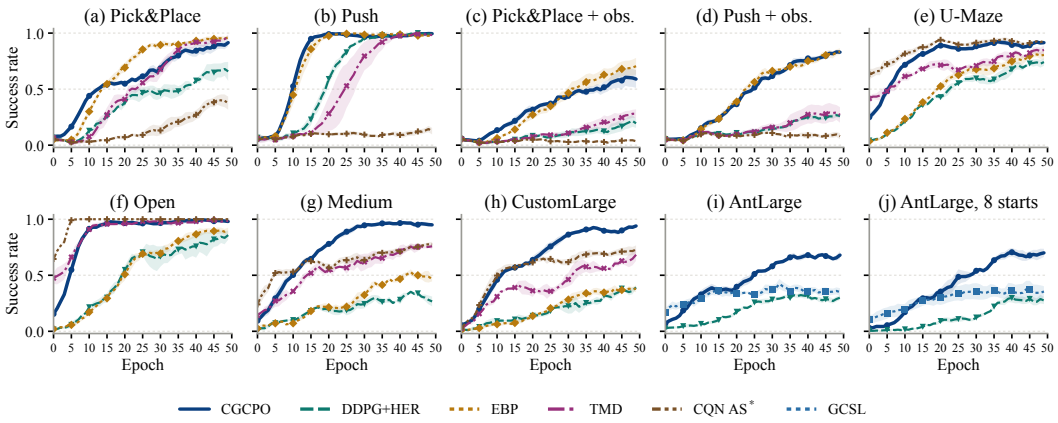


Figure 3: Success on four Fetch and six PointMaze tasks. Shaded bands show standard errors across the five independent training runs.

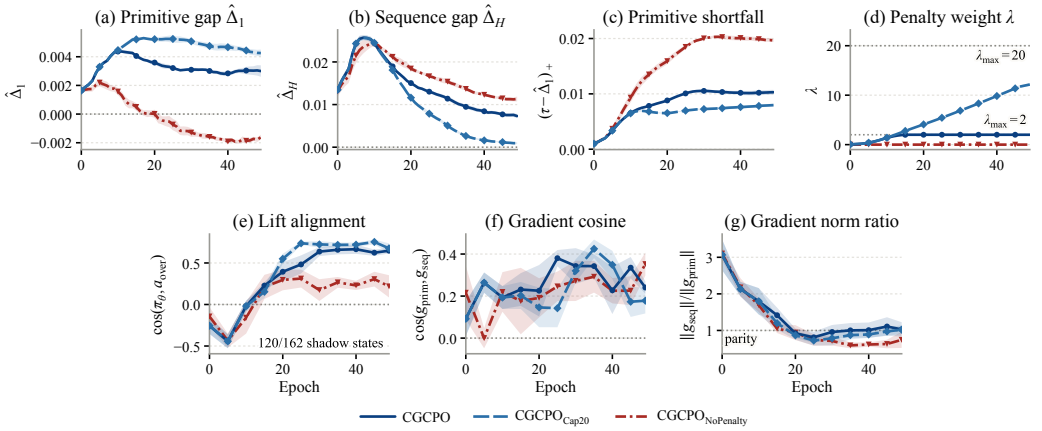


Figure 4: Value gaps, primitive penalties, lift alignment and action gradients on obstacle Pick&Place. Bands show standard errors across the five training runs of each penalty setting.

Lift alignment measures the cosine between the policy action and combined upward and horizontal motion toward the goal, on the states where a direct path to the goal crosses the obstacle. Figure 4 plots the two value gaps in panels a and b, the primitive shortfall with the penalty weight in panels c and d, and lift alignment with the two gradient measures in panels e through g.

As illustrated in Figure 4, the sequence gap of every setting peaks in early training and then declines, and the primitive gap separates the settings more sharply. A negative primitive gap means that the primitive critic prefers the reference action over the candidate action at the probed states. $\text{CGCPO}_{\text{NoPenalty}}$ holds the largest sequence gap through late training, while its primitive gap crosses zero after the early phase and stays negative, so its shortfall accumulates because no penalty term corrects that comparison. $\text{CGCPO}_{\text{Cap20}}$ keeps the largest positive primitive gap and the stronger lift alignment through late training, and CGCPO stays positive as well. The penalty weight explains the difference, because the weight of CGCPO reaches its cap in early training while the weight of $\text{CGCPO}_{\text{Cap20}}$ keeps growing toward a higher cap and correcting more shortfalls. A higher cap therefore leaves $\text{CGCPO}_{\text{Cap20}}$ with the larger late correction weight, which agrees with its smaller shortfall. This pattern agrees with Equation (5), because a larger active penalty gives primitive corrections more weight, whereas sequence guidance alone can favor actions that the primitive comparison rejects. The gradient panels show that the cosine between the two gradients stays positive while the norm ratio falls to near parity, so early updates follow sequence guidance and later updates receive comparable contributions. The answer to the second question is therefore that the penalty shifts the

