EFFICIENT ONLINE ESTIMATION OF ACHIEVABLE GOAL DISTRIBUTIONS DURING TRAINING FOR GOAL-CONDITIONED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Goal-conditioned reinforcement learning (GCRL) trains a single policy to achieve diverse goals. Central to GCRL is the achievable goal distribution p_{ag}, the distribution of goals that the current policy can achieve. Accurate estimation of p_{ag} helps curriculum learning methods in GCRL sample behavioral goals (the goals used to collect training data) of intermediate difficulty, which substantially improves training efficiency. Estimating p_{ag} online, however, is challenging: the distribution evolves throughout training, and the estimator must track this moving target with data that are both fresh and sufficient. Existing methods meet only one of these requirements: replay-buffer-based estimation requires no extra environment interaction but is biased toward where past policies happened to visit, whereas policy evaluation faithfully reflects the current policy but demands a large number of fresh environment samples for every estimate. We empirically find that p_{ag} evolves smoothly during training, so historical policy evaluation data remain informative for estimating the current p_{ag}. Exploiting this finding, we propose PE-GCRL (GCRL with **P**eriodic Policy **E**valuation), which interleaves training with a small number of policy evaluations and estimates p_{ag} by reusing the accumulated historical evaluation data. Experiments on four mainstream GCRL tasks show that, across multiple curriculum algorithms, PE-GCRL reduces the estimation error of p_{ag} by roughly an order of magnitude, keeps behavioral goals at the frontier of achievability, and improves learning speed and final success rates with less than 5% training-time overhead.

1 INTRODUCTION

Goal-Conditioned Reinforcement Learning (GCRL) (Liu et al., 2022) aims to learn a versatile policy capable of achieving diverse target states, formulated as goals. By sharing knowledge across different goals, GCRL enables agents to master complex skills that are often intractable for standard Reinforcement Learning (RL) (Sutton & Barto, 2018). This paradigm has demonstrated success in various domains, ranging from high-dimensional control tasks (Pitis et al., 2020; Gong et al., 2025a) to complex strategy games (Schaul et al., 2015; Yuan et al., 2024).

A core concept in GCRL is the achievable goal distribution p_{ag}, i.e., the distribution of goals that the current policy is capable of achieving, which delimits the capability boundary of the policy. Accurate estimation of p_{ag} plays an important role in GCRL training. For instance, curriculum learning algorithms (Narvekar et al., 2020) rely on an online estimate of p_{ag}, denoted $\tilde{p}_{ag}$, to sample goals of intermediate difficulty (GOIDs) (Florensa et al., 2018; Castanet et al., 2023) for training data collection (the goals used to collect training data are referred to as behavioral goals), which substantially improves training efficiency. Consequently, the effectiveness of such curricula hinges on the accuracy of $\tilde{p}_{ag}$, and online estimation of p_{ag} becomes a critical bottleneck. This estimation is inherently challenging, since p_{ag} evolves throughout training: every policy update shifts the capability boundary, and the estimator must track this moving target with data that are both fresh (reflecting the current policy) and sufficient (covering the goal space).

Existing approaches handle this tension at opposite extremes; we review related work in detail in Appendix F. One class of methods (Pitis et al., 2020; Warde-Farley et al., 2018; Nair et al., 2018)

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.

pursues estimation efficiency at the cost of accuracy: they fit $\tilde{p}_{ag}$ from the achieved goals of past transitions in the replay buffer (RB) (Mnih et al., 2013), thereby avoiding extra environment interaction. The RB, however, accumulates data generated by outdated policies under considerable exploration noise; the resulting distribution of achieved goals reflects where the under-trained, exploring policy happened to visit, rather than where the current policy can reliably reach, rendering $\tilde{p}_{ag}$ biased and lagging behind the evolving capability boundary. Conversely, the other class of methods pursues estimation accuracy through policy evaluation (Rutherford et al., 2024; Gaven et al., 2025): whenever an estimate is needed, the current policy is evaluated, and the evaluation outcomes faithfully reflect the policy's current capability. Every such estimation, however, requires collecting a large number of fresh environment samples, severely compromising training efficiency. Estimating p_{ag} online with both accuracy and efficiency therefore remains an open problem.

We empirically analyze how p_{ag} evolves during training and find that it changes smoothly over time. This smoothness implies that policy evaluation results from earlier training stages remain informative for estimating the current p_{ag}, so they can be reused rather than re-collected. Motivated by this observation, we propose PE-GCRL (GCRL with **P**eriodic Policy **E**valuation), which interleaves training with a small number of periodic policy evaluations and estimates p_{ag} from the historical evaluation data. Our main contributions are summarized as follows:

- We identify the smooth temporal evolution of p_{ag} during training, which motivates reusing historical evaluation data for online estimation.
- We propose PE-GCRL, a general framework that estimates p_{ag} online via periodic policy evaluation, instantiated with recency-based weighting to balance accuracy and efficiency. The framework is orthogonal to the density estimator and the curriculum algorithm.
- Experiments on four tasks and multiple curriculum algorithms show that PE-GCRL consistently reduces the estimation error of p_{ag}, steers behavioral goals toward GOIDs, and improves curriculum learning performance.

2 PRELIMINARIES

GCRL can be described by a goal-augmented Markov Decision Process (Liu et al., 2022), denoted $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, r, \gamma, \mathcal{G}, p_{dg}, \phi \rangle$. Here, $\mathcal{S}$ and $\mathcal{A}$ represent the state and action spaces, respectively; γ is the discount factor; $\mathcal{G}$ is the goal space; and p_{dg} denotes the desired goal distribution. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition function, where $\mathcal{P}(\mathcal{X})$ denotes the set of probability distributions over a set $\mathcal{X}$. $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$ is the goal-conditioned reward function, and $\phi : \mathcal{S} \to \mathcal{G}$ is a tractable mapping that projects a state to a specific goal. The objective of GCRL is to learn a goal-conditioned policy $\pi : \mathcal{S} \times \mathcal{G} \to \mathcal{P}(\mathcal{A})$ that maximizes the expected return over the desired goal distribution: $J(\pi) = \mathbb{E}_{a_t \sim \pi(\cdot|s_t,g), g \sim p_{dg}, s_{t+1} \sim \mathcal{T}(\cdot|s_t,a_t)} [\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, g)]$.

Following Liu et al. (2022), we distinguish two additional goal types alongside the desired goal: *achievable goals*, i.e., goals that the policy can achieve on command (with distribution p_{ag}), and *behavioral goals*, i.e., the goals targeted for training data collection within a rollout episode. Training also gives rise to *achieved goals*: every visited state s_t realizes a goal $\phi(s_t)$, and these accumulate in the RB as training proceeds. In general, achieved and achievable goals differ: the former are realized by states visited in past rollouts, the latter those the current policy can achieve on command.

Pitis et al. (2020) distilled common GCRL techniques into a unified framework of five functions, reproduced in black in Algorithm 1 (our modifications are in blue and detailed in Section 3). SELECT generates behavioral goals for rollouts from p_{dg} or through a curriculum; ROLLOUT rolls out the policy conditioned on these goals and stores the transitions in the RB; RELABEL assigns a new goal to a transition, typically one of its achieved goals, and recomputes the reward, improving sample efficiency (Andrychowicz et al., 2017); OPTIMIZE updates the policy with an off-policy RL algorithm on RB data; and TRAIN optimizes the policy by alternating between ROLLOUT and OPTIMIZE.

Research on curriculum learning primarily focuses on generating behavioral goals within the SELECT function. Typically, this involves estimating p_{ag} and then sampling behavioral goals by integrating this estimate with p_{dg}. A predominant paradigm involves sampling from the policy's current capability boundary, where goals are of intermediate difficulty (GOIDs) (Florensa et al., 2018). Representative examples include the MEGA and OMEGA algorithms (Pitis et al., 2020) (see Ap-

pendix A for details). These methods utilize data from RB to estimate p_{ag}, denoted as $\tilde{p}_{ag}$, and employ a generate-and-test strategy (Newell, 1969) to produce behavioral goals.

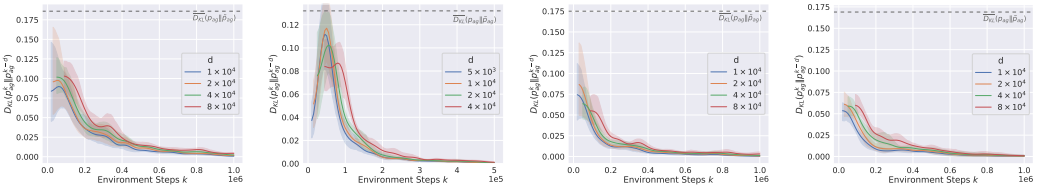
Specifically, MEGA samples N candidate goals $\{g_i\}_{i=1}^{N} \sim \tilde{p}_{ag}$ and selects the candidate with the minimum density, $\hat{g} = \arg\min_{g_i} \tilde{p}_{ag}(g_i)$. Building upon MEGA, OMEGA introduces an indicator $\alpha = 1/\max(b + D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag}), 1)$ to measure learning progress, thereby determining whether to sample behavioral goals from p_{dg} or via the MEGA method. A larger α implies a higher probability of sampling behavioral goals from p_{dg}. Here, b is a hyperparameter; a smaller b allows p_{dg} to influence the sampling of behavioral goals at an earlier stage. As $\tilde{p}_{ag}$ approximates p_{dg}, indicating that the policy has become more capable of achieving desired goals, the value of α increases, leading to a higher probability of sampling behavioral goals directly from p_{dg}.

3 GCRL WITH PERIODIC POLICY EVALUATION

This section proceeds in three steps: Section 3.1 empirically analyzes how p_{ag} evolves during training, Section 3.2 presents the PE-GCRL framework built upon this analysis, and Section 3.3 instantiates the general weighting scheme with recency-based weights.

3.1 EMPIRICAL ANALYSIS OF ACHIEVABLE GOAL DISTRIBUTION

As a first step toward designing an estimation method for p_{ag} that balances accuracy and efficiency, we empirically analyze how p_{ag} evolves during GCRL training (see Section 4.1 for the experimental settings). Fig. 1 shows the evolution of $D_{\mathrm{KL}}(p_{ag}^{k}\|p_{ag}^{k-d})$, the divergence between the achievable goal distributions of the policies at environment steps k and $k-d$, during training on the Push task for four algorithms: GC-SAC, HER, OMEGA-RB, and OMEGA-PE, where OMEGA-RB and OMEGA-PE are OMEGA variants that estimate p_{ag} from the RB and from policy evaluations, respectively. For reference, the dashed horizontal line in each subplot marks the average error of estimating p_{ag} from the RB throughout training, $\overline{D_{\mathrm{KL}}}(p_{ag}\|\tilde{p}_{ag})$.



(a) GC-SAC (b) HER (c) OMEGA-RB (d) OMEGA-PE

Figure 1: $D_{\mathrm{KL}}(p_{ag}^{k}\|p_{ag}^{k-d})$ between the achievable goal distributions of the policy at environment steps k and $k-d$ during training on the Push task; each curve corresponds to a different time lag d. For reference, the dashed horizontal line marks the average estimation error of p_{ag} throughout training, $\overline{D_{\mathrm{KL}}}(p_{ag}\|\tilde{p}_{ag})$, where $\tilde{p}_{ag}$ is estimated from the RB.

The results reveal two patterns. First, for every algorithm and every lag d, the KL curves peak early in training, decrease steadily thereafter, and eventually level off near zero; in other words, p_{ag} evolves ever more slowly as training proceeds. The evolution of p_{ag} during training is thus smooth and regular, and such smoothness makes p_{ag} easier to predict. To corroborate this smoothness through comparison, we contrast $D_{\mathrm{KL}}(p_{ag}^{k}\|p_{ag}^{k-d})$ with $J^{k}(\pi) - J^{k-d}(\pi)$ for all four algorithms on all four tasks in Appendix E.1, and further quantify the trends of both with the trajectory smoothness metric $\sigma(\cdot)$ (Mysore et al., 2021) (formula in Appendix D.4): the KL curves are far smoother than the return-difference curves, confirming the smooth evolution of p_{ag}.

Second, throughout training, including at the early-training peaks, the KL curves of all algorithms lie well below the dashed reference line, and the smaller the lag d, the lower the curve.

