Which Rewards Matter? Reward Selection for Reinforcement Learning from Limited Feedback

Anonymous authors
Paper under double-blind review

Keywords: Reward Selection, Reinforcement Learning, Learning from Limited Feedback

Summary

The effectiveness of reinforcement learning algorithms is fundamentally determined by the reward feedback they receive during training. However, in practical settings, obtaining large quantities of reward feedback is often infeasible due to computational or financial constraints, particularly when relying on human feedback. When reinforcement learning must proceed with limited feedback—labeling rewards for only a fraction of samples—a fundamental question arises: *which* samples should be labeled to maximize policy performance? We formalize this *reward selection* problem for reinforcement learning from limited feedback (RLLF), introducing a general problem setup to enable the study of different selection strategies. Our investigation proceeds in two parts, evaluating the efficacy of (i) simple heuristics that prioritize high-frequency or high-value states, and (ii) learned selection strategies, trained in advance to identify impactful samples for labeling. These strategies tend to select rewards that (1) guide the agent along optimal trajectories, and (2) support recovery toward near-optimal behavior after deviations. Optimal selection methods yield near-optimal policies with significantly fewer labeled rewards than full supervision, highlighting reward selection as a powerful paradigm for scaling reinforcement learning in feedback-limited settings.

Contribution(s)

1. Formalize the problem of acquiring limited evaluative feedback for reinforcement learning in a general and domain-agnostic way, highlighting its relevance across diverse real-world applications such as RLHF for LLMs and AI-driven drug discovery.
   Context: Existing RL frameworks typically assume access to full reward information during training; our formulation centers the setting where only a limited subset of rewards can be acquired, a regime that remains underexplored.
2. Design and evaluate a range of zero-shot heuristic strategies for reward selection, illustrating how different selection principles influence downstream policy performance.
   Context: In the absence of prior information about impact of rewards, simple strategies like uniform sampling or visitation-based selection offer natural starting points, but their performance has not been systematically explored.
3. Propose training-phase optimization of selection strategies, enabling data-driven approaches to improve reward acquisition decisions prior to evaluation.
   Context: The search space of the reward selection problem is inherently combinatorial, but strategies optimized during the training phase offer tractable and effective approximations to the optimal solution.
4. Analyze the behavior of optimal reward selections and uncover key structural factors—such as reward sparsity and transition structure—that help answer the central question of this work: *which rewards matter?*
   Context: The utility of acquiring rewards at different states depends on factors like transition dynamics and reward sparsity; understanding their effects helps guide more effective reward selection.

Which Rewards Matter? Reward Selection for Reinforcement Learning from Limited Feedback

Anonymous authors
Paper under double-blind review

Abstract

The effectiveness of reinforcement learning algorithms is fundamentally determined by the reward feedback they receive during training. However, in practical settings, obtaining large quantities of reward feedback is often infeasible due to computational or financial constraints, particularly when relying on human feedback. When reinforcement learning must proceed with limited feedback—labeling rewards for only a fraction of samples—a fundamental question arises: *which* samples should be labeled to maximize policy performance? We formalize this *reward selection* problem for reinforcement learning from limited feedback (`RLLF`), introducing a general problem setup to enable the study of different selection strategies. Our investigation proceeds in two parts, evaluating the efficacy of (i) simple heuristics that prioritize high-frequency or high-value states, and (ii) learned selection strategies, trained in advance to identify impactful samples for labeling. These strategies tend to select rewards that (1) guide the agent along optimal trajectories, and (2) support recovery toward near-optimal behavior after deviations. Optimal selection methods yield near-optimal policies with significantly fewer labeled rewards than full supervision, highlighting reward selection as a powerful paradigm for scaling reinforcement learning in feedback-limited settings.

1 Introduction

Various real-world scenarios of sequential decision-making share a striking asymmetry: while data is abundant (or cheaply generated), obtaining evaluative feedback is prohibitively costly and therefore limited by practical constraints. Consider the following examples: in reinforcement learning from human feedback (RLHF) for training large language models (LLMs), billions of tokens can be generated easily, but acquiring reliable human feedback carries significant operational overhead (Christiano et al., 2017; Ouyang et al., 2022; Bai et al., 2022; ABAKA AI, 2025). In the field of AI-driven drug discovery, modern generative models can enumerate billions of syntactically valid molecular graphs in silico, sweeping through an estimated chemical space of $\approx 10^{60}$ drug-like molecules (Reymond, 2015; Gómez-Bombarelli et al., 2018; Jin et al., 2019). Yet confirming that any one of those structures is synthesizable, binds to the intended target, and is non-toxic requires weeks of wet-lab assays and thousands of dollars per compound (DiMasi et al., 2016; Anon, 2023). In these and many similar problems (Appendix A), where evaluative feedback is limited, it becomes critical to identify which subset of the abundant data should be selected for feedback in order to achieve maximal performance gain with minimal feedback.

Reinforcement learning (RL) is the widely adopted approach for solving sequential decision-making problems (Popova et al., 2018; Ouyang et al., 2022; Feng et al., 2023). In the RL framing of the above scenarios, feedback corresponds to rewards, but obtaining rewards for all data points is infeasible. In this work, we study the important question of *reward selection*—which subset of the data should be labeled with rewards to maximize the performance of the learned policy? Acquiring rewards for different subsets leads to policies of varying quality, and the goal is to select the parts

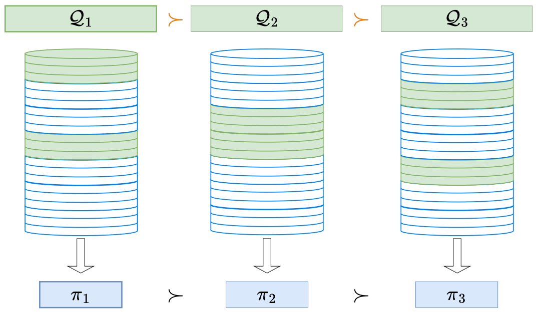

Figure 1: Each row represents a data sample; shaded green rows indicate samples that have been labeled with rewards. The strategy $\mathcal{Q}_i$ determines which states to select for reward labeling. In the limited feedback setup, only a subset of states can be labeled. Different choices of reward-labeled subsets yield learnt policies of varying performances. The objective is to identify the subset that leads to the highest-performing policy.

of the dataset to be reward-labeled such that the resulting policy achieves the highest performance, as illustrated in Figure 1. Thus, reward selection is the problem of determining where to acquire limited feedback under the setting of reinforcement learning from limited feedback (`RLLF`). The question of which data points to acquire rewards for is equivalent to selecting the states at which to observe rewards. Consequently, the problem is formulated as the selection of a subset of states at which to obtain rewards.

To accurately emulate the problem of reward selection, we formulate the setting such that the only degree of freedom allowed is the selection of states (as an input to `RLLF`), and the outcome observed is the resultant policy (as the output of `RLLF`), as illustrated in Figure 2 and detailed in Section 2.2. This abstraction of the details of the `RLLF` mechanism allows us to remain agnostic to specific design choices—particularly the methodology for obtaining rewards—thereby enabling the setup and analysis to generalize to future methods of reward generation. Furthermore, we consider `RLLF` with an offline dataset to disentangle the difficulty of reward selection from that of online exploration. That is, any selection of states can be labeled with rewards, rather than requiring online exploration to encounter states to be labeled. This contrasts with prior setups within active RL (Krueger et al., 2020) and partially observable rewards (Parisi et al., 2024a), which share similar motivations. We adapt aspects of an existing algorithm that incorporates unlabeled data with labeled data for (offline) RL (Yu et al., 2022) as a stand-in algorithm within `RLLF`, in addition to a pessimistic adaptation of Q-learning (in Appendix D.6).

Through extensive experiments across diverse domains, we find that zero-shot heuristic strategies are highly sensitive to domain properties. In environments with deterministic transitions and frequent rewards, selecting states along high-return trajectories leads to strong policy performance. In contrast, stochastic transitions or sparse rewards demand broader coverage of alternative or critical states. Since it is impractical to know which rewards are most impactful without feedback on the effects of the selection, we introduce a training phase where state selection strategies can be evaluated and optimized using aggregate feedback on the performance of resulting policies. We show that optimal state sets—identified using training-phase optimization of selection strategies—can yield near-optimal policies with far fewer labeled rewards than full supervision, underscoring the potential of feedback-efficient learning. In this work, **our contributions** are:

1. Formalize the problem of acquiring limited evaluative feedback for reinforcement learning in a general and domain-agnostic way, highlighting its relevance across diverse real-world applications enumerated in Appendix A, such as RLHF for LLMs and AI-driven drug discovery.
2. Design and evaluate a range of zero-shot heuristic strategies for reward selection, illustrating how different selection principles influence downstream policy performance.
3. Propose training-phase optimization of selection strategies, enabling data-driven approaches to improve reward acquisition decisions prior to evaluation.
4. Analyze the behavior of optimal reward selections and uncover key structural factors—such as reward sparsity and transition structure—that help answer the central question of this work: *which rewards matter?*

2 Problem Formulation and Preliminaries

Preliminaries: An MDP is a tuple $M := (\mathcal{S}, \mathcal{A}, p, r, \gamma, \eta)$ where $\mathcal{S}$ is a finite set of states, S_t is the state at time $t \in \{0, 1, \dots\}$, $\mathcal{A}$ is a finite set of actions, A_t is the action at time t, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the *transition function* that characterizes state transition dynamics according to $p(s, a, s') := \Pr(S_{t+1}{=}s' | S_t{=}s, A_t{=}a)$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the *reward function* that characterizes rewards according to $r(s, a) := \mathbb{E}[R_t | S_t{=}s, A_t{=}a]$, $\gamma \in [0, 1]$ is the reward discount parameter, and $\eta : \mathcal{S} \to [0, 1]$ characterizes the initial state distribution according to $\eta(s) := \Pr(S_0{=}s)$. A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ characterizes how actions can be selected given the current state according to $\pi(s, a) := \Pr(A_t{=}a | S_t{=}s)$. We consider finite horizon MDPs (Sutton and Barto, 2018) where episodes terminate by some (unspecified) time $T \in \mathbb{N}$.

2.1 Reinforcement Learning from Limited Feedback

We study the problem of reinforcement learning from limited feedback (RLLF) in the offline setting. An offline dataset $\mathcal{D}_n = \{(S_t, A_t, S_{t+1})^{(i)}\}_{i=1}^{n}$ of n samples is obtained by the interaction of a *data-collecting policy* π_D with M.[1] The dataset contains no reward, i.e., evaluative feedback. To emulate the limited feedback setting, the restriction imposed by the problem setup is that environment rewards are permitted to be obtained at only a subset B of the states. Let $\mathcal{S}_{[B]}$ denote the states that are reward-labeled. For samples in $\mathcal{D}$ where $S_t \in \mathcal{S}_{[B]}$, reward labels are assigned; the remaining samples in $\mathcal{D}$ are unlabeled. In practice, since the *labeling budget* is smaller than the total number of states $|\mathcal{S}|$, only a subset of the dataset can be reward-labeled. The process of reward-labeling part of the dataset and learning a policy from the resulting partially labeled data is referred to as reinforcement learning from limited feedback, and is denoted by $\texttt{RLLF}(\mathcal{D}, \mathcal{S}_{[B]})$ (see the box in Figure 2). Different choices of $\mathcal{S}_{[B]}$ result in different policies learned from the partially labeled dataset, with varying performance (see Figure 5 in Appendix D.2).

Rather than passively learning a policy from a given partially labeled dataset, we study the problem of actively selecting the states to label with rewards in order to obtain the best-performing policy. Formally, the **reward selection** problem is to *identify a subset of states $\mathcal{S}_{[B]}$, subject to a labeling budget B, to be labeled with rewards such that the policy learned from the resulting partially labeled dataset achieves maximum performance*.

Policy Learning from Partially Reward-Labeled Data: The problem setup involves learning a policy from partially reward-labeled data. We use the UDS algorithm by Yu et al. (2022) to learn a policy from the partially reward-labeled dataset. This algorithm follows a simple procedure: unknown rewards are replaced with zero (or $R_{\min}$), and a policy is learned using these imputed rewards. We adopt Q-learning as the policy update rule, as is standard in offline RL settings (Levine et al., 2020; Kostrikov et al., 2021). Other methods for handling partially labeled data could also be employed, but the focus of this work is on identifying a reward selection strategy that is effective for this instantiation of RLLF. An alternative policy learning rule—specifically, a pessimistic adaptation of Q-learning—is also studied in Appendix D.6.

