FAST REWARD EVALUATION IN REINFORCEMENT LEARNING VIA SIMULATION OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) plays an increasingly critical role in modern artificial intelligence, including decision-making tasks and large language model (LLM) training. In RL, the reward function determines what behavior the policy is optimized to learn. In practice, different candidate rewards can lead to substantially different policy performance, making *reward evaluation* crucial for selecting rewards that effectively guide policy optimization. However, evaluating candidate rewards often requires training a separate policy for each candidate, resulting in substantial computational cost and making reward evaluation a major bottleneck. To address this challenge, we reframe reward evaluation as a *simulation optimization* problem in operations research, which studies efficient selection under limited budgets. Under this formulation, reward evaluation is cast as a sequential budget allocation problem. We then propose *Fast Reward Evaluation* by adapting Optimal Computing Budget Allocation (OCBA), a representative simulation optimization method, to guide the allocation of training budget across candidate reward functions. Empirical results on sparse-reward RL, automatic reward generation, and LLM reasoning tasks demonstrate that evaluation efficiency is significantly improved, and that high-quality reward functions are more reliably identified under the same computational budget.

1 INTRODUCTION

Reinforcement learning (RL) has been widely applied to sequential decision-making tasks, including robotic control (Polydoros & Nalpantidis, 2017; Singh et al., 2022), game playing (Vinyals et al., 2019; Ye et al., 2020), and resource allocation (Li et al., 2021a; Sheng et al., 2022). More recently, RL has also become central to LLM reasoning and feedback-driven training (Kumar et al., 2025; Shao et al., 2024; Guo et al., 2025). Across these applications, reward design remains a fundamental challenge (Dann et al., 2023; Hua et al., 2025; Yu et al., 2025): the reward function directly shapes training efficiency and policy behavior, while poorly specified rewards may induce unintended optimization or reward hacking (Skalse et al., 2022; Miao et al., 2024).

To reduce manual reward engineering, prior work has explored inverse reinforcement learning (Ng & Russell, 2000; Arora & Doshi, 2021; Zakka et al., 2022), reward transfer (Brys et al., 2015; Hua et al., 2021; 2025), and, more recently, LLM-based reward generation (Kwon et al., 2023; Xie et al., 2024; Ma et al., 2024; Li et al., 2025b). While these methods enable large-scale candidate generation, they also make *reward evaluation* increasingly costly. Evaluating a candidate requires downstream policy training, yielding a bi-level optimization problem in which the outer loop searches over rewards and the inner loop trains policies (Stadie et al., 2020; Chentanez et al., 2004). As candidate sets grow, exhaustively training every policy becomes computationally prohibitive, making evaluation a primary bottleneck in automated reward design (Gleave et al., 2021; Frick et al., 2025).

Some methods attempt to avoid inner-loop training by directly comparing reward functions, for example through reward distance (Gleave et al., 2021; Wulfe et al., 2022; Skalse et al., 2024). However, these approaches require sufficiently broad state–action coverage, which is difficult to obtain in online RL, particularly under sparse rewards (Chentanez et al., 2004; Andrychowicz et al., 2017; Pathak et al., 2017). Thus, actual policy training remains necessary for reliable reward evaluation. This raises a practical question: *under a fixed training budget, how should training steps be allocated across reward candidates to efficiently identify the best one?*

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

We view this problem as stochastic resource allocation. Candidate rewards form a finite set of alternatives, while inner-loop policy training provides costly and noisy evaluations, naturally connecting reward evaluation to *simulation optimization* (Fu, 2014). Unlike standard bandit settings (Bubeck et al., 2009; Audibert & Bubeck, 2010), our goal is terminal selection under a fixed compute budget, and additional computation advances the policy training process rather than repeatedly evaluating a fixed alternative. More specifically, this formulation corresponds to a *ranking-and-selection* (R&S) problem (Panchapakesan, 1992; Chen et al., 2014; Hong et al., 2021): each candidate reward is an alternative, its quality is evaluated through policy training under the external true reward, and the available training compute constitutes the total evaluation budget.

Motivated by this observation, we propose *Fast Reward Evaluation*, a training-budget allocation framework inspired by Optimal Computing Budget Allocation (OCBA) (Chen et al., 2000; Boesel et al., 2003; Chen et al., 2008; Chen & Lee, 2011). Classical OCBA allocates more evaluations to informative alternatives, particularly those close to the current best or associated with high uncertainty. However, it assumes static alternatives, whereas RL candidates evolve continuously during training. To address this mismatch, we introduce *Reward-adaptive OCBA*, which extends OCBA to learning-dependent evaluations by leveraging training trajectories. Each candidate's late-stage trajectory is summarized by a lightweight stochastic surrogate capturing its current performance, learning progress, and residual uncertainty. These statistics are then used to dynamically allocate more training steps to promising or insufficiently resolved candidates, while reducing computation on unlikely contenders. The resulting allocation rule is theoretically justified for identifying high-quality reward candidates under a limited budget.