These two patterns carry three implications for the design of the estimator. (i) The achievable goal distributions of policies at adjacent time steps are nearly identical, so historical evaluation data remain representative of the current p_{ag} over a long horizon; p_{ag} can thus be estimated by reusing them, and the evaluations need not be performed frequently. (ii) The temporal drift of p_{ag}—the error

Algorithm 1 PE-GCRL

function TRAIN(policy π_θ, algorithm $\mathcal{A}$, $*args$)
  Initialize experience replay buffer $\mathcal{B}$, evaluation result buffer $\mathcal{B}_e$
  while π_θ not converged **do**
    EVALUATE(π_θ, $\mathcal{B}_e$) ▷ Evaluate policy
    ROLLOUT(π_θ, $\mathcal{B}$, $\mathcal{B}_e$) ▷ Collect experience
    OPTIMIZE(π_θ, $\mathcal{B}$, $\mathcal{A}$) ▷ Optimize policy
  end while
end function

function ROLLOUT(policy π_θ, experience replay buffer $\mathcal{B}$, evaluation result buffer $\mathcal{B}_e$, $*args$)
  $g \leftarrow$ SELECT($\mathcal{B}_e$, $*args$), $s_0 \leftarrow$ initial state
  for t in $0 \dots T-1$ **do**
    $a_t, r_t, s_{t+1} \leftarrow$ execute $\pi_\theta(s_t, g)$ with some exploration strategy in environment
    Store $(s_t, a_t, s_{t+1}, r_t, g)$ in $\mathcal{B}$
  end for
end function

function OPTIMIZE(policy π_θ, experience replay buffer $\mathcal{B}$, algorithm $\mathcal{A}$)
  Sample mini-batch $B = \{(s, a, s', r, g)_i\}_{i=1}^m \sim \mathcal{B}$
  $B' \leftarrow$ RELABEL(B, $*args$)
  Optimize π_θ using $\mathcal{A}$ and relabeled B'
end function

function SELECT(desired goal distribution p_{dg}, evaluation result buffer $\mathcal{B}_e$, $*args$)
  Estimate achievable goal distribution $\tilde{p}_{ag}$ from the data in $\mathcal{B}_e$, and return a behavioral goal by jointly considering $\tilde{p}_{ag}$ and p_{dg}.
end function

function RELABEL(transition batch B, $*args$)
  Relabel goals and rewards in minibatch B according to a specific strategy.
end function

function Evaluate(policy π_θ, evaluation result buffer $\mathcal{B}_e$)
  Evaluate π_θ on sampled desired goals and store results in $\mathcal{B}_e$ upon triggering the evaluation.
end function

that historical evaluation data incur through staleness—stays below the RB-based estimation error throughout training; the bottleneck of p_{ag} estimation therefore lies in the data source (the achieved goals recorded in the RB do not coincide with the goals achievable by the current policy) rather than in the timeliness of the data. (iii) Since a smaller lag d implies a smaller divergence, evaluations collected closer to the current time step should contribute more to the estimate, which motivates weighting the historical data by their recency.

Overall, p_{ag} evolves smoothly and drifts little over time; as long as the data source is appropriate, low-frequency evaluations suffice for both accurate and efficient estimation.

3.2 THE PE-GCRL FRAMEWORK

Guided by the preceding analysis, we propose PE-GCRL, whose overall procedure is outlined in Algorithm 1. PE-GCRL modifies the unified five-function training framework of Section 2 in only two respects (the blue parts of Algorithm 1). First, TRAIN periodically interleaves EVALUATE, which evaluates the current policy on a set of desired goals and stores the evaluation results in an evaluation result buffer $\mathcal{B}_e$. Second, in SELECT, the data source for estimating $\tilde{p}_{ag}$ is changed from the data in the RB to the historical evaluation data in $\mathcal{B}_e$. All remaining functions are unchanged, so PE-GCRL composes with arbitrary implementations of SELECT, RELABEL, and OPTIMIZE.

Specifically, each evaluation result is stored in $\mathcal{B}_e$ as a tuple (g_i, b_i, c_i): g_i is the desired goal being evaluated; $b_i \in \{0, 1\}$ is a success flag that equals 1 if the policy achieves g_i; and $c_i := I(\pi_{t_i}, t_i)$ is the characteristic of this evaluation, where the function I maps the policy under evaluation and the training time to the evaluation characteristic. Compared with the RB, this data source is aligned with the definition of p_{ag}: an evaluation directly tests which goals the current policy can achieve, and the goals g_i with $b_i = 1$ are exactly samples of achievable goals, eliminating at the source the bias caused by the mismatch between achieved goals and achievable goals.

As the samples in $\mathcal{B}_e$ come from policies at different training times, and staler samples deviate more from the current p_{ag}, PE-GCRL—following implication (iii)—assigns each sample a weight that decays exponentially with the distance between evaluation characteristics: at training time t, a sample (g_i, b_i, c_i) has weight $w_i = e^{-d\left(I(\pi_t,t),\, I(\pi_{t_i},t_i)\right)/\tau}$, where $d(\cdot,\cdot)$ is a distance between the characteristics of two evaluations, and $\tau > 0$ is a temperature parameter. When estimating p_{ag}, PE-GCRL performs a weighted fit over the historical data in $\mathcal{B}_e$. The specific density estimator for this fit is a free design choice orthogonal to the framework.

Two limiting cases bracket the role of τ: as $\tau \to +\infty$, $w_i \to 1$ and all historical data are reused nearly equally; as $\tau \to 0^+$, only the latest data retain non-negligible weights. In between, τ balances the amount of data against its freshness: a larger τ yields a more stable but staler estimate, whereas a smaller one yields fresher data at the price of fewer samples and larger estimation noise. Since p_{ag} evolves smoothly and its temporal drift stays below the RB-based estimation error throughout training, a moderately large τ suffices to keep the data fresh while accumulating enough of them.

3.3 A Recency-Based Instantiation

The framework of the preceding subsection leaves the evaluation characteristic function I and the distance $d(\cdot,\cdot)$ open. Since the staleness of an evaluation is determined by the training stage at which it was conducted, we take the characteristic to be the training time itself, $I(\pi_t, t) = t$ (hence $c_i = t_i$), and the distance between two characteristics to be the elapsed time, $d(t_1, t_2) = |t_2 - t_1|$. Substituting both into the general weighting scheme yields $w_i = e^{-(t-t_i)/\tau}$ ($t \geq t_i$): each weight decays exponentially with the time elapsed since the evaluation, and the scheme reduces to weighting the historical evaluation data purely by their recency—a natural measure of staleness, since Section 3.1 shows that the divergence between the p_{ag} at two training times shrinks as their gap decreases.

Beyond this empirical motivation, the exponentially decaying form receives theoretical support. In Appendix B, we show that the detrended evolution of p_{ag} is approximately weakly stationary, and therefore model it as a first-order auto-regressive (AR(1)) process (Box et al., 2015). The autocorrelation of an AR(1) process decays exponentially with the time gap, so the relevance of an evaluation at t_i to the current p_{ag} likewise decays exponentially with $t - t_i$—precisely the form of the weights.

4 Experiments

4.1 Settings

We evaluate PE-GCRL on four tasks spanning two simulation engines: the manipulation tasks **Reach**, **Push**, and **Slide** on a Franka Emika Panda robotic arm (Gallouédec et al., 2021), and **Velocity Vector Control** (VVC) on the FlyCraft fixed-wing aircraft simulation engine (Gong et al., 2025a). These four tasks possess distinct characteristics; see Appendix C for details.

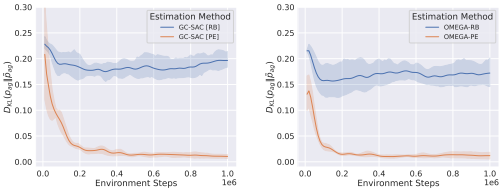
We compare three algorithms. **GC-SAC** directly applies SAC (Haarnoja et al., 2018) to GCRL: behavioral goals are sampled exclusively from p_{dg} during SELECT. **OMEGA-RB** augments GC-SAC with the OMEGA curriculum: during SELECT, $\tilde{p}_{ag}$ is estimated from the achieved goals in the **RB**, and behavioral goals are drawn from $\tilde{p}_{ag}$ and p_{dg}. **OMEGA-PE**, the practical implementation of PE-GCRL, differs from OMEGA-RB only in estimating $\tilde{p}_{ag}$ from the historical evaluation data in $\mathcal{B}_e$. See Appendices D and E for implementation details and additional baselines.

All evaluations test the policy on desired goals sampled from p_{dg}. Two divergence metrics recur throughout this section: the estimation error $D_{\mathrm{KL}}(p_{ag} \| \tilde{p}_{ag})$, measuring how well $\tilde{p}_{ag}$ fits p_{ag}, and the goal-achievement discrepancy $D_{\mathrm{KL}}(p_{dg} \| \tilde{p}_{ag})$, from which OMEGA computes the learning progress α. The former requires the ground-truth p_{ag}; how we approximate it and compute both

divergences is detailed in Appendices D.1 and D.2. Training performance is measured by the goal achievement success rate recorded during training, and the training overhead by the total training time. All results are averaged over five random seeds; the shaded regions denote the 95% confidence intervals of the mean, and all curves are smoothed with a Gaussian filter ($\sigma = 2$) for visual clarity.

4.2 Accuracy of p_{ag} Estimation

We first verify the core claim of PE-GCRL: estimating p_{ag} from historical evaluation data is more accurate than estimating it from replay-buffer data. To this end, we compare the estimation error $D_{\mathrm{KL}}(p_{ag} \| \tilde{p}_{ag})$ of the two data sources under two training regimes on Push. In the first regime, the policy is trained with GC-SAC, and both estimates are computed purely for measurement: neither influences behavioral goal selection, so the comparison isolates the effect of the data source itself. In the second regime, the estimates directly drive the curriculum, as in OMEGA-RB and OMEGA-PE, where $\tilde{p}_{ag}$ governs behavioral goal sampling and thereby shapes the contents of the RB; the comparison thus reflects the accuracy attained in actual use. Fig. 2 presents the results.



(a) GC-SAC (b) OMEGA

Figure 2: Estimation error $D_{\mathrm{KL}}(p_{ag} \| \tilde{p}_{ag})$ of the RB-based and the evaluation-based estimates during training on Push. (a) GC-SAC [RB] and GC-SAC [PE] estimate $\tilde{p}_{ag}$ from the RB and from historical evaluation data, respectively; the estimates are not used to direct behavioral goal selection. (b) OMEGA-RB and OMEGA-PE, where the estimate drives behavioral goal sampling.

First, in both regimes, the RB-based estimate incurs a persistently high error with no clear downward trend. Second, the evaluation-based estimate shows a brief error spike at the very beginning of training, when $\mathcal{B}_e$ still contains only a few evaluation results; the error then drops rapidly to a level roughly an order of magnitude below that of the RB-based estimate, and remains low thereafter. The gap thus persists throughout training, both when the estimates are measured passively and when they actively drive the curriculum.

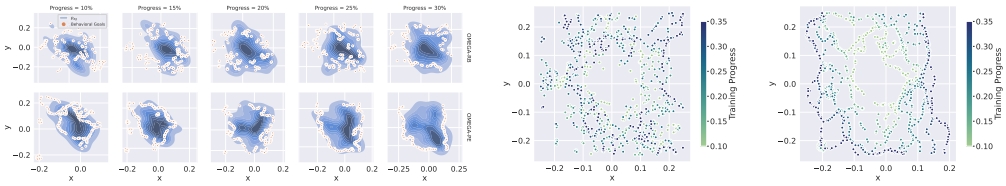
These results substantiate implication (ii) of Section 3.1: the bottleneck of online p_{ag} estimation lies in the data source rather than in the timeliness of the data. The achieved goals stored in the RB are generated by past policies under exploration noise; they record where those policies happened to visit rather than which goals the current policy can achieve, and fitting $\tilde{p}_{ag}$ to them therefore incurs a bias that does not vanish as the buffer grows. In contrast, the goals achieved during evaluation are samples of achievable goals by construction (Section 3.2), so no such bias arises. Moreover, as p_{ag} evolves smoothly, the recency-weighted historical evaluations in $\mathcal{B}_e$ remain representative of the current p_{ag}, keeping the error introduced by data staleness small.

Repeating the same comparison on all four tasks, with GC-SAC, HER, and OMEGA as the training algorithms, leads to the same conclusion in every combination (Appendix E.2): the accuracy advantage of the evaluation data source holds regardless of task and training algorithm.

4.3 Benefits for Curriculum Learning

Section 4.2 established that the evaluation data source yields a substantially more accurate $\tilde{p}_{ag}$. We trace how this accuracy benefits the curriculum built on top of it, by examining its two components in turn: where behavioral goals fall, and when their sampling is handed over to p_{dg}.

Behavioral goals fall on the frontier of achievability. Fig. 3a overlays the behavioral goals sampled by OMEGA-RB and OMEGA-PE on the ground-truth p_{ag} at five stages of Push training (10%–30%, the window in which most behavioral goals are still generated via MEGA; Appendix D.3). Recall from Section 2 that both methods sample behavioral goals via MEGA: N candidates are drawn from $\tilde{p}_{ag}$ and the one with the lowest estimated density is kept, so that behavioral goals should fall at the low-density boundary of p_{ag}—the GOIDs. OMEGA-PE behaves exactly in this way: its behavioral goals rapidly spread along the boundary of p_{ag} and track it as it expands outward with training. OMEGA-RB fails to do so: its behavioral goals scatter over the interior of the visited region, in-

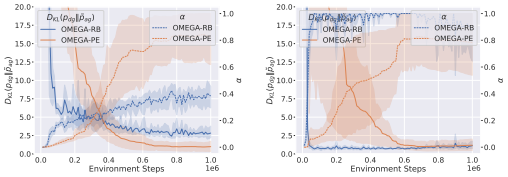


(a) Behavioral goals on p_{ag} (b) OMEGA-RB (c) OMEGA-PE

Figure 3: (a) Behavioral goals sampled during SELECT (orange points) overlaid on the ground-truth p_{ag} (blue shading) at five stages of Push training (10%–30%); rows correspond to OMEGA-RB (top) and OMEGA-PE (bottom), and the axes to the two Push goal-space dimensions. (b, c) Behavioral goals sampled throughout training, colored by training progress (colorbars); the inner and outer regions correspond to easy and hard goals, respectively. OMEGA-PE exhibits a clear progression from the inner to the outer region, whereas OMEGA-RB shows no distinct trend.

cluding its high-density core. When $\tilde{p}_{ag}$ is biased, its lowest-density candidates no longer coincide with the frontier of achievability, and the sampled goals largely miss the GOIDs. The same contrast is observed on Slide (Appendix E.2). Over the entire training, the behavioral goals of OMEGA-PE progress from the inner, easy region toward the outer, hard one, whereas those of OMEGA-RB exhibit no distinct trend (Figs. 3b and 3c).