2.2 Reward Selection

The strategy for selecting the B states from $\mathcal{D}$ to label with rewards is denoted by $\mathcal{Q}^{(B)} : \mathcal{D} \to \mathcal{S}^B$. Formally, given a budget B, the set of states at which rewards are observed is defined as $\mathcal{S}_{[B]} = \mathcal{Q}^{(B)}(\mathcal{D})$. The resulting policy is denoted by $\pi_{[B]} = \texttt{RLLF}\left(\mathcal{D}, \mathcal{Q}^{(B)}(\mathcal{D})\right)$.[2] The effectiveness of a strategy $\mathcal{Q}^{(B)}$ is quantified by the expected return of the policy produced by RLLF when trained using the rewards selected by $\mathcal{Q}^{(B)}$. The objective, denoted by $P(\cdot)$, is to maximize the *average* expected return of the resulting policy, $J(\pi) := \mathbb{E}_\pi\left[\sum_{t=0}^{T} \gamma^t R_t\right]$, averaged over possible datasets

[1] The data-collecting policy π_D can be a single policy, or a mixture of policies of which the weighted average is denoted by π_D. For clarity, we drop the subscript n unless explicitly needed, and denote the dataset by $\mathcal{D}$.

[2] To make the dependence on $\mathcal{S}_{[B]}$ explicit, $\pi_{[B]}$ is equivalently denoted as $\pi_{[\mathcal{S}_{[B]}]}$ when relevant.

$\mathcal{D}$. That is,

$$\max_{\mathcal{Q}^{(B)}} P(\mathcal{Q}^{(B)}) := \max_{\mathcal{Q}^{(B)}} \mathbb{E}_{\mathcal{D}}\left[J\left(\pi_{[B]}\right)\right] = \max_{\mathcal{Q}^{(B)}} \mathbb{E}_{\mathcal{D}}\left[J\left(\texttt{RLLF}\left(\mathcal{D}, \mathcal{Q}^{(B)}(\mathcal{D})\right)\right)\right]. \quad (1)$$

When $\mathcal{Q}^{(B)}$ is stochastic, the definition of $P(\cdot)$ includes an additional nested expectation over $\mathcal{Q}^{(B)}$.

Optimality: Given a budget B, the *optimal* reward selection strategy $\mathcal{Q}^{(B)}$ maximizes the performance of the resultant policy $\pi_{[\mathcal{S}_{[B]}]}$. There are $\binom{|\mathcal{S}|}{B}$ candidate state sets that may be chosen by $\mathcal{Q}^{(B)}$, all resulting in varying policies with varying performances (Appendix D.2). The optimal strategy entails selecting a state set, denoted by $\mathcal{S}^*_{[B]}$, that results in a policy with the highest performance, i.e.,

$$\mathcal{S}^*_{[B]} = \arg \max_{\mathcal{S}_{[B]} \subseteq \mathcal{S}, |\mathcal{S}_{[B]}|=B} P(\pi_{[\mathcal{S}_{[B]}]}) = \arg \max_{\mathcal{Q}^{(B)}(\mathcal{D}) \subseteq \mathcal{S}} P(\texttt{RLLF}(\mathcal{D}, \mathcal{Q}^{(B)}(\mathcal{D}))). \quad (2)$$

It must be noted that $\mathcal{S}^*_{[B]}$ may not be a unique set, rather, it belongs to a set of *equally optimal state sets*. For ease of exposition, we pick one such state set. The efficacy of any other strategy, that selects a different state set $\mathcal{S}_{[B]}$, can be quantified by the *optimality gap*, i.e., the gap from the performance of the optimal policy under the labeling budget $\pi^*_{[B]} = \pi^*_{[\mathcal{S}^*_{[B]}]}$, given by:

$$\text{OptimalityGap}(\mathcal{S}_{[B]}) = P(\pi^*_{[B]}) - P(\pi_{[\mathcal{S}_{[B]}]}). \quad (3)$$

Without any insight into how selecting specific states affects the final performance of the policy, it is challenging to design effective reward selection strategies. To enable more informed strategy design, we introduce an optional *training phase* in which the reward selection strategy learner $\mathcal{Q}^{(B)}$ may leverage feedback from an evaluator Ξ. This evaluator provides the expected return of any given policy under the true reward function of M. In practice, for example, this could correspond to the online deployment of a policy (trained on limited feedback) to assess its performance. This performance can then serve as a signal to refine the reward selection strategy, in turn improving the policy performance. Once the reward selection strategy is trained, it is evaluated in a *test phase*, where access to the evaluator is no longer available. This setup is illustrated in Figure 2.

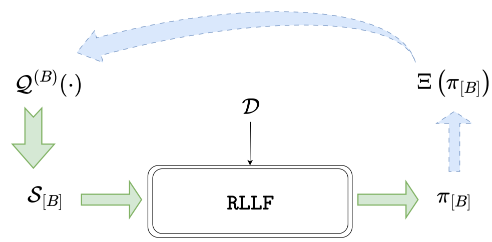

Figure 2: Problem setup for reward selection: The green arrows indicate the test phase, during which the reward selection strategy is evaluated. The blue arrows represent access to, and feedback from, an evaluator available within the training phase loop.

During the **training phase**, the evaluator Ξ helps assess the performance of different selections of state subsets. It is important to note that Ξ provides only aggregate performance evaluations of policies—individual rewards obtained during evaluation are neither stored nor reused. The `RLLF` procedure is treated as a black box: individual state-reward values and the specific policy update mechanisms remain inaccessible to the designer of the reward selection strategy. `RLLF` simply outputs a policy given a state (or set of states) and the unlabeled dataset as input; the resulting policy may optionally be evaluated using Ξ to guide subsequent updates to the selection strategy.

During the **test phase**, trained reward selection strategies are evaluated along two dimensions: (1) their performance, as defined in Equation 1, and (2) their training cost, measured by the number of calls made to the evaluator Ξ during the training phase. A strategy that maximizes test performance while minimizing evaluator usage during training is preferred. For strategies that undergo training, the training dataset $\mathcal{D}_{\text{train}}$ is generated using a data-collecting policy π_{train}, while the test datasets $\mathcal{D}_{\text{test}}$ are generated using a set of data-collecting policies $\Pi_{\text{test}} = \{\pi_1, \pi_2, \ldots, \pi_m\}$. Empirically, test performance is computed by averaging over $\mathcal{D}_{\text{test}}$, as in Equation 1.

3 Methodology: Selection Strategies

We study two types of selection strategies. The first category consists of strategies guided by intuitive heuristics that are rule-based and do not rely on the training phase. Thus, they can be expected to perform well enough, though not optimally. The analysis serves to assess the utility of intuitive heuristics when applied to the problem of reward selection without access to any prior information. The second category includes strategies that incorporate a training phase prior to evaluation. Within this category, we study a spectrum of approaches: from strategies that identify the optimal reward-labeled state set $\mathcal{S}^*_{[B]}$, albeit at high training cost, to approximate strategies that reduce training overhead at the expense of marginal loss in performance. Additionally, the strategies we study can be classified based on how they construct the reward-labeled state set: *batch strategies*, which select all B states at once, and *iterative strategies*, which select one state at a time over B iterations. Iterative strategies are indexed by $b \in 1, \ldots, B$, with selected states and related quantities indexed by b, for instance the set of selected states $\mathcal{S}_{[b]}$. Appendix C provides a detailed categorization.

3.1 Heuristic-Based Selection: Training-Free Strategies

Given an offline dataset $\mathcal{D}$, without any prior feedback to inform how labeling different states with rewards impacts the performance of the policy, we must rely on heuristics to guide our selection of states to label with rewards. The state-visitation distribution of the data collecting policy π_D, captured within the offline test dataset, serves as a useful source of information to guide the selection of states for reward-labeling. Additionally, constructing the state set (of size B) iteratively, i.e., adding one state at a time, allows us to leverage the intermediate updates to the policy and its corresponding Q-function as guidance to inform subsequent selections. The heuristics investigated are:

(1) `visitation` sampling: This strategy encodes the intuitive notion that maximizing the fraction of the dataset that is reward-labeled is a good proxy for maximizing the expected return of the resultant policy. To do so, it samples the most commonly occurring states in the dataset without replacement from the state-visitation distribution d^{π_D}, i.e., $\mathcal{S}_{[B]} \sim \text{Sample}_{\text{w/o rep}}(\mathcal{S}, d^{\pi_D}, B)$.

   (a) If $\mathcal{S}_{[B]}$ is constructed in an *iterative* manner, i.e., adding one state at a time, as opposed to a *batch* manner as above, an additional *on-policy* variant of this strategy is studied, referred to as `visitation-on-policy`, where the state set $\mathcal{S}_{[B]}$ is constructed by sampling states from the state-visitation distribution of the updated policy $\pi_{[b-1]}$ at each iteration b.

(2) `uniform` sampling: This simple strategy samples B states without replacement from a uniform distribution over all unlabeled states, i.e., $\mathcal{S}_{[B]} \sim \text{UniformSample}_{\text{w/o rep}}(\mathcal{S}, B)$. Along with serving as a baseline for comparison with other strategies, this simple strategy turns out to be surprisingly effective in certain cases where states that are not frequently visited under π_D can have high utility when labeled with rewards.

(3) `guided` sampling : This is an iterative strategy that balances exploration and exploitation—by exploring via sampling from the state-visitation distribution, and exploiting by sampling from the neighborhood of the current highest valued state. Specifically, at each iteration b, the strategy samples from the distribution q_b defined as:

$$q_b(\cdot|Q^{\pi_{[b-1]}}, b) \propto \alpha_b \underbrace{d^{\pi_D}(\cdot)}_{\text{explore}} + (1-\alpha_b) \underbrace{d^{\pi_D}_{\text{prev}}(\cdot \mid \arg\max_{s\in\mathcal{S}} \max_{a\in\mathcal{A}} Q^{\pi_{[b-1]}}(s,a))}_{\text{exploit: focus on areas near the most promising Q-values}}$$

where $\widehat{d}^{\pi_D}_{\text{prev}}(\cdot \mid s')$ is the sample estimate of the distribution of states that lead to state s' as the next state under π_D. The term $\arg\max_{s\in\mathcal{S}_{[b-1]}} \max_{a\in\mathcal{A}} Q^{\pi_{[b-1]}}(s,a)$ identifies the state with the maximum (state-)value based on the rewards obtained thus far. The tradeoff weight α_b initially places more weight on the exploratory term and then decays as b increases, with decreasing α_b as Q-values become more reliable.

(a) The on-policy variant of this strategy, `guided-on-policy`, is also studied.

We estimate the state visitation distribution(s) $d^{\pi_D}(\cdot)$ from the dataset $\mathcal{D}$, denoted by $\widehat{d}^{\pi_D}(\cdot)$, as $\widehat{d}^{\pi_D}(s) := \frac{N(s)}{\sum_{s' \in \mathcal{S}} N(s')}$, where $N(s)$ denotes the number of occurrences of state s in $\mathcal{D}$. These strategies are empirically evaluated in Section 4 and compared to the training-based strategies described in the next section.

3.2 Strategies Leveraging the Training Phase

For the set of strategies that leverage the training phase, the feedback from the evaluator provides a key insight: the impact of the selected states on the performance of the resultant policy, and, consequently, the overall strategy performance as in Equation 1. The set of states selected can therefore be modified to improve the performance of the resultant policy. The cost of this training phase, prior to the strategy's evaluation, is quantified by the number of calls to the evaluator Ξ.

(1) The most straightforward strategy is to exhaustively search over all possible subsets of B states during the training phase, and select the one that results in the highest performing policy. This approach, referred to as `brute-force`, is guaranteed to find the optimal state set $\mathcal{S}^*_{[B]}$, *given sufficient coverage of the training data*. However, it makes a prohibitive number of calls to the evaluator (training cost) on the order of $O(|\mathcal{S}|^B)$—since the search space is *combinatorially* large: $\binom{|\mathcal{S}|}{B} \approx O(|\mathcal{S}|^{\min\{|\mathcal{S}|-B,B\}}) \approx O(|\mathcal{S}|^B)$—which is impractical for any reasonably sized state space $\mathcal{S}$.

(2) To mitigate the training cost, we investigate an iterative strategy that constructs the state set $\mathcal{S}_{[B]}$ one state at a time. Specifically, define the *utility* of adding s to $\mathcal{S}_{[b]}$ as

$$\Delta(s|\mathcal{S}_{[b]}) := P(\pi_{[\mathcal{S}_{[b]} \cup \{s\}]}) - P(\pi_{[\mathcal{S}_{[b]}]}). \tag{4}$$

The `sequential-greedy` strategy selects the state s that maximizes $\Delta(s|\mathcal{S}_{[b]})$, i.e., the marginal utility of adding state s to the current set of states $\mathcal{S}_{[b]}$ at each iteration b. As a result, this strategy has a training cost of $O(B|\mathcal{S}|)$, significantly lower than the brute force strategy. Furthermore, we empirically observe that the sequential-greedy strategy is approximately optimal in many cases.

(3) Lastly, instead of relying on a rule-based approach, we optimize the selection strategy $\mathcal{Q}^{(B)}$ using an evolutionary strategy (ES) (Rechenberg, 1989; Salimans et al., 2017). We parameterize the selection strategy $\mathcal{Q}^{(B)}$ with parameters θ, i.e., $\mathcal{Q}^{(B)}_\theta$. We define the fitness of each state set $\mathcal{S}_{[b]}$ as the performance of the resulting policy $J(\pi_{[\mathcal{S}_b]})$, and run a few iterations of ES to optimize θ. The number of samples k in each iteration, and the number of iterations m, determine the overall training complexity $O(km)$ of this strategy, referred to as `ES`.