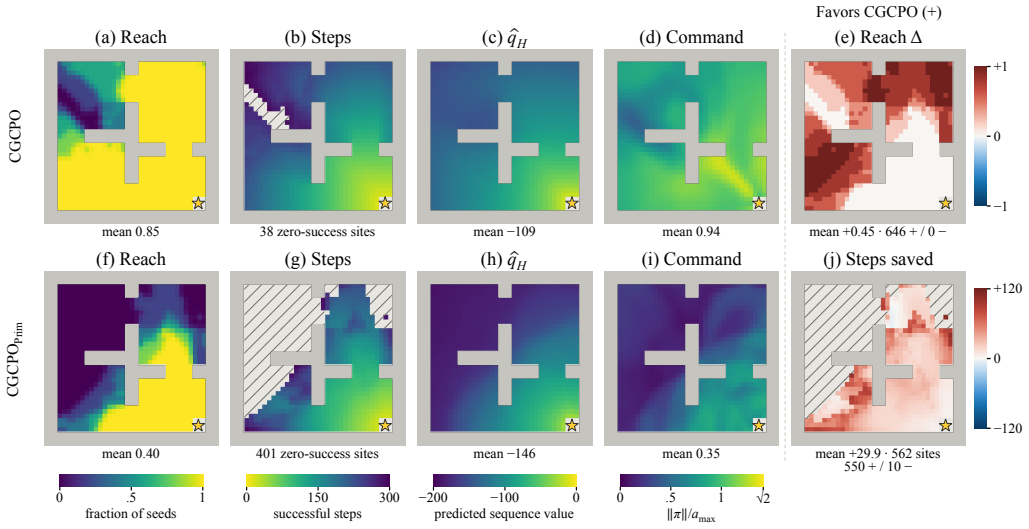


Figure 5: Reach, successful duration, sequence values and action magnitudes for CGCPO and $\mathrm{CGCPO}_{\mathrm{Prim}}$ on CustomLarge, with difference maps for reach and for successful duration.

policy toward lifting, with the corresponding success gain on obstacle Pick&Place in Table 2, where CGCPO and $\mathrm{CGCPO}_{\mathrm{Cap20}}$ reach 0.66 and 0.76 against 0.34 without the penalty.

5.3 Spatial Reachability around Maze Obstacles

We compare CGCPO with $\mathrm{CGCPO}_{\mathrm{Prim}}$, whose actor uses primitive values from a one-step target. Both methods start from shared starts toward a fixed goal in CustomLarge and share the generator, the replay and the evaluation settings. The maps therefore isolate the effect of the update rule. Reach is the fraction of trained policies that succeed at each site, and duration averages successful runs only.

As illustrated in Figure 5, the maps compare the two methods around the maze. Reach, successful duration, sequence values and action magnitudes appear for CGCPO in panels a through d and for $\mathrm{CGCPO}_{\mathrm{Prim}}$ in panels f through i. Panels e and j show the differences in reach and in successful duration. The reach map shows that CGCPO succeeds across most free cells while the reach map of $\mathrm{CGCPO}_{\mathrm{Prim}}$ leaves large regions unreached. Mean reach is 0.85 for CGCPO against 0.40 for $\mathrm{CGCPO}_{\mathrm{Prim}}$, and the difference map favors CGCPO at 646 sites and $\mathrm{CGCPO}_{\mathrm{Prim}}$ at none, especially in the upper and left passages separated from the goal by walls. The steps map shows that $\mathrm{CGCPO}_{\mathrm{Prim}}$ leaves many cells without a single successful run while CGCPO has far fewer such cells, and CGCPO reaches the goal in fewer steps where both methods succeed at least once. The action magnitude map shows that CGCPO commands action magnitudes close to the limit through those passages while $\mathrm{CGCPO}_{\mathrm{Prim}}$ commands smaller magnitudes, and the sequence value map orders the two methods in the same way. Where the two reach maps disagree, the disagreement always favors CGCPO, so the difference between them is a difference in coverage rather than in task competence. The answer to the third question is therefore that CGCPO extends coverage along the passages that a one-step comparison leaves unreached. Section 5.4 separates this effect from longer value targets with a four-step primitive target.

5.4 Ablation Studies

We compare sequence guidance with $\mathrm{CGCPO}_{\mathrm{TD4}}$, which extends the primitive target to four recorded transitions without sequence learning, and vary penalty strength and actor gradients. The comparisons below vary these ingredients and read the resulting gain on each task. The pinned controls $\mathrm{CGCPO}_{\mathrm{FullGrad}}$ and $\mathrm{CGCPO}_{\mathrm{SuffixGrad}}$ both score the policy action followed by a generated suffix under cap twenty. Only $\mathrm{CGCPO}_{\mathrm{FullGrad}}$ keeps the direct derivative through the first action, so the two settings differ only in the gradient path that reaches the actor parameters.