The learning progress α is tracked reliably. The second component, α, is computed from $D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag})$, so an inaccurate $\tilde{p}_{ag}$ corrupts it—in either direction, as Fig. 4 shows. On Slide, the RB-based $\tilde{p}_{ag}$ is over-dispersed and resembles p_{dg}; the discrepancy collapses within the first 5% of training, driving α to nearly 1 immediately (Fig. 4b). The curriculum is thereby terminated prematurely: behavioral goals are drawn almost exclusively from p_{dg} long before the policy can achieve the desired goals. On Push, the failure runs in the opposite direction (Fig. 4a): the RB-based discrepancy remains several times the evaluation-based one throughout training, so α stalls below 0.4 and the handover to p_{dg} barely proceeds, even though the policy continues to improve (Section 4.4). In contrast, the evaluation-based discrepancy decreases steadily as the policy genuinely expands its achievable set, and the resulting α rises smoothly in step with actual progress, reaching approximately 0.85 on Push. Distortions of the same kind appear on Reach and VVC (Appendix E.2).



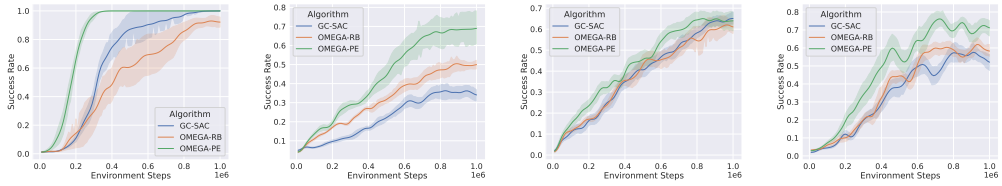
(a) Push (b) Slide

Figure 4: Trends of the goal-achievement discrepancy $D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag})$ (solid, left axis) and the learning progress α (dashed, right axis; Section 2) for OMEGA-RB (blue) and OMEGA-PE (orange) during training on Push and Slide.

4.4 TRAINING PERFORMANCE AND OVERHEAD

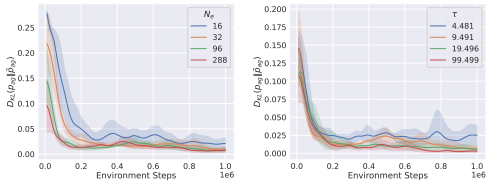
Having traced the improvements along the curriculum, we examine the training performance they are meant to deliver, and the computational cost they incur.

OMEGA-PE learns faster and attains higher success rates. Fig. 5 reports the goal achievement success rate recorded during training: on all four tasks, OMEGA-PE exhibits the fastest ascent. By the end of training, it converges to the highest success rate on Push and VVC; on Reach, it approaches a success rate of nearly 1 within roughly the first third of training, whereas GC-SAC attains a comparable level only near the end of training and OMEGA-RB settles at a lower one; on Slide, the three methods reach similar final levels, with OMEGA-PE leading for most of training. These gains stem from the improvements traced in the preceding subsections: the accurate $\tilde{p}_{ag}$ keeps behavioral goals at the frontier of achievability and makes the handover to p_{dg} follow the actual pace of learning, so the policy trains on goals of appropriate difficulty throughout. The advantage is not specific to the OMEGA curriculum: substituting the evaluation data source for the RB likewise reduces the estimation error of p_{ag} under the additional curricula ODISCERN and ORIG, and the



(a) Reach (b) Push (c) Slide (d) VVC

Figure 5: Goal achievement success rate of GC-SAC, OMEGA-RB, and OMEGA-PE recorded during training on the four tasks.



(a) N_e (b) τ

Figure 6: Ablations on N_e and τ with OMEGA-PE on Push, measured by $D_{\mathrm{KL}}(p_{ag}\|\tilde{p}_{ag})$.

Table 1: Total training time (in seconds) of different methods on the four tasks. Values in parentheses indicate the relative increase with respect to GC-SAC. Results are averaged over five random seeds.

	GC-SAC	OMEGA-RB	OMEGA-PE
Reach	28651	34023 (18.75%↑)	29441 (2.76%↑)
Push	39809	42386 (6.47%↑)	41622 (4.55%↑)
Slide	40360	40890 (1.31%↑)	41445 (2.69%↑)
VVC	41769	42263 (1.18%↑)	42239 (1.13%↑)

resulting gain in success rate varies with how strongly each curriculum's goal sampling depends on the accuracy of $\tilde{p}_{ag}$ (Appendix E.5).

The training overhead remains small. Table 1 reports the total training time. Relative to GC-SAC, OMEGA-PE increases the training time by less than 5% on all four tasks. Compared with OMEGA-RB, it consumes less time on Reach and Push, a comparable amount on VVC, and slightly more on Slide. The small overhead follows directly from the design of PE-GCRL. First, following implication (i) of Section 3.1, evaluations are interleaved at a low frequency (see Appendix E.3 for the specific numbers). Second, each estimation is an offline weighted fit on the compact tuples accumulated in $\mathcal{B}_e$, requiring no extra data collection or policy updates. The framework thus adds mainly environment rollouts, and in the inexpensive simulation environments studied here, the wall-clock time of off-policy training is dominated by OPTIMIZE, which repeatedly reuses RB data to update the policy; interleaving EVALUATE perturbs the total training time only marginally.

4.5 ABLATION STUDIES

The final ablation studies, conducted on Push with OMEGA-PE, examine in turn the estimation-side hyperparameters (N_e and τ), the curriculum-side ones (N and b), and the density estimator; default values are listed in Appendix D.5.

The estimation side: N_e and τ. Figs. 6a and 6b report the estimation error under different numbers of episodes per evaluation, N_e, and different temperatures τ. Increasing N_e from 16 to 96 steadily reduces the error, whereas the curves of $N_e = 96$ and $N_e = 288$ nearly coincide—the marginal benefit of enlarging the evaluation budget diminishes quickly, so a moderate N_e suffices. In addition to holding N_e fixed, we also examined letting it vary over training; see Appendix E.4 for the corresponding analysis. A larger τ, which retains more historical evaluations in each fit, leads to a lower error: p_{ag} evolves smoothly on Push (Section 3.1), so even stale evaluations remain representative, and the gain in data volume outweighs the loss in freshness—the tradeoff articulated in Section 3.2. Accordingly, both hyperparameters should match how fast p_{ag} evolves on the task: a smoothly evolving one favors a larger τ with a moderate N_e, a fast-changing one the opposite.

The curriculum side: N and b. Fig. 7 overlays the behavioral goals sampled by MEGA on the ground-truth p_{ag} for $N = 10$, 20, and 100: with $N = 10$, the retained goals scatter over the interior of the achievable region, whereas larger N brings them closer to the low-density boundary of p_{ag}—the GOIDs—at every stage examined; N should thus be set as large as the computational budget allows. The constant b acts through the learning progress, $\alpha = 1/\max(b + D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag}), 1)$: as

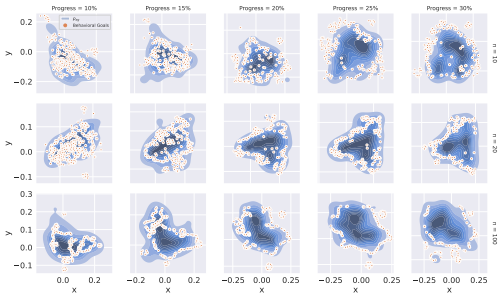
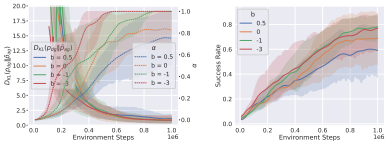


Figure 7: Behavioral goals sampled via MEGA (points), overlaid on the ground-truth p_{ag} (shading), for OMEGA-PE on Push. Rows correspond to $N = 10$, 20, and 100 (top to bottom), and columns to training progress (10%–30% of training; Appendix D.3).



(a) $D_{\mathrm{KL}}(p_{dg} \| \tilde{p}_{ag})$ and α (b) Success rate

Figure 8: Ablation of the balancing factor b with OMEGA-PE on Push. (a) Goal-achievement discrepancy $D_{\mathrm{KL}}(p_{dg} \| \tilde{p}_{ag})$ (solid, left axis) and learning progress α (dashed, right axis). (b) Corresponding goal achievement success rates.

Fig. 8 shows, varying b leaves the discrepancy itself nearly unchanged, but a smaller b makes α rise faster, so that the handover to p_{dg} occurs earlier. From the success rates, allowing p_{dg} to dominate behavioral goal sampling earlier helps the policy improve performance more quickly on the Push task, although how early this handover should occur remains task-dependent.

The choice of density estimator matters little here. We ablate the estimator itself, which Section 3.2 left open. Fig. 9 compares five estimators on Push: kernel density estimation (KDE) (Kristan et al., 2010), Gaussian mixture models (GMM) (Reynolds et al., 2009), neural networks (NN), normalizing flows (NF) (Grathwohl et al., 2019), and flow matching (FM) (Lipman et al., 2023). Under the evaluation data source, all five keep the error low, with a spread far smaller than the gap between the two data sources (Section 4.2). The nonparametric KDE, the simplest of the five, is the lowest and steadiest throughout training, whereas NF is the highest and most volatile during the middle stage, and GMM, NN, and FM start higher before converging close to KDE. The estimation accuracy of PE-GCRL thus derives from the evaluation data source rather than the expressiveness of the estimator, at least in the low-dimensional goal spaces studied here. Which estimator to prefer likely depends on the goal dimensionality: simple estimators such as KDE and GMM work well in this regime, whereas neural ones such as NN, NF, and FM are likely to perform better in higher dimensions.

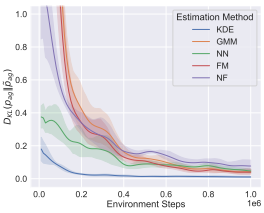


Figure 9: Estimation error $D_{\mathrm{KL}}(p_{ag} \| \tilde{p}_{ag})$ under five density estimators with OMEGA-PE on Push.

5 Conclusion and Limitations

Estimating the achievable goal distribution p_{ag} online confronts curriculum-based GCRL with an accuracy-efficiency dilemma: replay-buffer-based estimation requires no extra interaction but is biased, whereas policy evaluation is faithful but demands fresh samples for every estimate. Exploiting the smooth evolution of p_{ag} during training, PE-GCRL interleaves training with low-frequency policy evaluations and reuses the accumulated evaluation data with recency-based weighting; across four tasks and multiple curriculum algorithms, this reduces the estimation error by roughly an order of magnitude, keeps behavioral goals at the frontier of achievability, and improves learning speed and final success rates with less than 5% training-time overhead.

Three directions are worth exploring. First, estimation accuracy comes at the price of extra evaluation interaction; on sampling-expensive tasks such as physical robots, the biased replay-buffer data can still serve as a reference, and combining RB-based and evaluation-based estimation deserves further study. Second, the evaluation characteristic function I currently takes its simplest form, the training time itself, $I(\pi_t, t) = t$; a policy-dependent characteristic that is easy to compute would measure data staleness more directly. Third, the goal spaces of our tasks are all low-dimensional; verifying that the benefits of the evaluation data source persist on high-dimensional ones, with more expressive density estimators, is left for future work.

AI USE STATEMENT

In this work, we did not use generative AI tools for any tasks that require disclosure under the ICLR 2027 AI Policy for Authors. We used generative AI tools solely for language editing, i.e., polishing the grammar, phrasing, and overall readability of the manuscript, which falls under tasks with recommended disclosure. We have reviewed all AI-assisted work: every AI-assisted edit was checked by the authors to ensure correctness and to preserve the intended technical meaning. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All experiments in this work are conducted in simulation and involve no human subjects, no personal or sensitive data, no potentially harmful insights, methodologies, or applications, and no conflicts of interest; the work therefore raises no ethical concerns.

REPRODUCIBILITY STATEMENT

The source code of PE-GCRL and all baselines, together with the routines used to compute the reported metrics and the random seeds of all experiments, is available at `https://anonymous.4open.science/r/pe_gcrl-DB4F/`. Appendix C details the configurations of the four tasks; Appendix D.5 provides the complete implementation details, including the hyperparameters of every algorithm and of the density estimators; and Appendices D.1 and D.2 describe how the ground-truth p_{ag} is approximated and how the divergence metrics are computed. All results are averaged over five random seeds (Section 4.1).