A categorization of all selection strategies is provided in Appendix C.1.

4 Empirical Analysis

This section evaluates reward selection strategies across diverse domains. Rather than proposing new heuristics, we study key factors that influence effective selection. We assess heuristic and optimal selection methods and analyze how environment characteristics shape reward acquisition outcomes.

Domain: We evaluate performance across six small-scale domains and four large-scale MinAtar domains (Young and Tian, 2019) (Breakout, Freeway, Seaquest, Asterix). Some small domains (Graph, Tree, TwoRooms, TwoRooms-Trap) are hand-designed; others (FrozenLake, CliffWalk) are Gymnasium benchmarks (Brockman et al., 2016; Foundation, 2023). Additional domain details, transition dynamics, reward structures, expert policies, data collection, and full results for TwoRooms-Trap and FrozenLake appear in Appendix D.1.

Evaluation: The primary evaluation metric is the **average episode return**, reported across all experiments. For heuristic selection results, we additionally report the **optimality gap**, as defined in Equation 3. All reward acquisition budgets are expressed as percentage feedback relative to the total number of unique states $|S|$ in each dataset (Table 1, Figure 3, and Figure 4), allowing for consistent comparison across domains. After a selection strategy chooses a set of states, we train a policy using UDS and evaluate it; an alternative training algorithm and analysis are provided in Appendix D.6.

4.1 Heuristic Reward Selection Performance Varies Across Domains

We show the absolute return and optimality gap for the three reward selection strategies introduced in Section 3.1—`guided`, `visitation`, and `uniform`—on selected small-scale domains (Table 1; full results for all small-scale domains are provided in Appendix D.3) and large-scale domains (Figure 3). Since these strategies do not involve a training phase, they are evaluated directly on the test dataset. Results for small-scale domains are averaged over 100 random seeds, while results for large-scale domains are averaged over 10 random seeds.

We observe that the effectiveness of reward selection heuristics is highly domain-dependent , and no single strategy consistently dominates. Section 4.3 provides a more detailed analysis of when each heuristic tends to perform well. Below, we highlight several key empirical findings:

1. **Performance under low budgets:** When the reward labeling budget is small, the learned Q-values are typically inaccurate and unstable. In such cases, using visitation-based heuristics often provides a more reliable signal for state selection. For example, in Graph, the off-policy visitation distribution induced by the data-collecting policy aligns well with the optimal path early on, leading to improved performance. In CliffWalk, however, it is more effective to use the on-policy visitation distribution, as shown in Table 1.
2. **Performance under high budgets:** As the budget increases, the learned Q-function becomes more accurate and informative. In this setting, heuristics that rely more directly on the estimated Q-values—such as `guided`—tend to perform better. This trend is clearly observed in Graph, Tree, CliffWalk, TwoRooms-Trap, and all four MinAtar domains.
3. **Impact of bottleneck structures:** In domains with bottleneck states—states that must be traversed to reach certain regions of the environment, such as TwoRooms and FrozenLake—visitation-based heuristics may underperform. Although these heuristics prioritize high-value regions, their reliance on visitation frequency under the data-collecting policy π_D can lead them to overlook infrequently visited but critical states. In such cases, the `uniform` heuristic can sometimes outperform both `guided` and `visitation` by providing broader and more stable coverage across the entire state space, including areas rarely visited by π_D but essential for task completion.
4. **Optimality gap trends:** Across all domains, we observe that the optimality gap is large when the budget is small and gradually decreases as the budget increases. This reflects the inherent challenge of selecting the most informative states under tight budget constraints, and the improvement in the quality of the learned policy as more targeted reward feedback becomes available.

4.2 Training-Optimized Selection Strategies Achieve Near-Optimal Performance

This section evaluates the quality of optimal state sets identified by the search algorithms introduced in Section 3.2: `brute-force`, `sequential-greedy`, and `ES`. Comparisons are conducted on both training and testing datasets to assess the robustness of the identified sets to variations in the datasets collected by different data-collecting policies.

Due to its extreme training computational cost, `brute-force` search is only applied to small-scale domains. For example, in a domain with $|S| = 50$, exhaustively evaluating all possible state sets of size $B = 25$ would require $\binom{50}{25} \approx 10^{14}$ calls to the evaluator. Even at a rate of 2,000 calls to the evaluator per minute, completing this search would still take about one million years.

Table 1: Comparison of `guided`, `visitation`, and `uniform` heuristic selection strategies on small-scale domains. For each domain, the table presents the mean policy return (± standard error) and the corresponding optimality gap (in parentheses) across five feedback levels.

Domains	Percentage Feedback	guided	guided-on-policy	visitation	visitation-on-policy	uniform
Graph	0.1	3.701 ± 0.129 (3.302)	3.208 ± 0.139 (3.795)	**3.797 ± 0.151 (3.206)**	3.300 ± 0.142 (3.703)	2.949 ± 0.137 (4.054)
	0.3	5.831 ± 0.137 (2.169)	5.760 ± 0.127 (2.240)	**5.871 ± 0.146 (2.129)**	5.690 ± 0.146 (2.310)	4.617 ± 0.156 (3.383)
	0.5	7.110 ± 0.099 (0.890)	**7.690 ± 0.070 (0.310)**	7.199 ± 0.090 (0.801)	7.583 ± 0.086 (0.417)	5.978 ± 0.114 (2.022)
	0.7	7.830 ± 0.040 (0.170)	**8.000 ± 0.000 (0.000)**	7.599 ± 0.060 (0.401)	7.991 ± 0.009 (0.009)	6.920 ± 0.084 (1.080)
	0.9	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**
Tree	0.1	**8.003 ± 0.468 (9.053)**	7.403 ± 0.869 (9.653)	6.133 ± 0.428 (10.924)	4.658 ± 0.370 (12.398)	5.665 ± 0.532 (11.392)
	0.3	**12.846 ± 0.373 (4.921)**	12.755 ± 0.632 (5.013)	11.763 ± 0.427 (6.004)	12.601 ± 0.414 (5.167)	10.341 ± 0.524 (7.427)
	0.5	16.072 ± 0.205 (1.695)	**16.415 ± 0.207 (1.352)**	15.395 ± 0.297 (2.372)	16.379 ± 0.216 (1.388)	13.218 ± 0.430 (4.550)
	0.7	17.193 ± 0.083 (0.575)	**17.444 ± 0.037 (0.323)**	17.135 ± 0.153 (0.633)	17.174 ± 0.120 (0.594)	15.258 ± 0.312 (2.509)
	0.9	17.673 ± 0.013 (0.094)	**17.731 ± 0.031 (0.036)**	17.609 ± 0.158 (0.049)	17.521 ± 0.110 (0.246)	17.695 ± 0.141 (0.072)
CliffWalk	0.1	-1248.872 ± 117.272 (1152.914)	-616.760 ± 105.578 (520.803)	-1262.067 ± 119.207 (1166.109)	**-392.040 ± 84.184 (296.082)**	-1156.960 ± 61.025 (1061.002)
	0.3	-369.964 ± 86.539 (285.948)	-93.637 ± 0.373 (9.621)	-462.530 ± 100.633 (378.515)	**-92.981 ± 0.358 (8.965)**	-1274.561 ± 118.910 (1190.545)
	0.5	-132.671 ± 32.819 (57.629)	-89.390 ± 0.827 (14.348)	-165.870 ± 46.201 (90.828)	**-86.746 ± 0.677 (11.704)**	-1235.823 ± 137.366 (1160.781)
	0.7	-98.870 ± 0.647 (32.665)	-74.821 ± 2.171 (8.615)	-100.000 ± 0.000 (33.794)	**-72.995 ± 1.872 (6.790)**	-956.208 ± 136.611 (890.003)
	0.9	-72.646 ± 3.909 (59.646)	**-38.592 ± 3.373 (25.592)**	-100.000 ± 0.000 (87.000)	-96.819 ± 1.466 (83.819)	-425.837 ± 99.188 (412.837)
TwoRooms	0.1	0.012 ± 0.010 (0.988)	0.022 ± 0.014 (0.978)	0.042 ± 0.020 (0.959)	0.022 ± 0.014 (0.979)	**0.261 ± 0.044 (0.739)**
	0.3	0.077 ± 0.027 (0.923)	0.092 ± 0.029 (0.908)	0.071 ± 0.025 (0.929)	0.081 ± 0.027 (0.919)	**0.530 ± 0.050 (0.470)**
	0.5	0.173 ± 0.039 (0.827)	0.151 ± 0.036 (0.849)	0.182 ± 0.038 (0.818)	0.181 ± 0.038 (0.819)	**0.720 ± 0.045 (0.280)**
	0.7	0.270 ± 0.046 (0.730)	0.371 ± 0.048 (0.629)	0.481 ± 0.050 (0.519)	0.501 ± 0.050 (0.499)	**0.910 ± 0.029 (0.090)**
	0.9	0.732 ± 0.046 (0.268)	0.800 ± 0.040 (0.200)	0.870 ± 0.034 (0.130)	0.770 ± 0.042 (0.230)	**0.990 ± 0.010 (0.010)**

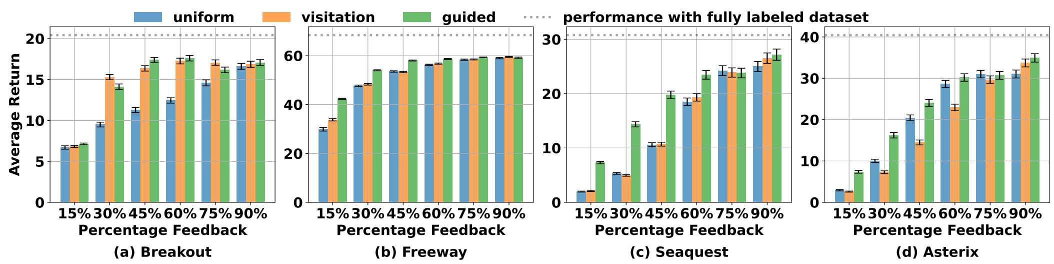

Figure 3: Comparison of `guided`, `visitation`, and `uniform` heuristic selection strategies on four large-scale domains: Breakout, Freeway, Seaquest, and Asterix. For each domain, the plot shows the mean policy return with error bars indicating the standard error.

`Sequential-greedy` search, while more scalable, is also limited to small domains; its training cost scales linearly with both the budget and dataset size. For instance, applying it to Breakout would require roughly 10^8 evaluations when the budget equals the number of unique states in the dataset, which under the same assumptions would take about a month. A notable exception exists in sparse-reward domains, where only states with non-zero rewards need to be considered, substantially reducing the search space (see Appendix D.4).

In contrast, the training computational complexity of `ES` depends only on the number of samples per iteration M and number of iterations K, remaining independent of the state space size and budget. Unlike `sequential-greedy`, which constructs the set incrementally, `ES` evaluates full candidate sets in batch. On small-scale domains, we set $K = 10$ and evaluate two variants: `ES 50` ($M = 5$) and `ES 200` ($M = 20$), where the number in the method name (e.g., 50, 200) refers to the total number of candidate sets evaluated per run, computed as $K \times M$; additional ablations varying K and M on small domains are provided in Appendix D.4. For large-scale domains, we use $K = 10, M = 100$ (`ES 1000`) to accommodate the greater complexity of the state space.

The results on selected small-scale domains (Figure 4) yield three key findings, discussed below. We report only test dataset performance, as policies trained on the same selected state sets across different test datasets exhibit minimal variance, resulting in near-zero standard errors that are omitted for clarity. Full results including training scores and standard errors for all small-scale domains are provided in Appendix D.4.

- `Sequential-greedy` achieves performance comparable to `brute-force`, validating its effectiveness as a scalable approximation to the optimal state set.
- On small-scale domains, `ES 200` achieves performance similar to `sequential-greedy`, while `ES 50` underperforms but can still occasionally exceed the performance of the `guided`

heuristic; this highlights `ES` as a viable alternative when sufficient samples and iterations are used.

- Optimal state sets identified on training datasets generalize well when evaluated on test datasets generated by different data-collecting policies, suggesting robustness to moderate dataset distribution shifts (assuming the test datasets cover the same state space as the training dataset).

On large-scale domains, we observe that `ES` achieves reasonable performance but does not consistently match the best performance at the same budget, as determined by reduced `brute-force` evaluation in sparse-reward domains. This highlights the inherent difficulty of identifying optimal state sets in large state spaces, where the vast number of possible combinations makes accurate estimation challenging. Results and further details are provided in Appendix D.4. Nevertheless, `ES` provides a scalable approximation method when `sequential-greedy` becomes computationally infeasible.