We evaluate the method on classical RL benchmarks, automatic reward-generation tasks, and LLM reasoning settings. Across these settings, it improves evaluation efficiency and enables more reliable reward selection under the same training budget. Our contributions are: **(1). A simulation-optimization perspective on reward evaluation.** We formulate RL reward evaluation as budget allocation over online training trajectories, connecting reward selection to simulation optimization. **(2). A fast reward evaluation framework with theoretical guarantees.** We propose *Fast Reward Evaluation*, extending OCBA to RL through a lightweight stochastic surrogate, and establish asymptotic optimality in probability of correct selection. **(3). Empirical validation across RL and LLM reasoning.** Experiments on sparse-reward RL, automatic reward generation, and mathematical reasoning show improved evaluation efficiency and reward selection under the same training budget.

2 Background

2.1 Reward Evaluation in Reinforcement Learning

We consider a set of candidate reward-shaping functions $\mathcal{F} = \{F_1, \ldots, F_K\}$, where each shaping function induces an implicit reward model. Each candidate defines a modified reward:

$$R_k(s, a) = r(s, a) + F_k(s, a). \tag{1}$$

For each implicit reward model, we define the corresponding optimal policy:

$$\pi_k^* \in \arg\max_{\pi \in \Pi} \mathbb{E}_{\tau \sim \pi} \left[\sum_{t=0}^{T} R_k(s^t, a^t) \right]. \tag{2}$$

Instead of explicitly selecting a reward function, we formulate the reward evaluation problem as selecting among the induced optimal policies. The final objective is:

$$\pi^\star = \arg \max_{k \in \{1, \ldots, K\}} \mathbb{E}_{\tau \sim \pi_k^*} \left[\sum_{t=0}^{T} r(s^t, a^t) \right]. \tag{3}$$

This formulation highlights the computational challenge of reward evaluation. The quality of a candidate reward can only be reliably assessed by training a policy under it and evaluating the resulting performance, which is particularly costly in sparse-reward settings. As a result, the key challenge is how to allocate a limited interaction budget across candidates effectively. Under a fixed environment interaction budget B, reward selection becomes a sequential resource allocation problem, where b_k denotes the number of interaction steps allocated to reward R_k, with $\sum_{k=1}^{K} b_k \leq B$.

2.2 Optimal Computing Budget Allocation

The current problem formulation reduces reward selection to an R&S (Panchapakesan, 1992; Chen et al., 2014; Hong et al., 2021) problem under a limited budget in simulation optimization. In the standard R&S setting, one is given K competing candidates, where candidate i has an unknown mean performance μ_i. Running one simulation replication for candidate i yields a noisy observation $X_{i,\ell} \sim \mathcal{D}_i$, with $\mathbb{E}[X_{i,\ell}] = \mu_i$ and $\mathrm{Var}(X_{i,\ell}) = \sigma_i^2$. Given a total simulation budget N, the task is to choose $n_1, \ldots, n_K$, subject to $\sum_{i=1}^{K} n_i = N$, so as to maximize the probability of correct selection (PCS):

$$i^\star = \arg\max_{1 \le i \le K} \mu_i. \tag{4}$$

OCBA (Boesel et al., 2003; Chen et al., 2008; Chen & Lee, 2011) is one of the most influential methods for this objective. Rather than allocating samples uniformly, OCBA allocates more samples to candidates that are both competitive (close to the current best) and uncertain (high variance), while reducing allocation to clearly inferior ones.

Let b denote the current best candidate according to the sample mean, and define the performance gap $\Delta_{i,b} = \bar{X}_b - \bar{X}_i$ for $i \neq b$. Under standard asymptotic analysis, OCBA (Chen et al., 2000) yields the allocation rule:

$$\frac{n_i}{n_j} \approx \frac{\sigma_i^2/\Delta_{i,b}^2}{\sigma_j^2/\Delta_{j,b}^2}, \qquad i, j \neq b, \tag{5}$$

with a companion allocation for the incumbent best. This rule allocates more samples to candidates with higher variance or smaller performance gaps, as they are more likely to affect the final selection.

This result provides the key conceptual foundation for our work: *budget should be allocated according to both performance and uncertainty*. However, these assumptions break down in RL reward evaluation: (1) budget is spent on sequential environment interaction within training rather than independent replications; (2) policy updates induce distribution shift and non-stationary return estimates; and (3) performance feedback is generated through an evolving learning process, making evaluations sequentially correlated. Thus, directly applying OCBA is inappropriate.

Nevertheless, this departure from classical OCBA assumptions does not preclude adapting OCBA to RL. Prior work has shown that Gaussian-process models provide a principled framework for representing uncertainty in RL, including policy evaluation and optimization (Engel et al., 2005; Hennig, 2011; Rao et al., 2020). In particular, (Hennig, 2011) analyzes RL learning dynamics under Gaussian-process beliefs, suggesting that complex and correlated training processes can be summarized through tractable predictive means and uncertainties. This provides a principled basis for adapting OCBA-style allocation to RL.