REFERENCES

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

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Nicolas Castanet, Sylvain Lamprier, and Olivier Sigaud. Stein variational goal generation for reinforcement learning in hard exploration problems. *International Conference on Machine Learning*, 2023.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. C-learning: Learning to achieve goals via recursive classification. In *International Conference on Learning Representations*, 2020.

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

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-source goal-conditioned environments for robotic learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021.

Loris Gaven, Thomas Carta, Clément Romac, Cédric Colas, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. Magellan: Metacognitive predictions of learning progress guide autotelic llm agents in large goal spaces. In *International Conference on Machine Learning*. PMLR, 2025.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*, 2020.

Xudong Gong, Dawei Feng, Kele Xu, Weijia Wang, Zhangjun Sun, Xing Zhou, Si Zheng, Bo Ding, and Huaimin Wang. Vvc-gym: A fixed-wing uav reinforcement learning environment for multi-goal long-horizon problems. In *International Conference on Learning Representations*, 2025a.

Xudong Gong, Sen Yang, Feng Dawei, Kele Xu, Bo Ding, Huaimin Wang, and Yong Dou. Improving the continuity of goal-achievement ability via policy self-regularization for goal-conditioned reinforcement learning. In *Forty-second International Conference on Machine Learning*. PMLR, 2025b.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Matej Kristan, Danijel Skočaj, and Ales Leonardis. Online kernel density estimation for interactive learning. *Image and Vision Computing*, 28(7):1106–1116, 2010.

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

Minghuan Liu, Menghui Zhu, and Weinan Zhang. Goal-conditioned reinforcement learning: Problems and solutions. *International Joint Conference on Artificial Intelligence*, 2022.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing action policies for smooth control with reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1810–1816. IEEE, 2021.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in Neural Information Processing Systems*, 31, 2018.

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

Allen Newell. *Artificial intelligence and the concept of mind*. Carnegie-Mellon University, Department of Computer Science, 1969.

Panda-Gym. `https://panda-gym.readthedocs.io/en/latest/usage/manual_control.html`, 2024.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

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

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

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

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009.

Sheldon M Ross. *Stochastic processes*. John Wiley & Sons, 1995.

Alex Rutherford, Michael Beukman, Timon Willi, Bruno Lacerda, Nick Hawes, and Jakob Foerster. No regrets: Investigating and improving regret approximations for curriculum discovery. *Advances in Neural Information Processing Systems*, 37:16071–16101, 2024.

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

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

Skipper Seabold, Josef Perktold, et al. Statsmodels: econometric and statistical modeling with python. *SciPy*, 7(1):92–96, 2010.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Antonio Visioli. *Practical PID control*. Springer Science & Business Media, 2006.

David Warde-Farley, Tom Van de Wiele, Tejas Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. In *International Conference on Learning Representations*, 2018.

Haoqi Yuan, Zhancun Mu, Feiyang Xie, and Zongqing Lu. Pre-training goal-based models for sample-efficient reinforcement learning. In *International Conference on Learning Representations*, 2024.

Tianjun Zhang, Benjamin Eysenbach, Ruslan Salakhutdinov, Sergey Levine, and Joseph E Gonzalez. C-planning: An automatic curriculum for learning goal-reaching tasks. In *International Conference on Learning Representations*, 2021.

Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020.

Appendix

A MEGA AND OMEGA DETAILS

Algorithm 2 O/MEGA SELECT functions

function OMEGA_SELECT(env goal g_{ext}, bias b, desired goal distribution p_{dg}, achievable goal distribution $\tilde{p}_{ag}$, $*args$)
  $\alpha \leftarrow 1/\max(b + D_{\text{KL}}(p_{dg} \| \tilde{p}_{ag}), 1)$
  if $x \sim \mathcal{U}(0,1) < \alpha$ **then**
    return g_{ext}
  else
    return MEGA_SELECT($\tilde{p}_{ag}, *args$)
  end if
end function

function MEGA_SELECT(achievable goal distribution $\tilde{p}_{ag}, *args$)
  Sample N candidates $\{g_i\}_{i=1}^{N} \sim \tilde{p}_{ag}$
  Eliminate unachievable candidates
  return $\hat{g} = \arg\min_{g_i} \tilde{p}_{ag}(g_i)$
end function

B THEORETICAL INTUITION BEHIND THE PRACTICAL IMPLEMENTATION OF PE-GCRL

In this section, we attempt to model the evolution of p_{ag} with stochastic processes and provide a rationale for employing an exponentially decaying weighting scheme for historical evaluation data. Although the following content does not constitute a rigorous proof, it offers a reasonable explanation for the validity of our approach.

The achievable goal distribution is a stationary distribution. Let $d_\pi \in \mathbb{R}^n$ be the state visitation distribution of policy π, which can be calculated as: $d_\pi^{\mathrm{T}} = d_\pi^{\mathrm{T}} P_\pi$, where $P_\pi \in \mathbb{R}^{n \times n}$ is the transition probability matrix under the given policy π, with elements defined as $[P_\pi]_{ij} = \Pr(S_{t+1} = s_j | S_t = s_i)$ (assuming all states are indexed as $s_1, s_2, \ldots, s_n$). Then, for any policy π, d_π is a stationary distribution (Sutton & Barto, 2018).

In the goal-augmented Markov Decision Process, ϕ represents a tractable mapping from states to goals. Without loss of generality, assume that all goals can be indexed as $g_1, g_2, \ldots, g_m$. Then, $P_g \in \mathbb{R}^{n \times m}$ is the state-to-goal probability matrix, with elements $[P_g]_{ij} = \Pr(G = g_j | S = s_i)$. Consequently, $p_{ag}^{\mathrm{T}} = d_\pi^{\mathrm{T}} P_g$. From this, it is evident that p_{ag} is also a stationary distribution, which is the primary reason enabling us to predict it.

The evolution of goal-achievement capability can be modeled with an AR(1) model. Let $p_g(t)$ denote the probability that policy π_t can achieve goal g at time t. For a fixed g, the progression of $p_g(t)$ over time forms a stochastic process influenced by the reinforcement learning algorithm. As observed in Fig. 5, the goal-achieving success rate exhibits a relatively stable upward trend during training; let the mean of this trend be denoted as $\mu(t)$. By defining $p'_g(t) = p_g(t) - \mu(t)$, we construct a detrended residual series with a constant mean. We also assume the variance of this series exists. We used *statsmodels* (Seabold et al., 2010) to analyze the stationarity of the $p'_g(t)$ series for the GC-SAC algorithm across four tasks, with results presented in Fig. 10. It can be observed that on the Reach, Push, and Slide tasks, approximately half of the training runs exhibit **weak stationarity** (Ross, 1995) in the $p'_g(t)$ series. For the VVC task, nearly all training runs exhibit weak stationarity. We will analyze the specificity of the results on VVC later.

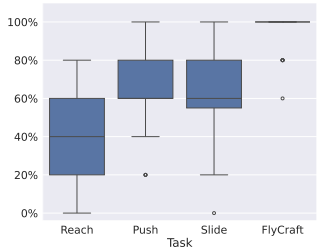


Figure 10: Percentage of experimental runs that exhibit weak stationarity across different tasks. For each task, we trained GC-SAC five times with different random seeds and constructed $p'_g(t) = p_g(t) - \mu(t)$ sequences for each training run. The y-axis indicates how many of the training runs exhibited weak stationarity. To mitigate the influence of goal selection, we evaluated on 30 randomly selected desired goals for each task.

Through the experiments described above, we verified that the detrended achievable goal distribution in GCRL training exhibits a certain degree of weak stationarity. Based on this observation, we posit that the detrended achievable goal distribution can be modeled using a first-order auto-regressive (AR(1)) model (Box et al., 2015):

$$p'_g(t) = \phi p'_g(t-1) + \epsilon_t, \tag{1}$$

where ϕ is the autoregressive coefficient (note that we reuse the symbol ϕ here; it no longer refers to the tractable mapping from states to goals), characterizing the linear influence of $p'_g(t-1)$ on $p'_g(t)$, and ϵ_t is the stochastic disturbance term of the model. This term satisfies the basic assumptions of white noise: $\mathbb{E}(\epsilon_t) = 0$, $\mathrm{Var}(\epsilon_t) = \sigma_\epsilon^2 < \infty$, $\mathrm{Cov}(\epsilon_t, \epsilon_s) = 0 (\forall t \neq s)$, and the disturbance term is independent of past values of the sequence, i.e., $\mathrm{Cov}(\epsilon_t, p'_g(s)) = 0 (\forall s \leq t)$. The autocorrelation function of the AR(1) model is $\rho(k) = \phi^k$ (Box et al., 2015). Since we use data to fit the achievable goal distribution, this first-order linear autoregressive relationship of the achievable goal distribution should be reflected in the data. Therefore, data spaced at an interval k should similarly exhibit a multiplicative relationship of ϕ^k. This explains our rationale for employing an exponentially decaying weighting scheme for historical evaluation data.

Note that the above analysis is specific to the GC-SAC algorithm. The choice of algorithm is a decisive factor influencing the $p'_g(t)$ series; we leave the verification of the weak stationarity of the $p'_g(t)$ series on additional algorithms for future work.

Explanation of Results on VVC. In the case of VVC, we observed that the evolution of its p_{ag} satisfies the weak stationarity assumption well. We attribute this to the fact that in the VVC task, the RL policy learns a guidance law model, while the FlyCraft provides a PID-based (Visioli, 2006) control law model (Gong et al., 2025a). This control law model converts the intermediate commands output by the RL policy into actual control commands, effectively acting as a smoothing mechanism. We hypothesize that this control smoothness facilitates the stable evolution of p_{ag}. In contrast, in the Reach, Push, and Slide tasks, the RL policies learn end-to-end control commands (specifically, we utilize the *joints* mode, as detailed in Table 2) (Gallouédec et al., 2021); consequently, the evolution of p_{ag} in these tasks exhibits some fluctuation compared to VVC.

C Environments Details

- **Reach**: This task requires controlling the robotic arm to move the end-effector from an initial position to a specific target location. Since Reach solely involves the control of the arm's kinematics and basic dynamics, it serves as a fundamental baseline for robotic manipulation.
- **Push**: In this task, the end-effector must push an object to a designated goal. Unlike Reach, this task requires establishing contact with the object and applying pushing forces. Consequently, it involves rigid body collisions and friction, introducing non-linearities into the dynamic equations, which makes Push more difficult than Reach.

Table 2: Environment hyper-parameters.

(a) Reach

Parameter	Value
control_type	joints
reward_type	sparse
goal_range	0.3
distance_threshold	0.02
max_episode_steps	100

(b) Push

Parameter	Value
control_type	joints
reward_type	sparse
goal_xy_range	0.5
obj_xy_range	0.0
distance_threshold	0.05
max_episode_steps	50

(c) Slide

Parameter	Value
control_type	joints
reward_type	sparse
goal_xy_range	0.5
goal_x_offset	0.4
obj_xy_range	0.0
distance_threshold	0.05
max_episode_steps	50

(d) VVC

Parameter	Value
v_min	150
v_max	250
mu_min	-10
mu_max	10
chi_min	-30
chi_max	30
max_episode_steps	400

- **Slide**: This task requires the end-effector to push a puck, allowing it to slide freely to the target. The Slide task is characterized by two distinct phases: a contact phase, where the arm applies force to the object, and a gliding phase, where the arm detaches and the object moves via inertia. Since the physical laws governing these two phases differ, the dynamic model becomes more complex. Furthermore, the consequences of the action are only realized during the gliding phase, forcing the agent to address the credit assignment problem associated with delayed rewards. Due to the complex hybrid dynamics and delayed rewards, Slide is more challenging than Push.

- **Velocity Vector Control** (VVC): This task, conducted on the FlyCraft fixed-wing aircraft simulation engine (Gong et al., 2025a), requires controlling the aircraft to achieve a target velocity vector. Due to the complex non-linear relationships inherent in the aircraft's dynamic equations and the requirement for longer control sequences, the VVC task presents a greater challenge than the robotic manipulation tasks.

The configurations for the Reach, Push, and Slide tasks are presented in Tables 2a, 2b, and 2c, respectively, with all other parameters adhering to the default settings of Panda-Gym (Panda-Gym, 2024). For the VVC task, the settings are detailed in Table 2d, with the remaining parameters set to the defaults of FlyCraft (Gong et al., 2025a).

D Implementation Details

D.1 Calculating p_{ag}

To approximate the ground-truth achievable goal distribution p_{ag}^k for policy π^k at timestep k, we investigated two distinct strategies. The first involved sampling a substantial number M of desired goals from p_{dg} and applying KDE to fit p_{ag}^k using only the subset of successfully achieved goals. The second strategy employed uniform grid sampling over the desired goal space—note that p_{dg} is uniformly distributed within this space for all four tasks—and fitted p_{ag}^k via KDE on the achieved goals. For the former method, we set $M = 1000$ across all four tasks. For the latter, we sampled 4096 desired goals for Reach, 676 for Push and Slide, and 3751 for VVC. Upon comparing the performance of both strategies, we observed no significant difference; consequently, we adopted the second method throughout our experiments.