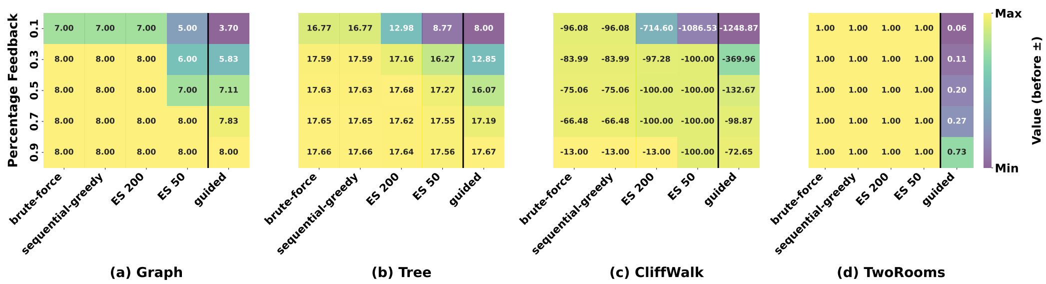

Figure 4: Performance comparison of `brute-force`, `sequential-greedy`, ES, and `guided` on selected small-scale domains. Values indicate mean policy return on test datasets; standard errors are near-zero and thus omitted. `ES 200` corresponds to $K = 10, M = 20$ and `ES 50` to $K = 10, M = 5$. Results are averaged over five test datasets at five percentage feedback levels.

4.3 Structural Patterns and Domain Characteristics

As shown in Figure 4, partially labeled datasets can achieve performance close to fully labeled datasets. To understand this, we examine structural patterns of optimal state sets across domains and identify domain characteristics that influence the effectiveness of reward selection heuristics.

Pattern 1: Prioritizing the Optimal Path. Optimal sets typically include states that allow the agent to consistently follow high-return trajectories. In environments with deterministic transition dynamics—such as Graph—state selection expands backward from the goal as the budget increases. In sparse-reward domains like TwoRooms, labeling the goal state alone suffices under low budgets. MinAtar domains exhibit similar behavior, with selected states reinforcing high-reward behaviors (e.g., paddle-ball alignment in Breakout). Detailed analysis of Breakout, FrozenLake, TwoRooms-Trap, and CliffWalk—which exhibit related but distinct behaviors—is provided in Appendix D.5.

Optimal sets typically include states that allow the agent to consistently follow high-return trajectories. In environments with deterministic transition dynamics—such as Graph—state selection expands backward from the goal as the budget increases. In sparse-reward domains like TwoRooms, labeling the goal state alone suffices under low budgets. MinAtar domains exhibit similar behavior, with selected states reinforcing high-reward behaviors, e.g., paddle-ball alignment in Breakout as shown in Appendix D.5, along with analysis for FrozenLake, TwoRooms-Trap, and CliffWalk, which exhibit related but slightly distinct behaviors.

Pattern 2: Expanding Coverage to Near-Optimal Paths. In stochastic domains like Tree, randomness prevents consistently staying on the optimal path. Optimal state sets expand to include secondary high-reward paths, providing robustness under uncertainty.

Explaining Heuristic Performance via Domain Properties. These patterns explain heuristic results in Section 3.1. `Visitation` works well when the data-collecting policy already visits valuable states (e.g., Graph, MinAtar). `Guided` improves over `visitation` by favoring states leading to high-value regions (Pattern 1). Both fail in domains where avoiding bad outcomes is critical, such as CliffWalk (Appendix D.5). Bottlenecks, like in TwoRooms, cause `visitation` to over-focus on frequent but suboptimal regions, where `uniform` achieves better stable coverage.

5 Related Work

The setup of active reward selection for RLLF has not been previously explored much. The closest formalization is by Parisi et al. (2024a), who consider *partially observable rewards* in online RL, but their setting conflates exploration with reward acquisition, making the focus different from our purely offline formulation. Zhan et al. (2023) propose a sampling approach for reward annotation but assume linear reward models, whereas our method does not impose such structural constraints. *Active RL* studies querying strategies under online exploration constraints, where agents must pay to observe rewards (Krueger et al., 2020; Schulze and Evans, 2018; Tucker et al., 2023). Our setting differs fundamentally: we study offline data with no additional exploration burden. Relatedly, Konyushova et al. (2021) address active off-policy data selection to improve policy evaluation, focusing on policy-level data collection rather than fine-grained reward state selection. The use of non reward labeled data has been studied for *online (state-based) exploration with unlabeled samples*. Some methods pseudo-label unlabeled samples to improve online exploration (Wilcoxson et al., 2024; Li et al., 2024), or develop exploration algorithms that operate under missing reward labels (Parisi et al., 2024b; Huang et al.). However, these primarily study exploration dynamics, whereas our focus is purely on optimizing offline reward label acquisition. Yu et al. (2022) show that imputing zero reward for unlabeled samples can work surprisingly well, which aligns with certain observations in our study. Other recent work explores reward modeling under uncertainty, for example, using priors over reward functions (Hu et al., 2023) or studying data influence (Munos and Moore, 2002; Koh and Liang, 2017; Gottesman et al., 2020). We complement these analyses by studying how selectively adding reward labels to previously unlabeled data influences the resulting policy performance. A detailed comparison with these and additional works is provided in Appendix B.

6 Discussion and Conclusion

We introduce reward selection as a critical but underexplored challenge in RLLF. By decoupling selection from policy learning, we present the first systematic evaluation of zero-shot heuristics and optimized strategies across diverse environments, defining simple yet strong baselines and offering insights for future reward-efficient algorithms in domains like RLHF and drug discovery. The effectiveness of reward selection varies with domain dynamics and reward structure: in deterministic settings with frequent rewards, path-following heuristics perform well; in stochastic or sparse-reward domains,strategies that promote broader state coverage prove more effective. No single heuristic dominates across all cases, and effective selection must align with both the domain and learning algorithm.

While our study focuses on value-based policy updates, extending selection strategies to policy-gradient methods is a promising direction. Additionally, our general framework abstracts away domain-specific structure; however, incorporating inductive biases, such as temporal correlations in time-series tasks, may further aid selection strategies. Exploring how to integrate such structured priors offers an exciting path for future work. Together, these findings establish reward selection as a powerful paradigm for scaling reinforcement learning in limited feedback settings.

References

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

ABAKA AI. Llm data cost breakdown: All you need to know about data costs for training an llm, 2025. URL https://www.abaka.ai/blog/llm-data-cost.

Jean-Louis Reymond. The chemical space project. *Accounts of chemical research*, 48(3):722–730, 2015.

Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical graph-to-graph translation for molecules. *arXiv preprint arXiv:1907.11223*, 2019.

Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. Innovation in the pharmaceutical industry: new estimates of r&d costs. *Journal of health economics*, 47:20–33, 2016.

Anon. Ai's potential to accelerate drug discovery needs a reality check. *Nature*, 622:217, 2023.

Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

Shuo Feng, Haowei Sun, Xintao Yan, Haojie Zhu, Zhengxia Zou, Shengyin Shen, and Henry X Liu. Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*, 615(7953): 620–627, 2023.

David Krueger, Jan Leike, Owain Evans, and John Salvatier. Active reinforcement learning: Observing rewards at a cost. *arXiv preprint arXiv:2011.06709*, 2020.

Simone Parisi, Montaser Mohammedalamen, Alireza Kazemipour, Matthew E Taylor, and Michael Bowling. Monitored markov decision processes. *arXiv preprint arXiv:2402.06819*, 2024a.

Tianhe Yu, Aviral Kumar, Yevgen Chebotar, Karol Hausman, Chelsea Finn, and Sergey Levine. How to leverage unlabeled data in offline reinforcement learning. In *International Conference on Machine Learning*, pages 25611–25635. PMLR, 2022.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Ingo Rechenberg. Evolution strategy: Nature's way of optimization. In *Optimization: Methods and Applications, Possibilities and Limitations: Proceedings of an International Seminar Organized by Deutsche Forschungsanstalt für Luft-und Raumfahrt (DLR), Bonn, June 1989*, pages 106–126. Springer, 1989.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. URL https://arxiv.org/abs/1606.01540.

Farama Foundation. Gymnasium. https://gymnasium.farama.org, 2023.

Wenhao Zhan, Masatoshi Uehara, Wen Sun, and Jason D Lee. How to query human feedback efficiently in rl? 2023.

Sebastian Schulze and Owain Evans. Active reinforcement learning with monte-carlo tree search. *arXiv preprint arXiv:1803.04926*, 2018.

Aaron D Tucker, Caleb Biddulph, Claire Wang, and Thorsten Joachims. Bandits with costly reward observations. In *Uncertainty in Artificial Intelligence*, pages 2147–2156. PMLR, 2023.

Ksenia Konyushova, Yutian Chen, Thomas Paine, Caglar Gulcehre, Cosmin Paduraru, Daniel J Mankowitz, Misha Denil, and Nando de Freitas. Active offline policy selection. *Advances in Neural Information Processing Systems*, 34:24631–24644, 2021.

Max Wilcoxson, Qiyang Li, Kevin Frans, and Sergey Levine. Leveraging skills from unlabeled prior data for efficient online exploration. *arXiv preprint arXiv:2410.18076*, 2024.

Qiyang Li, Jason Zhang, Dibya Ghosh, Amy Zhang, and Sergey Levine. Accelerating exploration with unlabeled prior data. *Advances in Neural Information Processing Systems*, 36, 2024.

Simone Parisi, Alireza Kazemipour, and Michael Bowling. Beyond optimism: Exploration with partially observable rewards. *arXiv preprint arXiv:2406.13909*, 2024b.

Audrey Huang, Mohammad Ghavamzadeh, Nan Jiang, and Marek Petrik. Non-adaptive online finetuning for offline reinforcement learning. In *NeurIPS 2023 Workshop on Generalization in Planning*.

Hao Hu, Yiqin Yang, Jianing Ye, Ziqing Mai, and Chongjie Zhang. Unsupervised behavior extraction via random intent priors. *Advances in Neural Information Processing Systems*, 36:51491–51514, 2023.

Remi Munos and Andrew Moore. Variable resolution discretization in optimal control. *Machine learning*, 49:291–323, 2002.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

Omer Gottesman, Joseph Futoma, Yao Liu, Sonali Parbhoo, Leo Celi, Emma Brunskill, and Finale Doshi-Velez. Interpretable off-policy evaluation in reinforcement learning by highlighting influential transitions. In *International Conference on Machine Learning*, pages 3658–3667. PMLR, 2020.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

David Lindner, Matteo Turchetta, Sebastian Tschiatschek, Kamil Ciosek, and Andreas Krause. Information directed reward learning for reinforcement learning. *Advances in Neural Information Processing Systems*, 34:3850–3862, 2021.

Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International Conference on Machine Learning*, pages 5084–5096. PMLR, 2021.

Tengyang Xie, Ching-An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. *Advances in neural information processing systems*, 34:6683–6694, 2021.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Additional Motivating Examples

1. **Reinforcement Learning from Human Feedback (RLHF) in LLMs:** In training large language models (LLMs), model-generated outputs are plentiful, but high-quality human preference labels remain costly and scarce (Ouyang et al., 2022; Christiano et al., 2017). This creates a reward selection challenge: which model completions should be labeled with human feedback to best guide downstream policy improvement? This mirrors our setup, where a budgeted selection of feedback points must be made to train a performant policy while minimizing labeling operational cost (ABAKA AI, 2025).
2. **AI-driven Drug Discovery:** Generative models can propose vast libraries of candidate molecules (Gómez-Bombarelli et al., 2018; Reymond, 2015; Jin et al., 2019), but only a limited subset can be experimentally evaluated for synthesizability, bioactivity, and toxicity due to the cost and time of wet-lab trials (DiMasi et al., 2016). Reward selection here involves choosing which molecular candidates to evaluate, analogous to selecting states for reward labeling in our framework to maximize downstream performance within a practically limited evaluation budget.
3. **Autonomous Driving:** Simulation platforms can produce diverse driving trajectories across environments and policies at scale (Dosovitskiy et al., 2017), but obtaining expert evaluations—such as comfort, rule compliance, or safety—is resource-intensive (Feng et al., 2023). Thus, a reward selection strategy is needed to determine which trajectories to annotate to yield robust, deployable policies, much like our proposed approach to feedback-efficient learning.
4. **Robotics:** Simulated environments enable generation of numerous trajectories, but transferring and evaluating those policies in the real world involves expensive and time-consuming physical experiments (Rajeswaran et al., 2017; Chebotar et al., 2019). Reward selection in this domain involves prioritizing which simulated or real-world interactions to evaluate, paralleling our method's goal of selecting the most informative reward-labeled samples for efficient policy learning under cost constraints.

B Additional Related Works

The setup of active reward selection for RLLF has not been previously explored much. The closest formulation of this problem is in Parisi et al. (2024a), who provide a formulation for *partially observable rewards* in online RL and propose algorithms for that setting. The online formulation conflates the difficulty on online exploration with the utility of rewards, the latter being the focus of this work. sampling approach to acquiring exploratory trajectories that enable accurate learning of hidden reward functions before collecting any human feedback. Zhan et al. (2023) propose a sampling approach to acquire data to be reward-annotated, although their analysis assumes linearity of reward functions. Similar to discovering high-utility reward states, Konyushova et al. (2021) study active collection of online data to determine promising policies and improve their performance estimates, as active off-policy selection.