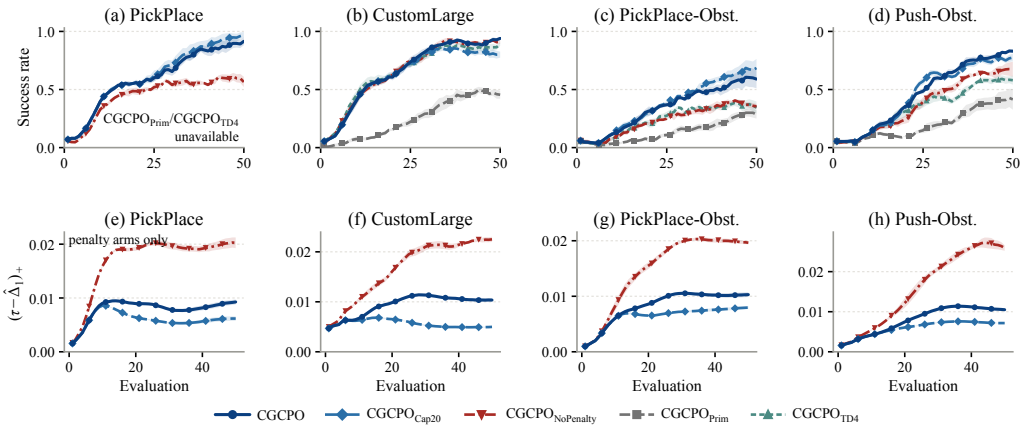


Figure 6: Success and primitive margin shortfalls under core ablations. Bands show standard errors.

As illustrated in Figure 6, success appears in panels a through d and primitive shortfalls in panels e through h. Sequence guidance supplies a gain on obstacle Pick&Place that longer targets do not, since the last-five mean of $\mathrm{CGCPO}_{\mathrm{TD4}}$ stays at 0.360 against 0.300 for $\mathrm{CGCPO}_{\mathrm{Prim}}$, while CGCPO and $\mathrm{CGCPO}_{\mathrm{Cap20}}$ reach 0.600 and 0.668. On CustomLarge the four-step target recovers most of the gain over $\mathrm{CGCPO}_{\mathrm{Prim}}$ where an episode requires many steps before the goal is reached, and CGCPO keeps the higher last-five mean of 0.928 ± 0.014 against 0.864 ± 0.021. $\mathrm{CGCPO}_{\mathrm{FullGrad}}$ exceeds $\mathrm{CGCPO}_{\mathrm{SuffixGrad}}$ on both tasks, so the direct first action derivative carries most of the sequence signal. Pinning the first scored action removes the mismatch between that action and its condition, while the later actions are still scored as an open loop, which is the mismatch that the primitive critic checks. Stronger penalties reduce primitive shortfalls, and the shortfall panels order the variants in the same way on all four tasks. Removing the penalty lowers final success on five of the six tasks in Table 2, with the largest drop on obstacle Pick&Place from 0.66 to 0.34. Across the six tasks of Table 2 the default setting attains the highest mean final success of any variant. The cap sets how strongly shortfalls are corrected, and the wider cap is higher on the two Pick&Place tasks while cap two is higher on obstacle Push, U Maze and CustomLarge. The default cap is two. Appendix A.3 also varies the penalty controller, the reference action and the generator, and none of these changes raises the mean final success above the default setting. The gain therefore does not depend on the flow generator or on the tracked reference policy in particular. Both the sequence guidance and the finite penalty carry a measured part of the gain, while the gradient control shows how the sequence signal reaches the actor.

6 Conclusion

Goal reaching near obstacles depends on comparisons that immediate feedback cannot supply, so CGCPO guides the action condition with imagined sequence returns while a separate primitive critic keeps the current action from falling below the chosen margin. For fixed policies the gap to the optimal value is bounded by the sequence and primitive estimation errors, the generated block discrepancy, the advantage spread and the selection defect. The penalty enters only through the last term and acts as a sampled correction rather than a condition imposed at every state and desired goal. CGCPO stays within the top three on all ten tasks, extends reach across passages that a primitive-only update leaves unreached, and supplies an obstacle manipulation gain that a four-step target does not reproduce.

Limitations. CGCPO finishes below EBP on both Pick&Place tasks and below TMD on plain Pick&Place, while the AntMaze-Large tasks compare two baselines only. The bound of Section 4.4 is an error decomposition rather than a tight performance guarantee, because the factor $1/(1-\gamma)$ is large compared with the gap observed in Appendix C.3, and it fixes the penalty weight and margin that the implementation adapts. The critic errors that the bound requires were not measured during training, and extending sequence guidance to the remaining manipulation tasks is left for future work.

AI USE STATEMENT

LLMs were used to polish the manuscript, develop the initial code, review mathematical derivations and assist with interpreting experimental results.

REPRODUCIBILITY STATEMENT

Task definitions and evaluation rules are specified in Appendix A, including the episode limits, success thresholds and return settings listed with each environment. The training protocol in Appendix A.1 covers replay sampling and goal relabeling, along with network structures and update settings. The behavior comparisons use spatial sampling and measures of reach and successful duration defined in Appendix B. Appendix C states the assumptions and gives complete proofs of both results. The code and the configuration files will be released upon publication.