D.2 CALCULATING $D_{\mathrm{KL}}(p\|q)$

We compared two methods for computing the KL divergence: (1) the rectangular rule, a first-order specialization of Newton-Cotes numerical integration. This method involves discretizing the desired goal space into equally spaced goals $x_0, x_1, \cdots, x_k$ with a uniform interval volume ΔV, yielding the approximation $D_{\mathrm{KL}}(p\|q) \approx \sum_{i=1}^{k} p(x_i) \log \frac{p(x_i)}{q(x_i)} \Delta V$; (2) Monte Carlo estimation, formulated as $D_{\mathrm{KL}}(p\|q) \approx \mathbb{E}_{x\sim p} \log \frac{p(x)}{q(x)}$. For the latter approach, we estimated the expectation by sampling 10^4 points from p. Our experiments revealed no significant difference between the two approaches; consequently, we adopted the Monte Carlo estimation method throughout our experiments.

D.3 SHOWING BEHAVIORAL GOALS ON p_{ag}

Figs. 3a, 7, 23, 30, and 31 illustrate the distribution of sampled behavioral goals on p_{ag} throughout the training progress. We present results specifically for the 10%–30% training stage for two main reasons. First, during the initial training phase, the data in RB for OMEGA-RB exhibits high randomness, while the volume of historical evaluation data for OMEGA-PE is relatively low; consequently, both approaches suffer from large estimation errors at this stage. Second, as we employ the OMEGA algorithm, Fig. 4 indicates that α generally exceeds 0.5 by the midpoint of training. This implies that more than 50% of behavioral goals are sampled from p_{dg} rather than $\tilde{p}_{ag}$, resulting in relatively fewer behavioral goals sampled from $\tilde{p}_{ag}$ during the latter half of training. Based on these considerations, we selected the 10%–30% stage to visualize the distribution of behavioral goals on p_{ag}, and we show only those behavioral goals sampled from $\tilde{p}_{ag}$.

D.4 CALCULATING TRAJECTORY SMOOTHNESS

Mysore et al. (2021) propose a smoothness metric based on the Fast Fourier Transform (FFT) frequency spectrum:

$$\sigma(\tau) = \frac{2}{nf_s} \sum_{i=1}^{n} M_i f_i, \tag{2}$$

where M_i is the amplitude of the i^{th} frequency component, f_i denotes the frequency for $i \in [1, n]$, and f_s is the sampling frequency (note that the maximum resolvable frequency component of the control signal spectrum is the Nyquist frequency, i.e., $\frac{1}{2} f_s$). By jointly considering the frequencies and amplitudes of the control signal components, this metric provides the mean weighted normalized frequency. For a given trajectory, higher values of this metric indicate the presence of high-frequency signal components, which often implies that future states are more difficult to predict.

It is worth noting that the amplitude of a trajectory can also affect the smoothness calculated by Equation 2. Since the numerical ranges of the $D_{\mathrm{KL}}(p_{ag}^{k}\|p_{ag}^{k-d})$ sequence and the $J^k(\pi) - J^{k-d}(\pi)$ sequence differ, we apply Min-Max scaling to normalize the sequences before calculating their smoothness via Equation 2, thereby eliminating the influence of different scales. The Min-Max scaling is calculated as follows: given a sequence $X = \{x_i\}$, the normalized value is defined as $x_i' = \frac{x_i - min(X)}{max(X) - min(X)}$.

D.5 ALGORITHM DETAILS

GC-SAC, HER, OMEGA-RB, and OMEGA-PE are all implemented with the Stable-Baselines3 framework (Raffin et al., 2021). Tables 3 and 4 present the hyperparameters for each algorithm on the Reach, Push, and Slide tasks, and the VVC task, respectively. For both OMEGA-RB and OMEGA-PE, the number of candidate goals is set to $N = 100$, and the balancing factor used to estimate learning progress is fixed at $b = 0$. For the KDE employed to estimate p_{ag}—the default estimator throughout our experiments—we utilized a Gaussian kernel with a bandwidth of 0.2, while other parameters retained the default settings of the scikit-learn framework (Pedregosa et al., 2011). When estimating p_{ag}, OMEGA-RB randomly samples 10^3 transitions from RB and estimates p_{ag} from the achieved goals corresponding to these transitions. In OMEGA-PE, the number of episodes per evaluation is set to $N_e = 96$, and the temperature of the recency weighting of historical evaluation data (Section 3.2) is set to $\tau = 9.491$.

Table 3: Algorithm hyper-parameters on Reach, Push, and Slide.

(a) GC-SAC

Parameter	Value
net_arch	[256,256]
gamma	0.95
train_steps	10^6
batch_size	256
rollout_process_num	1
learning_rate	3×10^{-4}
learning_starts	10^3
gradient_steps	1
buffer_size	2×10^5

(b) HER

Parameter	Value
net_arch	[256,256]
gamma	0.95
train_steps (Push, Slide/Reach)	5×10^5/10^6
batch_size	256
rollout_process_num	1
learning_rate	3×10^{-4}
learning_starts	10^3
gradient_steps	1
buffer_size	2×10^5

(c) OMEGA-RB/OMEGA-PE

Parameter	Value
net_arch	[256,256]
gamma	0.95
train_steps	10^6
batch_size	256
rollout_process_num	1
learning_rate	3×10^{-4}
learning_starts	10^3
gradient_steps	1
buffer_size	2×10^5

The four alternative estimators examined in the density-estimator ablation (Section 4.5) are implemented as follows:

- **GMM.** We use a directly weighted expectation–maximization procedure with 5 full-covariance Gaussian components. The number of initializations is 1, the maximum number of EM iterations is 200, the convergence tolerance is 10^{-3}, and the covariance regularization coefficient is 0.05.

- **NN.** The success-probability classifier is a two-layer MLP with hidden-layer sizes $(16, 16)$, ReLU activations, and the Adam optimizer. The learning rate is 10^{-3}, the L2 coefficient is 10^{-4}, and the maximum number of epochs is 500. We use a 10% validation split for weighted log-loss early stopping, with a minimum of 50 epochs, a patience of 30 epochs, and a tolerance of 10^{-4}. The predicted probabilities are converted into a normalized Gaussian kernel mixture with bandwidth 0.2.

- **NF.** We use a neural spline flow with 2 transforms and 8 bins per transform, with hidden-layer sizes $(16, 16)$. The maximum number of training epochs is 300, the learning rate is 10^{-3}, the weight-decay coefficient is 0, and Gaussian target noise with standard deviation 0.2 is used. We select the training duration by spatial validation using a 15% validation fraction, a minimum of 30 epochs, a patience of 50 epochs, a validation interval of 10 epochs, and a tolerance of 10^{-4}. If validation is unavailable, we use a fallback budget of 50 epochs. All flow models are trained on the CPU.

- **FM.** We train an ensemble of 3 independently initialized flow-matching models and average their densities with equal weights. Each model uses a three-layer MLP with hidden-layer sizes $(32, 32, 32)$, a learning rate of 10^{-3}, 500 training epochs, and 2000 sampled training points per epoch. Gaussian noise with standard deviation 0.1 is added to the target goals. Density evaluation uses fourth-order Runge–Kutta integration with 32 ODE steps and a likelihood batch size of 1024; the weight-decay coefficient is 0.

Table 4: Algorithm Hyper-Parameters on VVC

(a) GC-SAC

Parameter	Value
net_arch	[128,128]
gamma	0.995
train_steps	10^6
batch_size	1024
rollout_process_num	1
learning_rate	3×10^{-4}
learning_starts	10^4
gradient_steps	1
buffer_size	2×10^5

(b) HER

Parameter	Value
net_arch	[128,128]
gamma	0.995
train_steps	10^6
batch_size	1024
rollout_process_num	1
learning_rate	3×10^{-4}
learning_starts	10^4
gradient_steps	1
buffer_size	2×10^5

(c) OMEGA-RB/OMEGA-PE

Parameter	Value
net_arch	[128,128]
gamma	0.995
train_steps	10^6
batch_size	1024
rollout_process_num	1
learning_rate	3×10^{-4}
learning_starts	10^4
gradient_steps	1
buffer_size	2×10^5

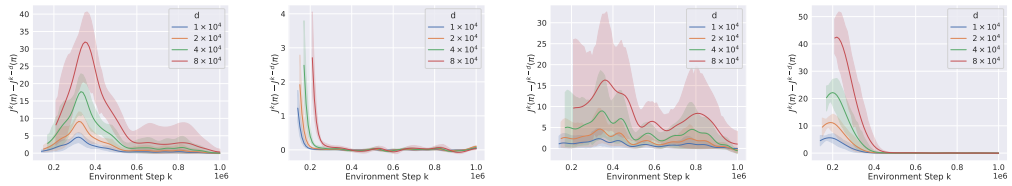
E Additional Results

In this section, we first present all the experimental results of GC-SAC, OMEGA-RB, and OMEGA-PE on the Reach, Push, Slide, and VVC tasks. Subsequently, to provide a more comprehensive validation of our algorithm's effectiveness, we evaluate additional baselines on the Push task:

- **HER**: Building upon GC-SAC, this algorithm incorporates the RELABEL process (Hindsight Experience Replay, (Andrychowicz et al., 2017)), which improves data utilization efficiency.
- **ODISCERN-RB**: Building upon DISCERN (Warde-Farley et al., 2018), which samples behavioral goals b uniformly from the support set of the achievable goal distribution, i.e., $b \sim \mathcal{U}[\mathrm{supp}(\tilde{p}_{ag})]$. ODISCERN adopts the OMEGA algorithm to estimate the learning progress, thereby mixing p_{dg} into the behavioral goal sampling (details are provided in Section 2); its p_{ag} is estimated from the data in the RB.
- **ODISCERN-PE**: It differs from ODISCERN-RB only in that it uses historical evaluation data to estimate p_{ag}.
- **ORIG-RB**: Building upon RIG (Nair et al., 2018), which samples behavioral goals b directly from the achievable goal distribution, i.e., $b \sim \tilde{p}_{ag}$. Similar to ODISCERN, ORIG adopts the OMEGA algorithm to estimate the learning progress and introduces p_{dg} into the behavioral goal sampling, with p_{ag} estimated from the data in the RB.
- **ORIG-PE**: It differs from ORIG-RB only in that it uses historical evaluation data to estimate p_{ag}.

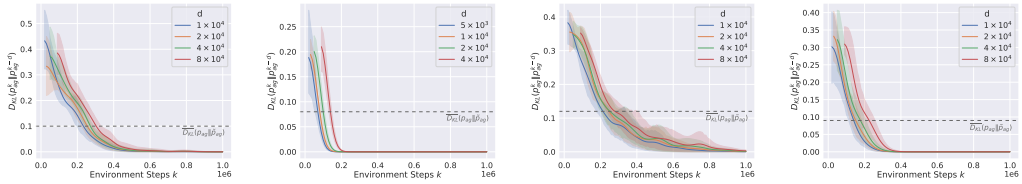
E.1 Additional Motivating Experimental Results

In this section, we present the trends of $J^k(\pi) - J^{k-d}(\pi)$ and $D_{\mathrm{KL}}(p^k_{ag} \| p^{k-d}_{ag})$ throughout the training process for GC-SAC, HER, OMEGA-RB, and OMEGA-PE across the Reach (Fig. 11 and 12), Push (Fig. 13 and 14), Slide (Fig. 15 and 16), and VVC (Fig. 17 and 18) environments. Table 5 reports the smoothness of these curves. The observations are consistent with the findings in Section 3.1.



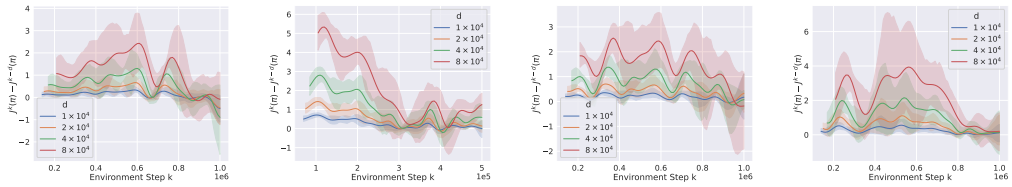
(a) GC-SAC (b) HER (c) OMEGA-RB (d) OMEGA-PE

Figure 11: $J^k(\pi) - J^{k-d}(\pi)$ between policies at different timesteps during training on the Reach task. Results are averaged over five random seeds.



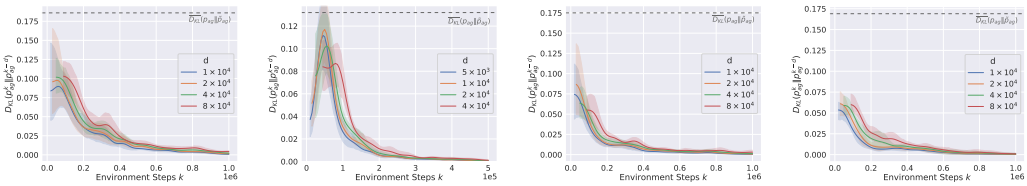
(a) GC-SAC (b) HER (c) OMEGA-RB (d) OMEGA-PE

Figure 12: $D_{\mathrm{KL}}(p_{ag}^k \| p_{ag}^{k-d})$ between policies at different timesteps during training on the Reach task. The dashed line in each subplot represents the average $D_{\mathrm{KL}}(p_{ag} \| \tilde{p}_{ag})$ for the corresponding algorithm throughout training, where $\tilde{p}_{ag}$ is estimated from the RB. Results are averaged over five random seeds.