The topic of reward selection has been studied under *Active RL*, which is perhaps closest in its motivation to our setting: where the agent must pay a cost to observe the reward, although for an online setting, yet again conflating the difficulty of exploration with the utility of rewards. Krueger et al. (2020) study this in the bandit setting, while Tucker et al. (2023) extend it to structured settings but retain the bandit-style objective of identifying the best arm by using reward queries to increase confidence in the average (stochastic) outcomes of each arm. This differs from our problem in two major ways: the stochasticity of rewards for each arm forces repeated sampling, and the lack of sequentiality of actions (leading to different outcomes for repeated pulls of the same arm) shifts the focus from reward utility to uncertainty mitigation. In contrast, Schulze and Evans (2018) propose a Bayes-optimal algorithm using Monte Carlo Tree Search (MCTS) to actively select reward observations. Finally, approaches like Lindner et al. (2021) actively select queries to maximize information gain about the reward function for modeling it.

The use of non-reward-labeled data has been extensively explored in the context of *online state-based exploration with unlabeled samples*. Wilcoxson et al. (2024) propose assigning pseudolabels to unlabeled data to guide exploration, while Li et al. (2024) leverage prior offline datasets and online rewards to pseudo-label new data for improved exploration. Parisi et al. (2024b) examine exploration under partially observed rewards, a setting closely related to ours but focused on online interaction. Huang et al. introduce a data collection strategy combining online RL with offline datasets to approach the performance of the optimal policy. Yu et al. (2022) show that setting unknown rewards to zero can perform surprisingly well in certain offline RL settings, a finding we also confirm in our experiments. Hu et al. (2023) propose using unlabeled data by assuming priors over possible reward functions and optimizing over sampled realizations of those reward functions.

Beyond data-driven exploration, *influence functions* have been proposed as signals for high-utility rewards. Munos and Moore (2002) defines the influence of a reward on value as $\frac{\partial V^*(s)}{\partial R(s')}$, equivalent to the state visitation frequency under the optimal policy. Other works, such as Koh and Liang (2017) and Gottesman et al. (2020), analyze the effect of removing known datapoints on prediction performance. In contrast, we study the anticipated influence of adding partially unknown datapoints, requiring assumptions about their potential impact. Finally, Lindner et al. (2021) provide an algorithm for learning reward models independently of the reward querying process, which relates directly to the focus of our study.

C Additional Notes on Methodology

C.1 Categorization of Reward Selection Strategies Investigated

We categorize the reward state selection strategies introduced in Section 3 according to three key design dimensions: (i) whether selection during the test phase is performed in an *open-loop* or *closed-loop* manner, (ii) whether training-phase selection operates in a *batch* or *iterative* mode, and (iii) the degree to which each strategy utilizes the *evaluator* during training. Table 2 presents a high-level taxonomy across these dimensions.

Selection Strategy	Test: Open/Closed Loop	Train: Batch/Iterative	Train: Evaluator Use
	Trained Strategies		
`brute-force`	Open loop	Batch	Yes
`sequential-greedy`	Open loop	Iterative	Yes
`evolutionary-strategy`	Open loop	Batch	Yes
	Training-free Heuristics		
`guided`	Closed loop	Iterative	No
`guided-on-policy`	Closed loop	Iterative	No
`visitation`	Open loop	Batch	No
`visitation-on-policy`	Closed loop	Iterative	No
`uniform`	Open loop	Batch	No

Table 2: Categorization of reward selection methods by design dimensions. Columns are shaded to distinguish test-phase (green) and training-phase (blue) attributes. Methods are grouped based on whether they use the evaluator during training.

C.2 Description and Notation for Iterative Reward Selection Strategies

Iterative reward selection strategies construct the reward-labeled state set $\mathcal{S}_{[B]}$ in a sequential manner. At each step $b \in \{1, \ldots, B\}$, a new state $s_b \in \mathcal{S}$ is selected—conditioned on relevant information such as the current estimates of the Q-values of the policy or current policy's state-visitation distribution—and added to the selection set $\mathcal{S}_{[b-1]}$ to form $\mathcal{S}_{[B]}$. Relevant notation:

- $\mathcal{S}_{[b]}$: The set of selected states after b iterations, i.e., $\mathcal{S}_{[b]} = \mathcal{S}_{[b-1]} \cup \{s_b\}$.
- q_b: The selection strategy or distribution used to sample the next state s_b at iteration b, potentially conditioned on policy information or prior selections.
- $\pi_{[b]}$: The intermediate policy obtained after the b^{th} reward selection and updated via `RLLF`.
- $Q^{\pi_{[b-1]}}$: The Q-function corresponding to $\pi_{[b-1]}$ after the $(b-1)^{\text{th}}$ reward selection and update.

D Additional Experiments and Empirical Details

D.1 Domain Details

Table 3 summarizes the domains and their corresponding experimental setup. We study six small-scale domains (Graph, Tree, TwoRooms, TwoRooms-Trap, FrozenLake, and CliffWalk) and four large-scale MinAtar domains (breakout, freeway, seaquest, and asterix). The graph, tree, twoRooms, and twoRooms-Trap domains are custom-designed to expose structural properties relevant for analyzing reward selection strategies, while frozenLake and cliffWalk are standard Gymnasium benchmarks (Brockman et al., 2016; Foundation, 2023).

Table 3: Summary of domains and their experimental setup.

	Small-scale Domains	**Large-scale Domains (MinAtar)**
Domain Names	graph, tree, twoRooms, twoRooms-Trap, frozenLake, cliffWalk	breakout, freeway, seaquest, asterix
State Representation	Numeric (tabular)	Image-based (10×10 pixels)
Expert Policy	Value Iteration	Online DQN
Policy Learning Algorithm	Offline Q-learning	Implicit Q-learning (IQL)

Domain description Brief descriptions of all domains are provided below.

- **Graph**: A two-row graph structure with 8 nodes per row. In each adjacent column, the 2×2 nodes are fully connected. Transitions are deterministic; actions move the agent between rows or advance to the next column in the same row. States correspond to nodes; every movement yields a dense reward.
- **Tree**: A complete binary tree where actions correspond to moving left or right. Transitions are stochastic: the agent moves in the intended direction with 85% probability and in the alternate direction with 15%. Rewards are dense and provided at every step.
- **TwoRooms**: Two 5×5 gridworld rooms connected by a narrow bottleneck state. The agent starts in one room and must reach a goal located in the other. Rewards are sparse: zero everywhere except a reward of 1 at the goal state.
- **TwoRooms-Trap**: A variant of TwoRooms with six additional trap states. Entering a trap terminates the episode immediately with a penalty of -100. The environment otherwise shares the layout and reward structure of TwoRooms.
- **FrozenLake**: A standard Gymnasium benchmark (Brockman et al., 2016; Foundation, 2023). The agent navigates a slippery grid from start to goal, avoiding holes that cause termination. Transitions are stochastic and rewards are sparse (reward only at the goal).
- **CliffWalk**: Another Gymnasium benchmark. The agent must traverse a grid from start to goal while avoiding a high-penalty cliff region. Transitions are deterministic.
- **Minatar**: A set of simplified Atari-inspired environments with compact state and action spaces (Young and Tian, 2019). We evaluate on Breakout, Freeway, Seaquest, and Asterix.

Policy training For small-scale domains, expert policies are generated using value iteration and policies are trained with offline Q-learning. For large-scale MinAtar domains, expert policies are obtained by training online DQN agents, and offline learning uses implicit Q-learning (IQL). Small-scale domains use tabular Q-functions due to their discrete, low-dimensional state spaces, while large-scale domains rely on neural network approximators for Q-values, given their high-dimensional 10×10 image-based states.

Dataset collection Datasets are collected using a mixture-based data-collecting policy that combines expert and random actions. At each timestep, the agent follows the expert policy with probability ϵ and takes a uniformly random action with probability $1 - \epsilon$. For training, we use a single data-collecting policy with $\epsilon = 0.5$. For evaluation, five test data-collecting policies are created with

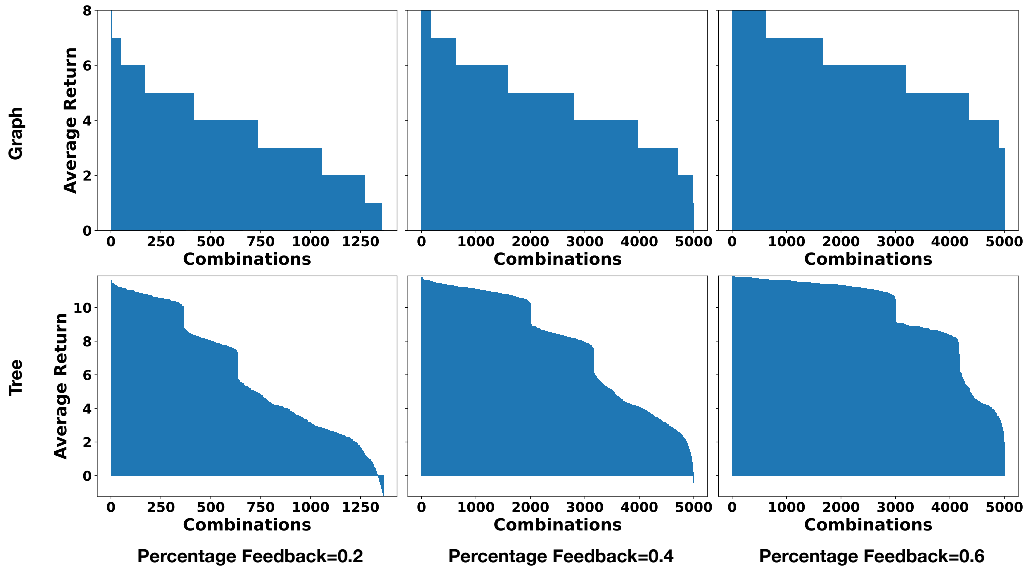

Figure 5: Performance variability across different reward-labeled state sets at fixed budgets. The first row shows results for the Graph domain; the second row shows results for the Tree domain. Columns correspond to percentage feedback levels of 20%, 40%, and 60%, respectively. The results illustrate that at the same feedback level, the choice of which states are labeled strongly affects the resulting policy performance.

$\epsilon \in \{0.55, 0.53, 0.51, 0.48, 0.45\}$ to study the robustness of learned policies under small distribution shifts.

Computing resource All experiments on small-scale domains were conducted on CPUs, while those on large-scale domains were run on GeForce RTX 2080 Tis.

D.2 Different reward-labeled-sets result in policies with varying performances

In Figure 1, we illustrate that different reward-labeled sets lead to policies with varying performance. We empirically validate this observation in two small-scale domains, Graph and Tree. For each domain, we select three percentage feedbacks (20%, 40%, and 60%), and report the average return of policies learned from all possible combinations at that budget. For example, in the Graph domain, which has 16 total states, selecting $b = 2$ yields $\binom{16}{2} = 120$ possible combinations; we report the average return across policies trained on datasets labeled by each of these 120 state sets. The results, shown in Figure 5, demonstrate that for a fixed budget, different combinations of labeled states can lead to significantly different policy performance.

D.3 Additional Results for Selection Heuristics

The heuristics results on all small-scale domains are shown in Table 4.

Table 4: Comparison of `guided`, `visitation`, and `uniform` heuristic selection strategies on small-scale domains. For each domain, the table presents the mean policy return (± standard error) and the corresponding optimality gap (in parentheses) across five percentage feedback levels.

Domains	Percentage Feedback	`guided`	`guided-on-policy`	`visitation`	`visitation-on-policy`	`uniform`
Graph	0.1	3.701 ± 0.129 (3.302)	3.208 ± 0.139 (3.795)	**3.797 ± 0.151 (3.206)**	3.300 ± 0.142 (3.703)	2.949 ± 0.137 (4.054)
	0.3	5.831 ± 0.137 (2.169)	5.760 ± 0.127 (2.240)	**5.871 ± 0.146 (2.129)**	5.690 ± 0.146 (2.310)	4.617 ± 0.156 (3.383)
	0.5	7.110 ± 0.099 (0.890)	**7.690 ± 0.070 (0.310)**	7.199 ± 0.090 (0.801)	7.583 ± 0.086 (0.417)	5.978 ± 0.114 (2.022)
	0.7	7.830 ± 0.040 (0.170)	**8.000 ± 0.000 (0.000)**	7.599 ± 0.060 (0.401)	7.991 ± 0.009 (0.009)	6.920 ± 0.084 (1.080)
	0.9	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**
Tree	0.1	**8.003 ± 0.468 (9.053)**	7.403 ± 0.869 (9.653)	6.133 ± 0.428 (10.924)	4.658 ± 0.370 (12.398)	5.665 ± 0.532 (11.392)
	0.3	**12.846 ± 0.373 (4.921)**	12.755 ± 0.632 (5.013)	11.763 ± 0.427 (6.004)	12.601 ± 0.414 (5.167)	10.341 ± 0.524 (7.427)
	0.5	16.072 ± 0.205 (1.695)	**16.415 ± 0.207 (1.352)**	15.395 ± 0.297 (2.372)	16.379 ± 0.216 (1.388)	13.218 ± 0.430 (4.550)
	0.7	17.193 ± 0.083 (0.575)	**17.444 ± 0.037 (0.323)**	17.135 ± 0.153 (0.633)	17.174 ± 0.120 (0.594)	15.258 ± 0.312 (2.509)
	0.9	17.673 ± 0.013 (0.094)	**17.731 ± 0.031 (0.036)**	17.609 ± 0.158 (0.049)	17.521 ± 0.110 (0.246)	17.695 ± 0.141 (0.072)
CliffWalk	0.1	−1248.872 ± 117.272 (1152.914)	−616.760 ± 105.578 (520.803)	−1262.067 ± 119.207 (1166.109)	**−392.040 ± 84.184 (296.082)**	−1156.960 ± 61.025 (1061.002)
	0.3	−369.964 ± 86.539 (285.948)	−93.637 ± 0.373 (9.621)	−462.530 ± 100.633 (378.515)	**−92.981 ± 0.358 (8.965)**	−1274.561 ± 118.910 (1190.545)
	0.5	−132.671 ± 32.819 (57.629)	−89.390 ± 0.827 (14.348)	−165.870 ± 46.201 (90.828)	**−86.746 ± 0.677 (11.704)**	−1235.823 ± 137.366 (1160.781)
	0.7	−98.870 ± 0.647 (32.665)	−74.821 ± 2.171 (8.615)	−100.000 ± 0.000 (33.794)	**−72.995 ± 1.872 (6.790)**	−956.208 ± 136.611 (890.003)
	0.9	−72.646 ± 3.909 (59.646)	**−38.592 ± 3.373 (25.592)**	−100.000 ± 0.000 (87.000)	−96.819 ± 1.466 (83.819)	−425.837 ± 99.188 (412.837)
FrozenLake	0.1	0.021 ± 0.007 (−0.721)	0.056 ± 0.017 (−0.686)	0.028 ± 0.010 (−0.714)	0.020 ± 0.007 (−0.722)	**0.145 ± 0.028 (−0.598)**
	0.3	0.087 ± 0.022 (−0.655)	0.078 ± 0.021 (−0.663)	0.079 ± 0.021 (−0.663)	0.050 ± 0.016 (−0.692)	**0.306 ± 0.036 (−0.436)**
	0.5	0.165 ± 0.029 (−0.578)	0.127 ± 0.026 (−0.617)	0.171 ± 0.030 (−0.573)	0.086 ± 0.022 (−0.657)	**0.467 ± 0.036 (−0.276)**
	0.7	0.261 ± 0.034 (−0.482)	0.251 ± 0.034 (−0.492)	0.326 ± 0.036 (−0.416)	0.160 ± 0.029 (−0.582)	**0.582 ± 0.031 (−0.160)**
	0.9	0.477 ± 0.035 (−0.263)	0.508 ± 0.033 (−0.232)	0.566 ± 0.031 (−0.174)	0.427 ± 0.036 (−0.313)	**0.697 ± 0.019 (−0.043)**
TwoRooms	0.1	0.012 ± 0.010 (0.988)	0.022 ± 0.014 (0.978)	0.042 ± 0.020 (0.959)	0.022 ± 0.014 (0.979)	**0.261 ± 0.044 (0.739)**
	0.3	0.077 ± 0.027 (0.923)	0.092 ± 0.029 (0.908)	0.071 ± 0.025 (0.929)	0.081 ± 0.027 (0.919)	**0.530 ± 0.050 (0.470)**
	0.5	0.173 ± 0.039 (0.827)	0.151 ± 0.036 (0.849)	0.182 ± 0.038 (0.818)	0.181 ± 0.038 (0.819)	**0.720 ± 0.045 (0.280)**
	0.7	0.270 ± 0.046 (0.730)	0.371 ± 0.048 (0.629)	0.481 ± 0.050 (0.519)	0.501 ± 0.050 (0.499)	**0.910 ± 0.029 (0.090)**
	0.9	0.732 ± 0.046 (0.268)	0.800 ± 0.040 (0.200)	0.870 ± 0.034 (0.130)	0.770 ± 0.042 (0.230)	**0.990 ± 0.010 (0.010)**
TwoRooms-Trap	0.1	−58.492 ± 0.642 (59.492)	−60.151 ± 0.673 (61.151)	−59.390 ± 1.156 (60.390)	−61.520 ± 0.723 (62.520)	**−46.850 ± 2.884 (47.850)**
	0.3	−45.692 ± 1.015 (46.692)	−47.391 ± 0.899 (48.391)	−47.130 ± 1.538 (48.130)	−49.960 ± 1.022 (50.960)	**−29.340 ± 2.983 (30.340)**
	0.5	−16.440 ± 0.968 (17.440)	−15.374 ± 0.874 (16.374)	−23.320 ± 1.396 (24.320)	−20.140 ± 0.934 (21.140)	**−13.270 ± 2.261 (14.270)**
	0.7	−0.336 ± 0.056 (1.336)	**−0.210 ± 0.065 (1.210)**	−0.300 ± 0.349 (1.300)	−0.700 ± 0.160 (1.700)	−1.600 ± 0.916 (2.600)
	0.9	**1.000 ± 0.000 (0.000)**	**1.000 ± 0.000 (0.000)**	**1.000 ± 0.000 (0.000)**	0.851 ± 0.040 (0.149)	**1.000 ± 0.000 (0.000)**

D.4 Additional Results for Selection Optimality

In sparse-reward environments, `brute-force` search can be accelerated by recognizing that only states with non-zero rewards must be labeled. This greatly reduces the number of combinations to consider, making exact evaluation tractable in small domains. The optimality results on all small-scale domains are shown in Table 5.

Table 5: Performance comparison of brute-force, sequential greedy, and Evolutionary Strategy (ES) on small-scale domains. Results are reported on training datasets, with test performance shown in parentheses (e.g., train score (test score)). Test scores are reported as mean ± standard error across five test datasets. ES 200 corresponds to $K = 10, M = 20$ and ES 50 to $K = 10, M = 5$.

Domains	Percentage Feedback	`brute-force`	`sequential-greedy`	`ES 200`	`ES 50`	`guided`
Graph	0.1	7.003(7.003 ± 0.000)	7.003(7.003 ± 0.000)	7.003(7.003 ± 0.000)	4.999(4.996 ± 0.000)	3.701
	0.3	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	6.000(6.000 ± 0.000)	5.831
	0.5	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	7.003(7.003 ± 0.000)	7.110
	0.7	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	7.830
	0.9	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000
Tree	0.1	17.056(16.773 ± 0.000)	17.056(16.773 ± 0.000)	12.990(12.978 ± 0.000)	8.841(8.768 ± 0.000)	8.003
	0.3	17.767(17.592 ± 0.017)	17.767(17.592 ± 0.033)	17.198(17.157 ± 0.000)	16.199(16.271 ± 0.000)	12.846
	0.5	17.767(17.629 ± 0.020)	17.767(17.629 ± 0.018)	17.781(17.680 ± 0.000)	17.445(17.275 ± 0.009)	16.072
	0.7	17.767(17.649 ± 0.000)	17.767(17.649 ± 0.000)	17.777(17.623 ± 0.000)	17.642(17.547 ± 0.000)	17.193
	0.9	17.767(17.657 ± 0.000)	17.767(17.657 ± 0.000)	17.736(17.639 ± 0.000)	17.746(17.564 ± 0.000)	17.673
CliffWalk	0.1	−95.958(−96.081 ± 0.001)	−95.958(−96.081 ± 0.001)	−713.261(−714.600 ± 0.019)	−1086.006(−1086.526 ± 0.039)	−1248.872
	0.3	−84.016(−83.986 ± 0.001)	−84.016(−83.986 ± 0.001)	−97.237(−97.276 ± 0.000)	−100.000(−100.000 ± 0.000)	−369.964
	0.5	−75.042(−75.059 ± 0.001)	−75.042(−75.059 ± 0.001)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−132.671
	0.7	−66.206(−66.477 ± 0.001)	−66.206(−66.477 ± 0.001)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−98.870
	0.9	−13.000(−13.000 ± 0.000)	−13.000(−13.000 ± 0.000)	−13.000(−13.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−72.646
FrozenLake	0.1	0.746(0.729 ± 0.010)	0.746(0.729 ± 0.010)	0.742(0.728 ± 0.009)	0.014(0.014 ± 0.000)	0.021
	0.3	0.746(0.736 ± 0.006)	0.746(0.736 ± 0.006)	0.738(0.702 ± 0.010)	0.738(0.730 ± 0.008)	0.087
	0.5	0.746(0.719 ± 0.012)	0.746(0.719 ± 0.012)	0.740(0.731 ± 0.009)	0.737(0.714 ± 0.009)	0.165
	0.7	0.746(0.728 ± 0.007)	0.746(0.728 ± 0.007)	0.733(0.730 ± 0.010)	0.742(0.737 ± 0.002)	0.261
	0.9	0.746(0.719 ± 0.008)	0.746(0.719 ± 0.008)	0.739(0.740 ± 0.001)	0.743(0.734 ± 0.005)	0.477
TwoRooms	0.1	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	0.055
	0.3	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	0.109
	0.5	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	0.195
	0.7	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	0.270
	0.9	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	0.732
TwoRooms-Trap	0.1	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	−37.204
	0.3	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	−16.440
	0.5	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	−1.397
	0.7	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	0.966
	0.9	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000

In addition to the two ES variants presented in Section 4.2, we provide an ablation study examining how performance varies with different numbers of samples per iteration M and iterations K. In Table 6, we fix $M = 20$ and vary K across $\{3, 5, 8, 10\}$. In Table 7, we fix $K = 10$ and vary M across $\{5, 10, 15, 20\}$. We find that larger values of $K \times M$ generally lead to better performance.

Notably, increasing M (the number of samples per iteration) tends to have a greater impact than increasing K (the number of iterations), suggesting that sampling more candidates per iteration contributes more significantly to performance gains than simply running additional iterations.

Table 6: Ablation study of ES performance as a function of the number of iterations K (with $M = 20$ fixed). Results are reported as $K \times M$ for consistency with the main paper (e.g., ES 10×20 indicates $K = 10$ and $M = 20$).

Domains	Percentage Feedback	ES 10 × 20	ES 8 × 20	ES 5 × 20	ES 3 × 20
Graph	0.1	7.003(7.003 ± 0.000)	7.003(7.003 ± 0.000)	7.003(7.003 ± 0.000)	7.003(7.003 ± 0.000)
	0.3	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)
	0.5	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)
	0.7	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)
	0.9	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)
Tree	0.1	12.990(12.978 ± 0.000)	12.990(12.978 ± 0.000)	12.880(12.884 ± 0.000)	11.820(11.897 ± 0.086)
	0.3	17.198(17.157 ± 0.000)	17.329(17.111 ± 0.000)	17.436(17.464 ± 0.000)	16.357(16.161 ± 0.000)
	0.5	17.781(17.680 ± 0.000)	17.692(17.518 ± 0.009)	17.583(17.535 ± 0.000)	17.016(16.911 ± 0.000)
	0.7	17.777(17.623 ± 0.000)	17.763(17.603 ± 0.000)	17.846(17.668 ± 0.000)	17.721(17.552 ± 0.000)
	0.9	17.736(17.639 ± 0.000)	17.746(17.564 ± 0.000)	17.746(17.564 ± 0.000)	17.746(17.564 ± 0.000)
CliffWalk	0.1	−713.261(−714.600 ± 0.019)	−713.261(−714.567 ± 0.012)	−767.641(−769.536 ± 0.028)	−783.801(−786.146 ± 0.021)
	0.3	−97.237(−97.276 ± 0.000)	−97.329(−97.361 ± 0.000)	−95.920(−95.841 ± 0.001)	−100.000(−100.000 ± 0.000)
	0.5	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)
	0.7	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)
	0.9	−13.000(−13.000 ± 0.000)	−13.000(−13.000 ± 0.000)	−13.000(−13.000 ± 0.000)	−14.000(−13.996 ± 0.000)
FrozenLake	0.1	0.742(0.728 ± 0.009)	0.743(0.741 ± 0.001)	0.740(0.740 ± 0.001)	0.737(0.721 ± 0.010)
	0.3	0.738(0.702 ± 0.010)	0.740(0.727 ± 0.006)	0.743(0.735 ± 0.006)	0.738(0.735 ± 0.006)
	0.5	0.740(0.731 ± 0.009)	0.743(0.735 ± 0.005)	0.740(0.710 ± 0.011)	0.737(0.734 ± 0.005)
	0.7	0.733(0.730 ± 0.010)	0.740(0.714 ± 0.014)	0.741(0.725 ± 0.013)	0.739(0.710 ± 0.008)
	0.9	0.739(0.740 ± 0.001)	0.739(0.723 ± 0.007)	0.742(0.710 ± 0.013)	0.740(0.734 ± 0.005)
TwoRooms	0.1	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.3	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.5	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.7	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.9	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
TwoRooms-Trap	0.1	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.3	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.5	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.7	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.9	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)

Table 7: Ablation study of ES performance as a function of the number of samples per iteration M (with $K = 10$ fixed). Results are reported as $K \times M$ for consistency with the main paper (e.g., ES 10×20 indicates $K = 10$ and $M = 20$).