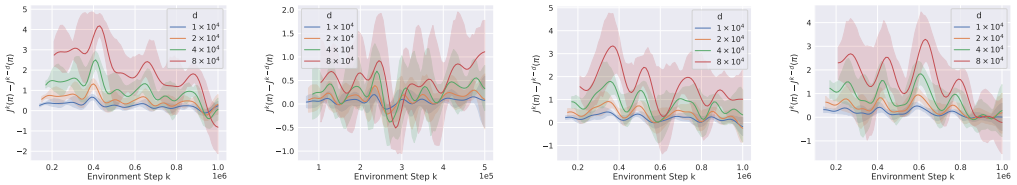
(a) GC-SAC (b) HER (c) OMEGA-RB (d) OMEGA-PE

Figure 13: $J^k(\pi) - J^{k-d}(\pi)$ between policies at different timesteps during training on the Push task. Results are averaged over five random seeds.



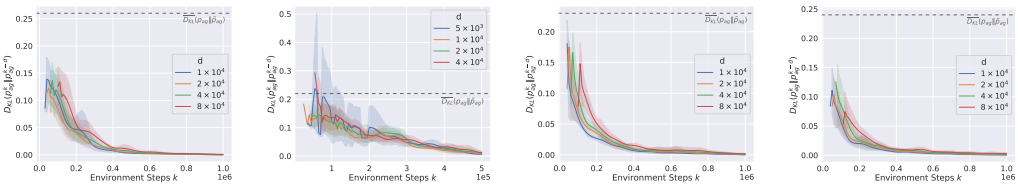
(a) GC-SAC (b) HER (c) OMEGA-RB (d) OMEGA-PE

Figure 14: $D_{\mathrm{KL}}(p_{ag}^k \| p_{ag}^{k-d})$ between policies at different timesteps during training on the Push task. The dashed line in each subplot represents the average $D_{\mathrm{KL}}(p_{ag} \| \tilde{p}_{ag})$ for the corresponding algorithm throughout training, where $\tilde{p}_{ag}$ is estimated from the RB. Results are averaged over five random seeds.



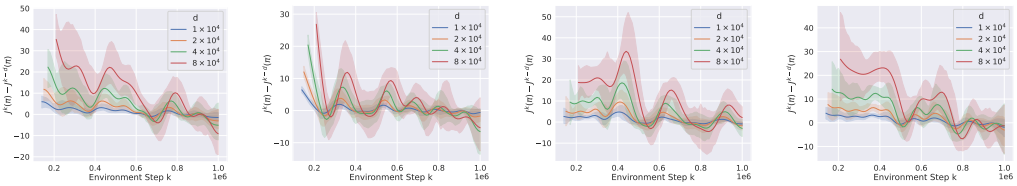
(a) GC-SAC (b) HER (c) OMEGA-RB (d) OMEGA-PE

Figure 15: $J^k(\pi) - J^{k-d}(\pi)$ between policies at different timesteps during training on the Slide task. Results are averaged over five random seeds.



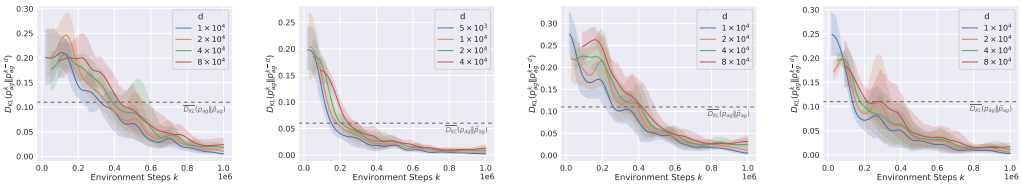
(a) GC-SAC (b) HER (c) OMEGA-RB (d) OMEGA-PE

Figure 16: $D_{\mathrm{KL}}(p_{ag}^k \| p_{ag}^{k-d})$ between policies at different timesteps during training on the Slide task. The dashed line in each subplot represents the average $D_{\mathrm{KL}}(p_{ag} \| \tilde{p}_{ag})$ for the corresponding algorithm throughout training, where $\tilde{p}_{ag}$ is estimated from the RB. Results are averaged over five random seeds.



(a) GC-SAC (b) HER (c) OMEGA-RB (d) OMEGA-PE

Figure 17: $J^k(\pi) - J^{k-d}(\pi)$ between policies at different timesteps during training on the VVC task. Results are averaged over five random seeds.



(a) GC-SAC (b) HER (c) OMEGA-RB (d) OMEGA-PE

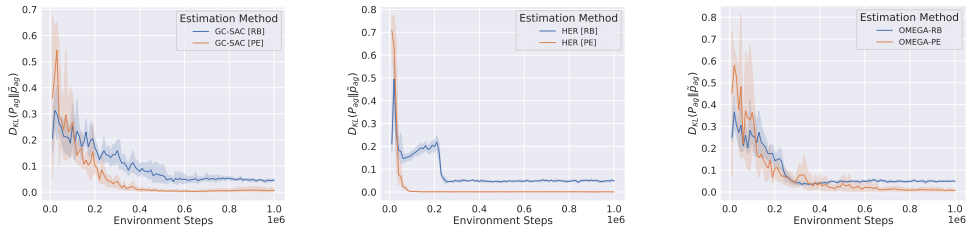
Figure 18: $D_{\mathrm{KL}}(p_{ag}^k \| p_{ag}^{k-d})$ between policies at different timesteps during training on the VVC task. The dashed line in each subplot represents the average $D_{\mathrm{KL}}(p_{ag} \| \tilde{p}_{ag})$ for the corresponding algorithm throughout training, where $\tilde{p}_{ag}$ is estimated from the RB. Results are averaged over five random seeds.

Table 5: Smoothness metric of temporal policy discrepancies in $J(\pi)$ and p_{ag}. Abbreviations: $\sigma(J(\pi))$ stands for $\sigma(J^k(\pi) - J^{k-d}(\pi))$ and $\sigma(p_{ag})$ stands for $\sigma(D_{\mathrm{KL}}(p_{ag}^k \| p_{ag}^{k-d}))$. Results are averaged over five random seeds.

Task \ Algo	GC-SAC		HER		OMEGA-RB		OMEGA-PE	
	$\sigma(J(\pi))$	$\sigma(p_{ag})$	$\sigma(J(\pi))$	$\sigma(p_{ag})$	$\sigma(J(\pi))$	$\sigma(p_{ag})$	$\sigma(J(\pi))$	$\sigma(p_{ag})$
Reach	0.014±0.002	0.011±0.002	0.008±0.001	0.008±0.001	0.014±0.003	0.012±0.002	0.011±0.001	0.010±0.001
Push	0.019±0.002	0.012±0.003	0.019±0.003	0.010±0.001	0.019±0.001	0.009±0.001	0.019±0.002	0.010±0.002
Slide	0.020±0.001	0.010±0.002	0.020±0.000	0.012±0.002	0.020±0.002	0.010±0.001	0.020±0.001	0.010±0.002
VVC	0.014±0.001	0.013±0.001	0.014±0.002	0.010±0.001	0.015±0.002	0.012±0.001	0.017±0.002	0.012±0.002

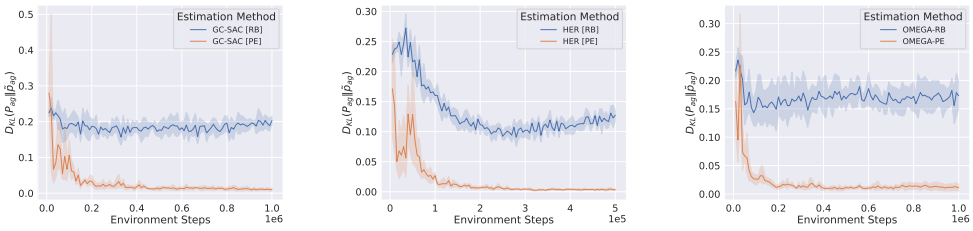
E.2 Additional Main Results

In this section, we compare the estimation error of p_{ag} when using data from RB versus the evaluation data from the PE-GCRL framework. We evaluate this across GC-SAC, HER, and OMEGA on the Reach, Push, Slide, and VVC environments. Figs. 19, 20, 21, and 22 present the results for Reach, Push, Slide, and VVC, respectively. It is evident that, regardless of the task or the algorithm used for training, utilizing the evaluation data from PE-GCRL yields lower estimation errors.



(a) GC-SAC (b) HER (c) OMEGA

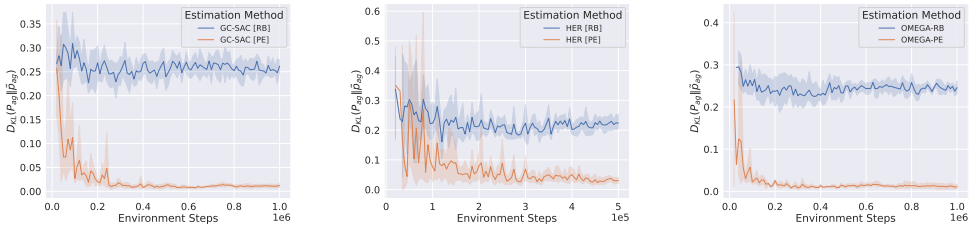
Figure 19: Estimation error of p_{ag} under different estimation methods during the training of various algorithms on Reach. In the left and middle panels, GC-SAC [RB] and HER [RB] refer to estimating p_{ag} from data in the RB, whereas GC-SAC [PE] and HER [PE] refer to the use of historical evaluation data; these estimates are not used to direct behavioral goal selection. In the right panel, OMEGA-RB and OMEGA-PE denote the algorithms trained with each data source, where the estimate drives behavioral goal sampling. Results are averaged over five random seeds.



(a) GC-SAC (b) HER (c) OMEGA

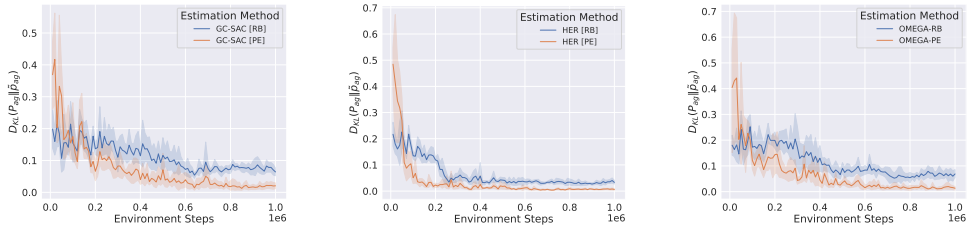
Figure 20: Estimation error of p_{ag} under different estimation methods during the training of various algorithms on Push. In the left and middle panels, GC-SAC [RB] and HER [RB] refer to estimating p_{ag} from data in the RB, whereas GC-SAC [PE] and HER [PE] refer to the use of historical evaluation data; these estimates are not used to direct behavioral goal selection. In the right panel, OMEGA-RB and OMEGA-PE denote the algorithms trained with each data source, where the estimate drives behavioral goal sampling. Results are averaged over five random seeds.

Since the goal spaces of both Reach and VVC exceed two dimensions, effective visualization is precluded. Therefore, we additionally visualize the distribution of sampled behavioral goals on p_{ag} over the course of training for the Slide task, as shown in Fig. 23. It can be observed that the behavioral goals sampled by OMEGA-PE rapidly distribute to the boundary of p_{ag} (i.e., GOIDs) as



(a) GC-SAC (b) HER (c) OMEGA

Figure 21: Estimation error of p_{ag} under different estimation methods during the training of various algorithms on Slide. In the left and middle panels, GC-SAC [RB] and HER [RB] refer to estimating p_{ag} from data in the RB, whereas GC-SAC [PE] and HER [PE] refer to the use of historical evaluation data; these estimates are not used to direct behavioral goal selection. In the right panel, OMEGA-RB and OMEGA-PE denote the algorithms trained with each data source, where the estimate drives behavioral goal sampling. Results are averaged over five random seeds.



(a) GC-SAC (b) HER (c) OMEGA

Figure 22: Estimation error of p_{ag} under different estimation methods during the training of various algorithms on VVC. In the left and middle panels, GC-SAC [RB] and HER [RB] refer to estimating p_{ag} from data in the RB, whereas GC-SAC [PE] and HER [PE] refer to the use of historical evaluation data; these estimates are not used to direct behavioral goal selection. In the right panel, OMEGA-RB and OMEGA-PE denote the algorithms trained with each data source, where the estimate drives behavioral goal sampling. Results are averaged over five random seeds.

training progresses, whereas OMEGA-RB fails to do so. These observations are consistent with the findings in Section 4.3.