Domains	Percentage Feedback	ES 10 × 20	ES 10 × 15	ES 10 × 10	ES 10 × 5
Graph	0.1	7.003(7.003 ± 0.000)	5.999(6.001 ± 0.000)	5.999(6.001 ± 0.000)	4.999(4.996 ± 0.000)
	0.3	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	6.000(6.000 ± 0.000)
	0.5	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	7.003(7.003 ± 0.000)
	0.7	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)
	0.9	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)	8.000(8.000 ± 0.000)
Tree	0.1	12.990(12.978 ± 0.000)	12.754(12.301 ± 0.116)	12.427(12.516 ± 0.000)	8.841(8.768 ± 0.000)
	0.3	17.198(17.157 ± 0.000)	17.319(17.219 ± 0.000)	17.082(17.015 ± 0.002)	16.199(16.271 ± 0.000)
	0.5	17.781(17.680 ± 0.000)	17.454(17.334 ± 0.000)	17.328(17.290 ± 0.034)	17.445(17.275 ± 0.009)
	0.7	17.777(17.623 ± 0.000)	17.742(17.603 ± 0.000)	17.727(17.726 ± 0.000)	17.642(17.547 ± 0.000)
	0.9	17.736(17.639 ± 0.000)	17.736(17.639 ± 0.000)	17.736(17.639 ± 0.000)	17.746(17.564 ± 0.000)
CliffWalk	0.1	−713.261(−714.600 ± 0.019)	−755.425(−754.434 ± 0.000)	−867.262(−865.682 ± 0.021)	−1086.006(−1086.526 ± 0.039)
	0.3	−97.237(−97.276 ± 0.000)	−98.579(−98.576 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)
	0.5	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)
	0.7	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)
	0.9	−13.000(−13.000 ± 0.000)	−13.000(−13.000 ± 0.000)	−100.000(−100.000 ± 0.000)	−100.000(−100.000 ± 0.000)
FrozenLake	0.1	0.742(0.728 ± 0.009)	0.739(0.698 ± 0.011)	0.741(0.720 ± 0.013)	0.014(0.014 ± 0.000)
	0.3	0.738(0.702 ± 0.010)	0.744(0.741 ± 0.002)	0.740(0.728 ± 0.007)	0.738(0.730 ± 0.008)
	0.5	0.740(0.731 ± 0.009)	0.739(0.723 ± 0.009)	0.740(0.733 ± 0.005)	0.737(0.714 ± 0.009)
	0.7	0.733(0.730 ± 0.010)	0.739(0.711 ± 0.014)	0.739(0.722 ± 0.006)	0.742(0.737 ± 0.002)
	0.9	0.739(0.740 ± 0.001)	0.738(0.737 ± 0.005)	0.738(0.731 ± 0.008)	0.743(0.734 ± 0.005)
TwoRooms	0.1	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.3	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.5	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.7	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.9	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
TwoRooms-Trap	0.34	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.48	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.61	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.75	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)
	0.89	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)	1.000(1.000 ± 0.000)

We also report ES results on large-scale MinAtar domains, using $K = 10, M = 100$ (ES 1000). Although the training computation of ES remains fixed, achieving accurate performance estimates still requires large $K \times M$ values. Even under this configuration, ES does not consistently outperform `guided`, illustrating the inherent difficulty of discovering optimal state sets in large state spaces even when an evaluator is available, as shown in Table 8.

In addition, Table 8 includes a column for reduced `brute-force`. By leveraging UDS, we only label the data points where rewards are non-zero. All four MinAtar domains exhibit sparse rewards, with fewer than 10% of states containing non-zero rewards. As a result, reduced `brute-force` is expected to identify a state set that achieves equivalent performance to the fully labeled dataset, while substantially reducing the labeling effort.

Table 8: Performance comparison of ES and `guided` for optimal state set selection on large-scale domains. Results are reported only on training datasets because the `guided` heuristic is defined with respect to the training dataset, and our comparison focuses on matching the settings for both methods. Although the training computation of ES is fixed, accurately evaluating its performance on large datasets remains costly, and small values of $K \times M$ yield poor results. Even with $K = 10, M = 100$ (denoted as ES 1000), ES does not consistently outperform `guided`. Scores are reported as mean $\pm$ standard error.

Domains	**Percentage Feedback**	Reduced Brute-force	ES 1000	`guided`
Breakout	0.15	17.75	17.75 $\pm$ 0.85	7.13 $\pm$ 0.11
	0.29		17.75 $\pm$ 0.40	14.12 $\pm$ 0.34
	0.44		17.66 $\pm$ 0.89	17.39 $\pm$ 0.29
	0.58		17.32 $\pm$ 1.05	17.60 $\pm$ 0.33
	0.73		17.46 $\pm$ 1.08	16.17 $\pm$ 0.35
	0.87		17.40 $\pm$ 1.43	17.06 $\pm$ 0.37
Freeway	0.16	58.28	43.44 $\pm$ 1.41	42.31 $\pm$ 0.25
	0.32		55.82 $\pm$ 0.93	54.01 $\pm$ 0.21
	0.48		58.28 $\pm$ 0.48	58.02 $\pm$ 0.20
	0.64		58.28 $\pm$ 0.46	58.28 $\pm$ 0.20
	0.80		58.28 $\pm$ 0.81	58.28 $\pm$ 0.15
	0.96		58.28 $\pm$ 0.45	58.28 $\pm$ 0.24
Seaquest	0.16	34.99	1.42 $\pm$ 0.26	7.30 $\pm$ 0.23
	0.32		9.16 $\pm$ 1.09	14.35 $\pm$ 0.47
	0.48		18.80 $\pm$ 2.25	19.77 $\pm$ 0.71
	0.64		23.58 $\pm$ 3.00	23.46 $\pm$ 0.80
	0.80		24.99 $\pm$ 3.44	23.79 $\pm$ 0.88
	0.96		25.48 $\pm$ 3.31	27.17 $\pm$ 1.04
Asterix	0.15	35.16	4.88 $\pm$ 0.74	7.38 $\pm$ 0.34
	0.30		9.06 $\pm$ 1.10	16.21 $\pm$ 0.65
	0.45		22.36 $\pm$ 2.45	24.00 $\pm$ 0.84
	0.60		28.92 $\pm$ 2.52	30.19 $\pm$ 0.91
	0.75		32.28 $\pm$ 3.03	30.71 $\pm$ 0.94
	0.90		35.16 $\pm$ 2.96	34.94 $\pm$ 1.01

D.5 Additional Pattern Analysis

In frozenLake and twoRooms-Trap, trap states can prematurely terminate episodes, leading to optimal sets focusing on avoiding trap states as well as reaching the goal. In cliffwalk, the large penalty for falling into the cliff causes optimal sets to include off-path states adjacent to the cliff, effectively constraining the agent's behavior. These effects are accentuated by the reward imputation strategy in UDS (Yu et al., 2022), which assumes unlabeled states have zero reward. Further ablation with alternative settings (e.g., Q-truncated) is shown in Appendix D.6.

To better understand the effectiveness of heuristic strategies in Breakout, we further analyze the state sets selected by `visitation` and `uniform` methods. As shown in Figure 3, `visitation` consistently outperforms `uniform` across all budget levels. To investigate this, we sampled 100 state sets from each strategy and calculated the cumulative reward present within the selected states.

Table 9 shows the average sum of rewards across these samples at varying feedback levels. The results indicate that state sets selected by `visitation` heuristics consistently contain a higher concentration of high-reward states compared to `uniform`. In Breakout, high-reward states often correspond to frames where the paddle is well-aligned with the ball to prevent it from being lost, which yields a reward of 1. The `visitation` heuristic is biased toward such frequently encountered high-value configurations during data collection, whereas `uniform` sampling provides more dispersed but less reward-focused coverage.

This quantitative observation directly supports the qualitative interpretation of the performance gap seen in Figure 3: `visitation`'s tendency to prioritize paddle-ball alignment states leads to a higher sum of rewards in the labeled dataset and therefore facilitates better value propagation during offline RL training.

Table 9: Sum of rewards in the state sets selected by `visitation` and `uniform` heuristics on Breakout. At each feedback level, we sample 100 state sets and report the mean (± standard error) of total rewards present in the selected states. Higher values for `visitation` indicate its stronger tendency to select high-reward (paddle-ball alignment) states.

Percentage Feedback	visitation	uniform
0.146	60936.980 ± 49.963	10039.340 ± 368.025
0.291	65967.740 ± 15.982	20394.690 ± 514.998
0.437	67718.620 ± 9.163	30736.510 ± 609.436
0.583	68640.250 ± 4.266	39807.620 ± 668.318
0.728	69182.230 ± 2.760	51333.890 ± 522.632
0.874	69522.340 ± 1.531	61671.510 ± 347.499

D.6 Ablation Study: A Variant of `Alg`

As illustrated in Figure 2, the core of this work is to propose and compare different reward selection strategies, which should be applicable to any `Alg`. While our main results focus on using UDS, in this section we apply the same selection strategies to an alternative `Alg` we propose.

D.6.1 Adapted Q-Learning

We use Q-learning—a value-based algorithm variants of which are widely used in offline settings (Levine et al., 2020; Kostrikov et al., 2021)—for policy updates in `Alg`. However, missing reward labels for some samples in `RLLF` pose a challenge: *how should the policy be updated when samples without rewards are encountered?* While assumptions might be made to facilitate modeling of unknown rewards, those reward estimates may be arbitrarily incorrect, especially in discrete domains.

Consequently, for states where rewards are unavailable (i.e., $s \notin \mathcal{S}_{[B]}$), we make no assumptions and treat the reward as being *undefined*. As a result, this algorithm sets unknown Q-values to zero, in contrast the UDS algorithm sets unknown reward values to zero. This approach aligns with the principle of *pessimism* in offline RL, which ensures that potentially erroneous value estimates from *unseen* data are not used to update values of *seen* data—a strategy whose benefits are widely studied (Jin et al., 2021; Xie et al., 2021). To accommodate undefined rewards, we modify the vanilla Q-learning update rule as follows:

$$Q(s,a) \longleftarrow \begin{cases} Q(s,a) + \alpha\left(r(s,a) + \gamma * \max_{a'} Q(s',a') - Q(s,a)\right), & s \in \mathcal{S}_{[B]} \ \& \ s' \in \mathcal{S}_{[B]} \\ \alpha\, r(s,a), & s \in \mathcal{S}_{[B]} \ \& \ s' \notin \mathcal{S}_{[B]} \\ \underbrace{\text{undefined}}_{=0}, & s \notin \mathcal{S}_{[B]} \end{cases} \tag{5}$$

For $B = |\mathcal{S}|$, i.e., when all rewards are known for all states, this reduces to the standard Q-learning update rule (Sutton and Barto, 2018). For $B < |\mathcal{S}|$, this update rule yields a *truncated* estimate of the standard Q-values, with a corresponding *truncated* Bellman operator. To distinguish these Q-values from the standard definition, we use $\tilde{Q}$ to denote Q-values estimated from the update rule in Equation 5.

The values $\tilde{Q}(s,a)$ are only defined for states $s \in \mathcal{S}_{[B]}$. Consequently, a greedy policy derived from the truncated Q-values can only be defined for $s \in \mathcal{S}_{[B]}$. For states $s \notin \mathcal{S}_{[B]}$, there is no reward feedback is available and $\tilde{Q}(s,a)$ is undefined, and we cannot evaluate the varying effects of actions in those states. In the absence of any evaluative signal for actions, we default to the data collecting policy π_D at those states.

$$\pi_{[B]} = \pi_{[\mathcal{S}_{[B]}]} = \begin{cases} \arg\max_a \tilde{Q}(s,a), & s \in \mathcal{S}_{[B]} \\ \pi_D, & s \notin \mathcal{S}_{[B]} \end{cases} \tag{6}$$

This update scheme is denoted by `Alg`, and the policy output by $\texttt{Alg}(\mathcal{D}, \mathcal{S}_{[B]})$ is denoted by $\pi_{[B]}$, or equivalently, $\pi_{[\mathcal{S}_{[B]}]}$ when emphasizing the dependence on $\mathcal{S}_{[B]}$. Policy updates only occur at states $s \in \mathcal{S}_{[B]}$. Selecting a set of states to label with reward amounts determines states at which the policy gets updated—potentially to differ from the data-collecting policy—and the strategy for selecting these states $\mathcal{Q}^{(B)}$ to optimize Equation (1) is the focus of the following sections.

D.6.2 Performance of Heuristics Selection Strategy

We evaluate `guided`, `visitation`, and `uniform` selection strategies under Adaptive Q-Learning on small domains as shown in the Table 10. The trends largely align with the findings in the main text and remain consistent with those observed under UDS. In domains such as Graph, Tree, CliffWalk, and TwoRooms-Trap, where the optimal policy follows a narrow set of trajectories, path-following methods (`guided` and `visitation`) perform best. In contrast, TwoRooms and FrozenLake contain multiple viable paths to the goal, making broader state coverage more advantageous; here, `uniform` selection achieves superior results. Adaptive Q-Learning confirms the strong dependence of heuristic effectiveness on domain characteristics, including transition determinism, reward sparsity, and bottleneck structures (as discussed in Section 4.1).

Table 10: Comparison of `guided`, `visitation`, and `uniform` heuristic selection strategies on small-scale domains. For each domain, the table presents the mean policy return (± standard error) and the corresponding optimality gap (in parentheses) across five percentage feedback levels.

Domains	Percentage Feedback	guided	visitation	uniform
Graph	0.1	**4.477 ± 0.040 (0.860)**	4.397 ± 0.036 (0.940)	4.171 ± 0.040 (1.166)
	0.3	**5.616 ± 0.069 (1.549)**	5.480 ± 0.068 (1.685)	5.048 ± 0.062 (2.117)
	0.5	**6.604 ± 0.098 (1.396)**	6.385 ± 0.101 (1.615)	5.697 ± 0.081 (2.303)
	0.7	**7.502 ± 0.086 (0.498)**	7.229 ± 0.093 (0.771)	6.019 ± 0.127 (1.981)
	0.9	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**	**8.000 ± 0.000 (0.000)**
Tree	0.1	**8.300 ± 0.144 (3.424)**	8.059 ± 0.116 (3.665)	6.753 ± 0.076 (4.971)
	0.3	**13.317 ± 0.337 (3.608)**	12.126 ± 0.238 (4.798)	8.484 ± 0.134 (8.440)
	0.5	**16.120 ± 0.183 (1.340)**	14.917 ± 0.277 (2.543)	10.445 ± 0.240 (7.014)
	0.7	**17.354 ± 0.041 (0.269)**	16.870 ± 0.151 (0.753)	11.637 ± 0.343 (5.985)
	0.9	**17.689 ± 0.012 (0.030)**	17.675 ± 0.012 (0.016)	16.280 ± 0.292 (1.379)
CliffWalk	0.1	**−414.059 ± 7.923 (171.814)**	**−414.059 ± 7.923 (171.814)**	−488.198 ± 6.642 (245.953)
	0.3	**−236.441 ± 18.131 (136.441)**	−237.081 ± 18.171 (137.081)	−433.176 ± 15.181 (333.176)
	0.5	−155.088 ± 13.893 (55.088)	**−154.042 ± 13.888 (54.042)**	−409.481 ± 20.146 (309.481)
	0.7	−123.490 ± 7.651 (92.459)	**−100.437 ± 0.881 (69.406)**	−378.334 ± 24.350 (347.302)
	0.9	−146.676 ± 11.375 (132.023)	**−107.590 ± 5.313 (92.937)**	−341.785 ± 27.414 (327.131)
FrozenLake	0.1	0.024 ± 0.000 (0.010)	**0.024 ± 0.000 (0.010)**	0.024 ± 0.000 (0.010)
	0.3	0.024 ± 0.000 (0.048)	0.024 ± 0.000 (0.048)	**0.025 ± 0.001 (0.047)**
	0.5	0.024 ± 0.000 (0.222)	0.023 ± 0.000 (0.223)	**0.027 ± 0.001 (0.218)**
	0.7	0.073 ± 0.015 (0.595)	0.036 ± 0.007 (0.631)	**0.098 ± 0.014 (0.569)**
	0.9	**0.374 ± 0.030 (0.336)**	0.267 ± 0.025 (0.443)	0.368 ± 0.026 (0.341)
TwoRooms	0.1	0.025 ± 0.001 (0.289)	0.025 ± 0.001 (0.289)	**0.030 ± 0.001 (0.283)**
	0.3	0.013 ± 0.001 (0.939)	0.012 ± 0.001 (0.939)	**0.033 ± 0.003 (0.919)**
	0.5	0.007 ± 0.000 (0.992)	0.008 ± 0.000 (0.992)	**0.043 ± 0.005 (0.956)**
	0.7	0.159 ± 0.035 (0.841)	0.085 ± 0.027 (0.915)	**0.230 ± 0.034 (0.770)**
	0.9	0.721 ± 0.044 (0.279)	**0.761 ± 0.043 (0.239)**	0.720 ± 0.042 (0.280)
TwoRooms-Trap	0.1	**−55.947 ± 0.920 (32.444)**	−57.720 ± 0.628 (34.217)	−62.899 ± 0.487 (39.396)
	0.3	**−41.188 ± 1.123 (39.950)**	−44.392 ± 0.832 (43.154)	−53.528 ± 0.692 (52.290)
	0.5	**−14.334 ± 0.837 (14.872)**	−21.868 ± 0.894 (22.406)	−40.030 ± 0.837 (40.568)
	0.7	**−0.178 ± 0.057 (1.176)**	−1.001 ± 0.332 (1.999)	−26.138 ± 0.966 (27.136)
	0.9	**1.000 ± 0.000 (0.000)**	**1.000 ± 0.000 (0.000)**	−4.577 ± 0.689 (5.577)

D.6.3 Performance of Optimal Selection Strategy

We evaluate `brute-force`, `sequential-greedy`, `ES 200`, and `ES 50` under Adaptive Q-Learning with the same setting as in the main text shown in Table 11. The findings closely mirror those observed with UDS. `Sequential-greedy` consistently matches the performance of `brute-force`, validating its effectiveness as a scalable approximation to the true optimal state set. `ES 200` reliably outperforms `ES 50`, and both evolutionary variants generally exceed the performance of `guided` selection at moderate to high budgets. These results reaffirm the relative ordering and conclusions reported in the main text, demonstrating that the effectiveness of optimized selection strategies remains stable across different policy learning algorithms.

Table 11: Performance comparison of brute-force, sequential greedy, and Evolutionary Strategy (ES) on small-scale domains. Results are reported on training datasets, with test performance shown in parentheses (e.g., train score (test score)). Test scores are reported as mean $\pm$ standard error across five test datasets. ES 200 corresponds to $K = 10, M = 20$ and ES 50 to $K = 10, M = 5$.

Domains	Percentage Feedback	brute-force	sequential-greedy	ES 200	ES 50	guided
Graph	0.1	$5.337(3.032 \pm 0.213)$	$5.337(3.032 \pm 0.213)$	$5.308(3.014 \pm 0.211)$	$4.214(1.521 \pm 0.226)$	4.477
	0.3	$7.165(6.004 \pm 0.128)$	$7.165(6.004 \pm 0.128)$	$7.157(5.994 \pm 0.124)$	$6.275(4.518 \pm 0.164)$	5.616
	0.5	$8.000(8.000 \pm 0.000)$	$8.000(8.000 \pm 0.000)$	$8.000(8.000 \pm 0.000)$	$6.589(5.256 \pm 0.115)$	6.604
	0.7	$8.000(8.000 \pm 0.000)$	$8.000(8.000 \pm 0.000)$	$8.000(8.000 \pm 0.000)$	$8.000(8.000 \pm 0.000)$	7.502
	0.9	$8.000(8.000 \pm 0.000)$	$8.000(8.000 \pm 0.000)$	$8.000(8.000 \pm 0.000)$	$8.000(8.000 \pm 0.000)$	8.000
Tree	0.1	$11.724(8.092 \pm 0.276)$	$11.724(8.092 \pm 0.276)$	$11.724(8.092 \pm 0.276)$	$9.073(4.078 \pm 0.384)$	8.300
	0.3	$16.925(16.349 \pm 0.056)$	$16.925(16.349 \pm 0.056)$	$13.282(10.283 \pm 0.238)$	$9.637(5.187 \pm 0.334)$	13.317
	0.5	$17.460(17.406 \pm 0.017)$	$17.460(17.406 \pm 0.017)$	$17.235(16.982 \pm 0.025)$	$12.656(9.909 \pm 0.184)$	16.120
	0.7	$17.623(17.627 \pm 0.006)$	$17.623(17.627 \pm 0.006)$	$17.513(17.489 \pm 0.009)$	$15.217(13.324 \pm 0.142)$	17.354
	0.9	$17.659(17.788 \pm 0.001)$	$17.659(17.788 \pm 0.001)$	$17.678(17.777 \pm 0.000)$	$17.655(17.728 \pm 0.001)$	17.689
CliffWalk	0.1	$-242.245(-231.272 \pm 6.042)$	$-242.245(-231.272 \pm 6.042)$	$-322.823(-347.828 \pm 12.005)$	$-409.384(-414.365 \pm 26.537)$	-414.059
	0.3	$-100.000(-100.000 \pm 0.000)$	$-100.000(-100.000 \pm 0.000)$	$-150.081(-150.586 \pm 4.002)$	$-320.748(-308.868 \pm 13.555)$	-236.441
	0.5	$-100.000(-100.000 \pm 0.000)$	$-100.000(-100.000 \pm 0.000)$	$-100.000(-100.000 \pm 0.000)$	$-180.969(-189.833 \pm 3.670)$	-155.088
	0.7	$-31.031(-31.142 \pm 1.045)$	$-31.031(-31.142 \pm 1.045)$	$-100.000(-100.000 \pm 0.000)$	$-186.756(-180.828 \pm 8.232)$	-123.490
	0.9	$-14.653(-14.506 \pm 0.138)$	$-14.653(-14.506 \pm 0.138)$	$-100.000(-100.000 \pm 0.000)$	$-100.000(-100.000 \pm 0.000)$	-146.676
FrozenLake	0.1	$0.034(0.032 \pm 0.001)$	$0.034(0.032 \pm 0.001)$	$0.031(0.030 \pm 0.000)$	$0.031(0.030 \pm 0.001)$	0.024
	0.3	$0.072(0.049 \pm 0.003)$	$0.072(0.049 \pm 0.003)$	$0.036(0.032 \pm 0.001)$	$0.032(0.034 \pm 0.001)$	0.024
	0.5	$0.246(0.347 \pm 0.029)$	$0.246(0.347 \pm 0.029)$	$0.067(0.057 \pm 0.004)$	$0.054(0.045 \pm 0.005)$	0.024
	0.7	$0.667(0.629 \pm 0.011)$	$0.667(0.629 \pm 0.011)$	$0.199(0.212 \pm 0.006)$	$0.196(0.223 \pm 0.008)$	0.073
	0.9	$0.710(0.688 \pm 0.013)$	$0.710(0.688 \pm 0.013)$	$0.679(0.703 \pm 0.006)$	$0.699(0.709 \pm 0.013)$	0.374
TwoRooms	0.1	$0.314(0.321 \pm 0.031)$	$0.314(0.321 \pm 0.031)$	$0.063(0.073 \pm 0.014)$	$0.038(0.046 \pm 0.009)$	0.025
	0.3	$0.952(0.952 \pm 0.005)$	$0.952(0.952 \pm 0.005)$	$0.310(0.314 \pm 0.029)$	$0.052(0.057 \pm 0.009)$	0.013
	0.5	$1.000(1.000 \pm 0.000)$	$1.000(1.000 \pm 0.000)$	$0.365(0.362 \pm 0.026)$	$0.270(0.270 \pm 0.022)$	0.007
	0.7	$1.000(1.000 \pm 0.000)$	$1.000(1.000 \pm 0.000)$	$1.000(1.000 \pm 0.000)$	$0.999(1.000 \pm 0.000)$	0.159
	0.9	$1.000(1.000 \pm 0.000)$	$1.000(1.000 \pm 0.000)$	$1.000(1.000 \pm 0.000)$	$1.000(1.000 \pm 0.000)$	0.721
TwoRooms-Trap	0.1	$-23.503(-23.047 \pm 0.319)$	$-23.503(-23.047 \pm 0.319)$	$-31.646(-32.431 \pm 0.736)$	$-53.449(-53.509 \pm 0.204)$	-55.947
	0.3	$-1.238(-1.243 \pm 0.017)$	$-1.238(-1.243 \pm 0.017)$	$-11.259(-10.935 \pm 0.174)$	$-35.621(-35.996 \pm 0.556)$	-41.188
	0.5	$0.538(0.540 \pm 0.016)$	$0.538(0.540 \pm 0.016)$	$-0.845(-0.793 \pm 0.030)$	$-17.590(-17.505 \pm 0.139)$	-14.334
	0.7	$0.998(0.998 \pm 0.000)$	$0.998(0.998 \pm 0.000)$	$-0.233(-0.258 \pm 0.024)$	$-14.739(-14.826 \pm 0.245)$	-0.178
	0.9	$1.000(1.000 \pm 0.000)$	$1.000(1.000 \pm 0.000)$	$1.000(1.000 \pm 0.000)$	$0.862(0.845 \pm 0.010)$	1.000