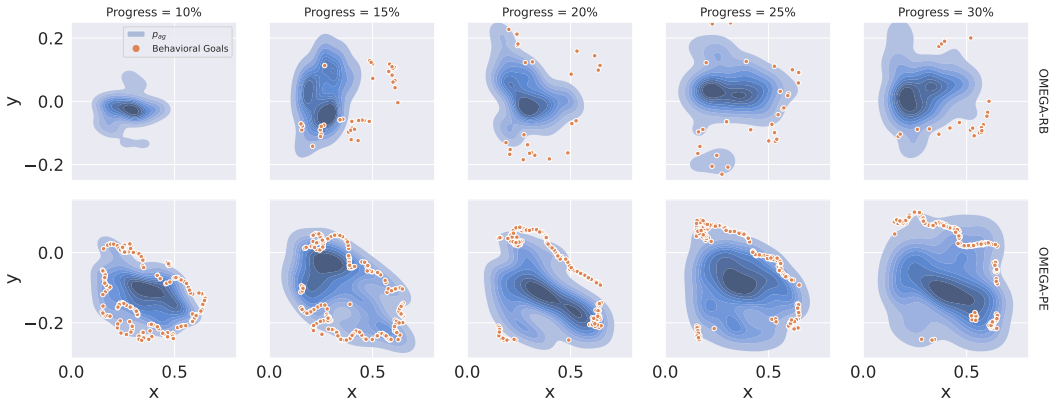
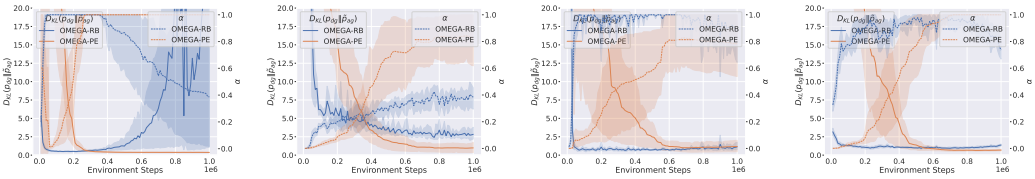


Figure 23: Distribution of sampled behavioral goals on p_{ag} during training on Slide. Columns represent distinct training stages, while rows represent different algorithms. Within each subplot, the x-axis and y-axis correspond to the two dimensions of the Slide task's goal space.

Fig. 24 presents the trends of $D_{\mathrm{KL}}(p_{dg} \| \tilde{p}_{ag})$ and α on all four tasks. Consistent with the observations in Section 4.3, the α of OMEGA-RB deviates from the actual learning progress in a task-dependent manner—for instance, rising prematurely toward 1 on Slide and VVC while stalling at a low level on Push—whereas the α of OMEGA-PE increases gradually as the policy improves.



(a) Reach (b) Push (c) Slide (d) VVC

Figure 24: Same as Fig. 4 in the main text.

E.3 Detailed Analysis of the Training Overhead

Table 6 complements Table 1 with a per-task breakdown of where the training time of OMEGA-PE goes. Training collects 10^6 environment steps on every task, yet the number of episodes these steps fill differs by more than an order of magnitude—about 1.1×10^5 short episodes on Reach versus 4×10^3 long ones on VVC—whereas the evaluations collect about 10^4 episodes on every task. The evaluation data therefore amount to 16%–37% of the training data on Reach, Push, and Slide, but reach 2.36 times the training data on VVC. Even in the latter case the added cost stays minor: evaluation rollout and KDE estimation together occupy only 1.1%–5.8% of the total training time, whereas OPTIMIZE alone occupies 84%–92%—rolling out the current policy for one transition is far cheaper than replaying that transition for repeated policy updates. The evaluation budget is also compressible: the N_e ablation of Section 4.5 shows that even the smallest budget examined, $N_e = 16$, suffices to keep the estimation error low, so the evaluation cost can be trimmed further on tasks where sampling is expensive.

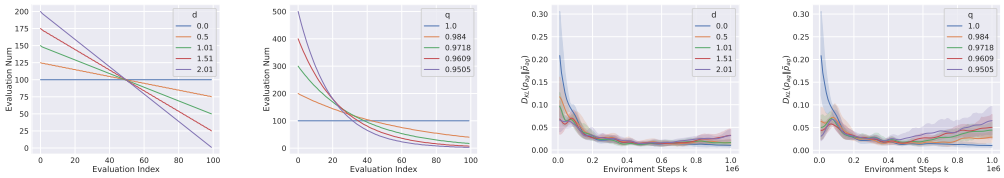
E.4 Decaying Evaluation Schedules

All experiments above interleave evaluations with a constant per-evaluation budget. A natural question is whether the budget should instead be allocated adaptively over training—front-loaded when $\mathcal{B}_e$ is still nearly empty, and tapered as it fills. Fig. 25 examines this on Push with OMEGA-PE: two families of decaying schedules, arithmetic decays of slope d (Fig. 25a) and geometric decays

Table 6: Per-task breakdown of the data volume and wall-clock time of OMEGA-PE training. Training collects 10^6 environment steps on every task; evaluations run in 16 parallel environment instances (a 2–4× speed-up), training in a single one. The time columns separate the rollout (ROLLOUT) and policy-update (OPTIMIZE) stages of TRAIN from the two stages the framework adds: evaluation rollout (EVALUATE) and KDE estimation of $\tilde{p}_{ag}$. Totals differ slightly from Table 1 owing to run-to-run variability. Results are averaged over five random seeds.

Task	Training		Evaluation		Time (s)				
	#Env Steps	#Episodes	#Env Steps	#Episodes	Train Rollout	OPTIMIZE	Eval Rollout	KDE Estimation	Total
Reach	1M	113k	0.16M	9.6k	2544	26107	294	789	29734
Push	1M	25k	0.28M	9.6k	2772	37037	371	87	40267
Slide	1M	51k	0.37M	9.6k	2912	37448	343	443	41146
VVC	1M	4k	2.36M	9.6k	4366	37403	2361	202	44332

of ratio q (Fig. 25b), redistribute the same total budget—100 evaluations and about 10^4 episodes—with $d = 0$ and $q = 1$ corresponding to the constant schedule of 100 episodes per evaluation. Front-loading does remove the early error spike of the constant schedule: the first evaluations each carry more episodes, so $\tilde{p}_{ag}$ rests on more data exactly when $\mathcal{B}_e$ would otherwise be sparse, and the early estimation error drops several-fold (Figs. 25c and 25d). The advantage does not persist: once the front-loaded budget is spent, every subsequent evaluation contributes fewer episodes than under the constant schedule, and the error settles at a higher level—the faster the decay, the higher that level. The two families behave alike, and neither yields a net improvement. This accords with the design of PE-GCRL: the recency-based weighting already discounts stale evaluations at estimation time (Section 3.2), so adaptivity to the evolution of p_{ag} is carried by the estimator rather than by the sampling schedule. The constant schedule is thus a simple and well-balanced choice; we leave more adaptive scheduling strategies to future work.



(a) Arithmetic schedules (d) (b) Geometric schedules (q) (c) Estimation error: arithmetic (d) Estimation error: geometric

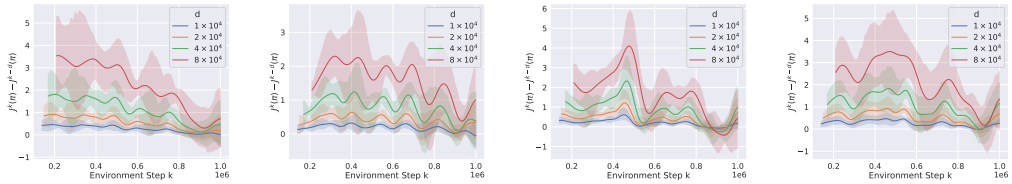
Figure 25: Evaluation schedules with decaying per-evaluation budgets on Push (OMEGA-PE): episodes per evaluation under arithmetic decays of slope d (a) and geometric decays of ratio q (b), and the corresponding estimation error $D_{\mathrm{KL}}(p_{ag}\|\tilde{p}_{ag})$ (c, d). Every schedule performs 100 evaluations with a total budget of about 10^4 episodes; $d = 0$ and $q = 1$ correspond to the constant schedule of 100 episodes per evaluation. Results are averaged over five random seeds.

E.5 EVALUATION ON MORE BASELINES

In this section, we present and analyze the evaluation results of ODISCERN-RB, ODISCERN-PE, ORIG-RB, and ORIG-PE on the Push task.

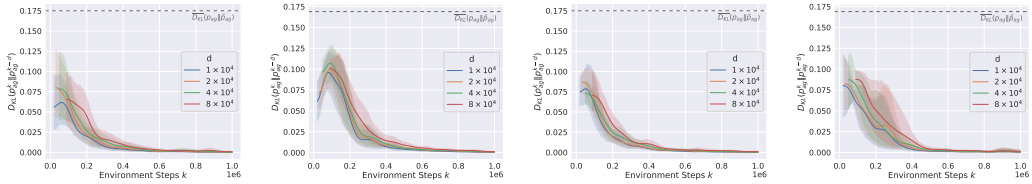
Fig. 26 illustrates the trend of $J^k(\pi) - J^{k-d}(\pi)$ on the Push task, while Fig. 27 depicts the trend of $D_{\mathrm{KL}}(p^k_{ag}\|p^{k-d}_{ag})$. Table 7 presents the smoothness of the aforementioned curves. It can be observed that, across all four algorithms, $D_{\mathrm{KL}}(p^k_{ag}\|p^{k-d}_{ag})$ is smoother than $J^k(\pi) - J^{k-d}(\pi)$. This finding is consistent with the conclusion in Section 3.1.

Fig. 28 illustrates the success rates of the four algorithms on the Push task. It can be observed that for ODISCERN, estimating p_{ag} from RB data yields a higher success rate, whereas for ORIG, using historical evaluation data yields a higher success rate. We subsequently analyze the reasons behind these findings by presenting more detailed results.



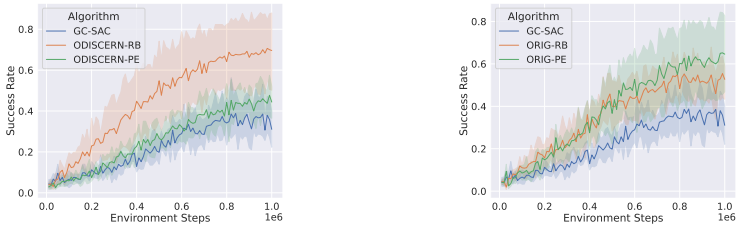
(a) ODISCERN-RB (b) ODISCERN-PE (c) ORIG-RB (d) ORIG-PE

Figure 26: $J^k(\pi) - J^{k-d}(\pi)$ between policies at different timesteps during training on the Push task. Results are averaged over five random seeds.



(a) ODISCERN-RB (b) ODISCERN-PE (c) ORIG-RB (d) ORIG-PE

Figure 27: $D_{\mathrm{KL}}(p^k_{ag} \| p^{k-d}_{ag})$ between policies at different timesteps during training on the Push task. The dashed line in each subplot represents the average value of $D_{\mathrm{KL}}(p_{ag} \| \tilde{p}_{ag})$ for the corresponding algorithm throughout training, where $\tilde{p}_{ag}$ is estimated from the RB. Results are averaged over five random seeds.



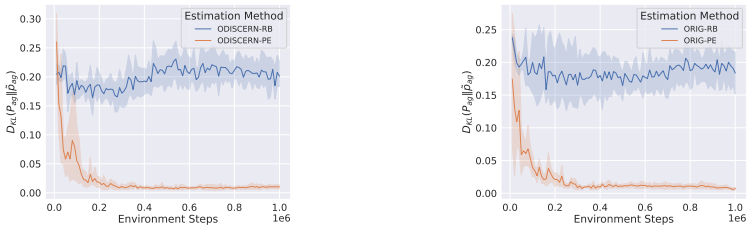
(a) ODISCERN (b) ORIG

Figure 28: Goal achievement success rate during the training of different methods on Push. Results are averaged over five random seeds.

Table 7: Smoothness metric of temporal policy discrepancies in $J(\pi)$ and p_{ag}. Abbreviations: $\sigma(J(\pi))$ stands for $\sigma(J^k(\pi) - J^{k-d}(\pi))$ and $\sigma(p_{ag})$ stands for $\sigma(D_{\text{KL}}(p_{ag}^k \| p_{ag}^{k-d}))$. Results are averaged over five random seeds.

Task \ Algo	ODISCERN-RB		ODISCERN-PE		ORIG-RB		ORIG-PE	
	$\sigma(J(\pi))$	$\sigma(p_{ag})$	$\sigma(J(\pi))$	$\sigma(p_{ag})$	$\sigma(J(\pi))$	$\sigma(p_{ag})$	$\sigma(J(\pi))$	$\sigma(p_{ag})$
Push	0.019±0.002	0.010±0.001	0.019±0.003	0.011±0.001	0.019±0.002	0.010±0.002	0.019±0.001	0.008±0.001

Fig. 29 illustrates the estimation error of p_{ag} when using RB data versus historical evaluation data for ODISCERN and ORIG. It can be observed that, for both algorithms, using historical evaluation data results in a lower estimation error of p_{ag}.



(a) ODISCERN (b) ORIG

Figure 29: Estimation error of p_{ag} during the training of ODISCERN (a) and ORIG (b) on Push. ODISCERN-RB/ORIG-RB estimate $\tilde{p}_{ag}$ from the RB, and ODISCERN-PE/ORIG-PE from historical evaluation data; the estimate drives behavioral goal sampling in each run. Results are averaged over five random seeds.

Fig. 30 illustrates the distribution of behavioral goals during the training of ODISCERN. It can be observed that for both ODISCERN-RB and ODISCERN-PE, the majority of the sampled behavioral goals lie within p_{ag}, with small portions distributed outside. There is no significant difference between the two. We hypothesize that this phenomenon occurs because the uniform sampling strategy over the support set of $\tilde{p}_{ag}$, employed by ODISCERN, has low sensitivity to estimation error of p_{ag}.

Fig. 31 illustrates the distribution of behavioral goals during the training of ORIG. It can be observed that the behavioral goals sampled by ORIG-PE converge more rapidly onto p_{ag}, whereas those sampled by ORIG-RB remain partly distributed outside p_{ag}. This indicates that, in the context of ORIG, using historical evaluation data to estimate p_{ag} more effectively guides the sampling of behavioral goals.

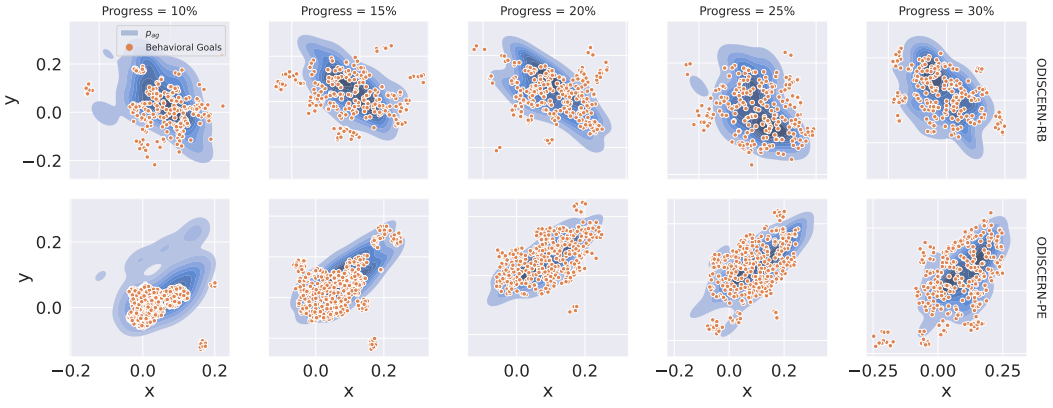


Figure 30: Distribution of sampled behavioral goals on p_{ag} during the training of ODISCERN on Push. Columns represent distinct training stages, while rows represent different algorithms. Within each subplot, the x-axis and y-axis correspond to the two dimensions of the Push task's goal space.

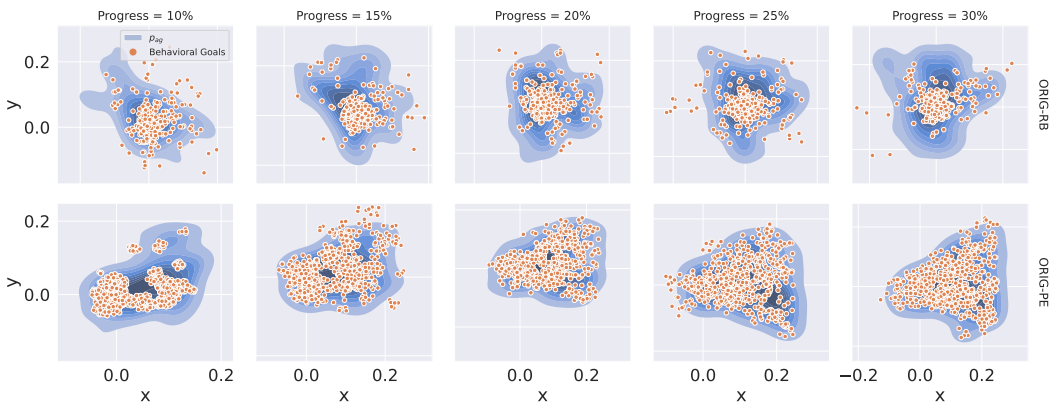
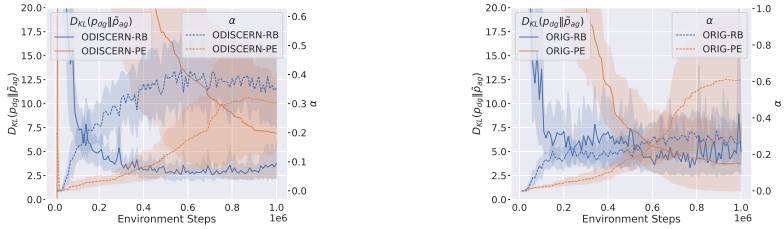


Figure 31: Distribution of sampled behavioral goals on p_{ag} during the training of ORIG on Push. Columns represent distinct training stages, while rows represent different algorithms. Within each subplot, the x-axis and y-axis correspond to the two dimensions of the Push task's goal space.

Fig. 32 illustrates the trends of the learning progress metric α and $D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag})$ within the ODISCERN and ORIG algorithms. By comparing Fig. 32a and Fig. 4a, we observe that in ODISCERN-PE, $D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag})$ declines very slowly, resulting in an insufficient contribution of p_{dg} to behavioral goal sampling during the mid-to-late training stages. Conversely, in ODISCERN-RB, the large estimation error of p_{ag} inadvertently causes a rapid decline in $D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag})$, which in turn leads to a significant contribution of p_{dg} to behavioral goal sampling during the mid-to-late training stages. This may explain why ODISCERN-RB outperforms ODISCERN-PE, which aligns with our finding in the ablation study on b (Section 4.5) that "allowing p_{dg} to dominate behavioral goal sampling earlier helps the policy improve performance more quickly on the Push task." Although ORIG-RB also exhibits a rapid decline in $D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag})$ due to high estimation errors, it fails to sustain this decline in subsequent training. In contrast, ORIG-PE maintains a continuous decrease in $D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag})$, thereby progressively increasing the contribution of p_{dg} to behavioral goal sampling and achieving superior training performance.



(a) ODISCERN (b) ORIG

Figure 32: Trends of goal-achievement discrepancy $D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag})$ and learning progress α during training on Push. The horizontal axis denotes the number of environment steps collected during training. The left vertical axis shows $D_{\mathrm{KL}}(p_{dg}\|\tilde{p}_{ag})$, representing the difference between estimated goal achievement distribution and the desired goal distribution. The right vertical axis indicates the learning progress parameter α (detailed in Section 2). Results are averaged over five random seeds.

In this section, we compare and analyze the training performance of ODISCERN-RB, ODISCERN-PE, ORIG-RB, and ORIG-PE on the Push task. It is evident that, regardless of the algorithm, using historical evaluation data to estimate p_{ag} yields more accurate estimation results. Since the ODISCERN algorithm samples behavioral goals uniformly over the support set of $\tilde{p}_{ag}$, the accuracy of the p_{ag} estimation has a relatively minor impact on its performance. Conversely, as the ORIG algorithm samples behavioral goals directly from $\tilde{p}_{ag}$, it is more sensitive to estimation accuracy; therefore, using historical evaluation data to estimate p_{ag} leads to superior training performance.

F Related Work

F.1 Goal-Conditioned Reinforcement Learning

GCRL aims to learn a single versatile policy capable of achieving diverse goals (Plappert et al., 2018), whereas standard RL typically seeks a unique optimal policy tailored to a specific goal (Sutton & Barto, 2018). This requirement challenges both training efficiency and generalization across goals (Liu et al., 2022), and research has accordingly pursued four directions. **Modeling**: universal value function approximation (Schaul et al., 2015) generalizes the value function to be goal-conditioned, transferring knowledge across goals. **Optimization**: auxiliary or alternative objectives improve training efficiency (Gong et al., 2025b; Eysenbach et al., 2020; Zhang et al., 2021; Ghosh et al., 2020). **Sample utilization**: Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) relabels failed trajectories with their achieved goals to generate positive rewards. **Sample collection**: curricula (Florensa et al., 2018) collect training data on goals of intermediate difficulty for the current policy rather than only the desired goals from the environment, yielding more positive rewards during training.

Our work focuses on sample collection. We investigate how to more accurately estimate the policy's goal-achievement capability—its achievable goal distribution p_{ag}—during training. Our method can be composed with any curriculum method to achieve more accurate sampling of GOIDs. The next two subsections accordingly review curriculum learning and the estimation of p_{ag} in turn.

F.2 Curriculum Learning

In machine learning, curriculum learning refers to a learning paradigm that organizes data or tasks in a meaningful sequence (Bengio et al., 2009). In reinforcement learning, curricula can operate at the sample level (Schaul et al., 2016) as well as at the task level (Florensa et al., 2018; Pitis et al., 2020). In the context of GCRL, a natural framework for implementing curricula is to organize the presentation of behavioral goals to the RL algorithm (Zhang et al., 2020). Several lines of work employ past achieved goals as behavioral goals: Warde-Farley et al. (2018) sample them directly from the RB (a non-parametric model); other methods first fit a distribution to the past achieved goals and then sample behavioral goals from this distribution (Nair et al., 2018; Pong et al., 2020), or from its low-density, sparsely explored regions (Pitis et al., 2020; Florensa et al., 2018). Given the substantial body of evidence that selecting GOIDs for the current policy yields superior performance, we adopt this setting in our experiments. The effectiveness of such curricula, however, ultimately hinges on how accurately the current policy's capability is estimated—the subject of the next subsection.

F.3 Estimating Achievable Goal Distribution

It is important to distinguish the past achieved goal distribution from the achievable goal distribution: the former is the distribution of the goals corresponding to the states visited during training, whereas the latter is the distribution of the desired goals that the current policy can actually achieve; the former serves merely as a proxy for the latter. Existing estimation methods differ mainly in the model used to fit the past achieved goal distribution. Florensa et al. (2018) filter a set of GOIDs according to the cumulative rewards of sampled trajectories and train a GAN over these moderately difficult achieved goals. Pitis et al. (2020) fit the distribution with KDE over the achieved goals stored in the RB and generate GOIDs through a generate-and-test strategy. Pong et al. (2020) fit the distribution with variational autoencoding (Kingma & Welling, 2013) and skew it toward its low-density regions to sample more diverse behavioral goals. Castanet et al. (2023) model the distribution with particles that are attracted to regions of appropriate difficulty via Stein variational gradient descent (Liu & Wang, 2016). All of these methods estimate p_{ag} indirectly from training data and therefore inherit the bias discussed in Section 1: the achieved goals record where past policies happened to visit rather than which goals the current policy can achieve.

A second class of methods estimates the policy's goal-achievement capability directly from evaluation outcomes—that is, the per-goal success probabilities underlying p_{ag}—but they treat historical evaluation data quite differently. The curriculum signals of Rutherford et al. (2024), such as absolute learning progress, are computed exclusively from the outcomes of fresh evaluations of the current policy: historical evaluations receive weight zero and the latest one full weight. This corresponds to the $\tau \to 0^+$ limit of the weighting scheme of PE-GCRL (Section 3.2), where only the latest

data retain non-negligible weight; it is maximally faithful to the current policy, yet every update of the signal requires a fresh batch of environment samples, the efficiency cost discussed in Section 1. In contrast, MAGELLAN (Gaven et al., 2025), which guides autotelic agents through large goal spaces, reuses historical results without recency weighting: it maintains a buffer of goal–outcome pairs collected over the most recent training episodes to train goal-conditioned outcome predictors, and estimates the learning progress of a goal as the absolute difference between the predictions of the current predictor and one saved a fixed number of updates earlier. Neither the outcome pairs within the buffer nor the two predictors are weighted by age—the oldest retained episodes count as much as the newest, and the earlier predictor counts as much as the current one. This corresponds to the $\tau \to +\infty$ limit, where all historical data are reused nearly equally (Section 3.2); it amortizes the evaluation cost but ignores the drift of the capability being estimated.

Compared with these two lines of work, our approach differs in three respects. First, in the data source: the RB-based methods fit the past achieved goal distribution from training data, whereas representing the states visited during training as desired goals does not guarantee that the policy can achieve them; PE-GCRL instead estimates p_{ag} from evaluation data, which align with the definition of p_{ag} by construction (Section 3.2). Second, in the design focus: prior work emphasizes which model and optimization method to use for fitting the distribution, whereas we focus on which data the distribution is fitted to; PE-GCRL prepares such data through periodic evaluation and is orthogonal to the density estimator, so the fitting techniques above can be incorporated. Third, in the treatment of history: existing evaluation-based methods implicitly fix the weighting at one of the two extremes, whereas PE-GCRL weights historical evaluation data by recency with a tunable τ; the empirical analysis of Section 3.1, the AR(1) analysis of Appendix B, and the ablation of Section 4.5 support a moderately large τ that keeps the data fresh while accumulating enough of them. Together, these differences allow PE-GCRL to reconcile the accuracy-efficiency dilemma that existing approaches handle at opposite extremes, reducing the estimation error of p_{ag} by roughly an order of magnitude (Section 4.2).