3 Fast Reward Evaluation

We now present the main pipeline of *Fast Reward Evaluation*, illustrated in Figure 1. Since direct analysis of RL training dynamics is difficult, we introduce a Gaussian learning-curve surrogate that recasts the evolving policy trajectory as a tractable random process. Its mean and variance summarize each reward candidate's estimated performance and uncertainty for sequential allocation. Building on this surrogate, we develop a reward-adapted OCBA rule that allocates future training budget toward competitive or uncertain candidates and establish its theoretical guarantee.

3.1 Gaussian Learning-Curve Surrogate

In this section, we derive a lightweight stochastic surrogate model for late-stage reward evaluation. Rather than modeling the full high-dimensional training process exactly, we seek a macroscopic approximation that captures the three quantities most relevant for sequential allocation decisions: the asymptotic performance plateau, the effective learning speed, and the residual uncertainty. Appendix H provides an empirical diagnostic of the Gaussian surrogate on the observed learning trajectory.

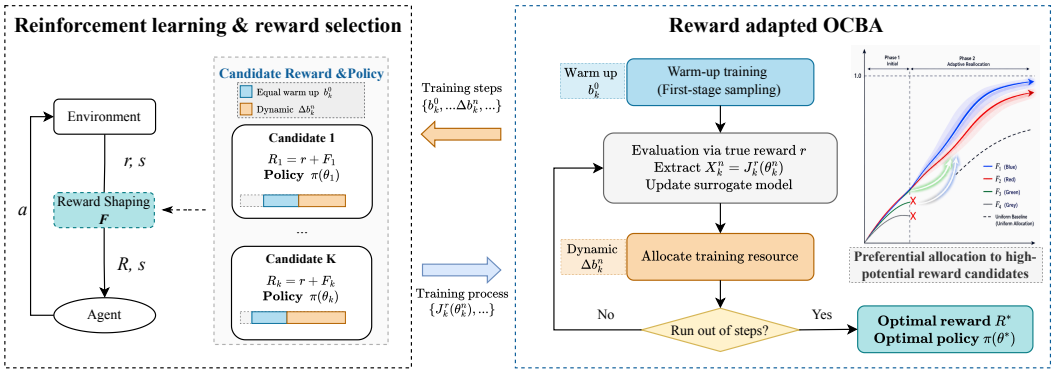


Figure 1: Framework of *Fast Reward Evaluation*. Candidate rewards are trained online under a shared budget, with surrogate statistics updated from true-reward evaluations to estimate each candidate's value and uncertainty. Fast Reward Evaluation instantiates a *reinforcement-learning-adapted OCBA* that dynamically plans training-resource allocation from the real-time training state, preferentially allocating budget to high-potential reward candidates for efficient reward selection.

(1) The Optimization and Evaluation Processes. For each candidate reward k, training is performed under the shaped reward $R_k = r + F_k$, while candidate selection is evaluated based on the *original* reward r. We define their expected discounted returns for given policy parameters θ:

$$J_k^R(\theta) := \mathbb{E}_{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t R_k(s_t, a_t)\right], \qquad J_k^r(\theta) := \mathbb{E}_{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t r(s_t, a_t)\right]. \tag{6}$$

Because the Bellman evaluation operators for R_k, r, and F_k are linear, the Banach fixed-point theorem guarantees unique fixed points, and the additivity of returns carries over to them. Taking expectations under the initial-state distribution then gives the return-level identity:

$$J_k^r(\theta) = J_k^R(\theta) - J_k^F(\theta), \qquad J_k^F(\theta) := \mathbb{E}_{\pi_\theta}\left[\sum_{t\geq 0}\gamma^t F_k(s_t, a_t)\right]. \tag{7}$$

This identity highlights the central mismatch of reward-shaping: policy optimization is intrinsically driven by J_k^R, whereas final performance selection strictly depends on J_k^r. To make this mismatch explicit in the dynamics, we introduce two coupled processes. The *optimization-state process* is the parameter error $Y_k^n := \theta_k^n - \theta_k^\star$, where θ_k^n is the policy parameter at the n-th budget checkpoint under R_k and $\theta_k^\star$ is the induced local optimum, tracking the high-dimensional gradient updates under J_k^R. The *evaluation-observation process* is the scalar learning curve $X_k^n := J_k^r(\theta_k^n)$, which is the true-reward score used for candidate comparison.

(2) Stability Assumptions. To formalize the ensuing learning trajectory, we consider the late phase of training and use the following assumptions (further discussion is provided in the Appendix B).

Assumption 1 (Stable Policy Around a Local Optimum)**.** *For each candidate k, in the late stage of training ($n \geq n_{0,k}$), the policy parameters θ_k^n enter and remain within a stable local neighborhood $B(\theta_k^\star, r_k)$ almost surely, where $\theta_k^\star$ is a local optimum of the population objective satisfying $\nabla J_k^R(\theta_k^\star) = 0$. Moreover, for a sufficiently small learning rate α, the local dynamics satisfy*

$$\nabla J_k^R(\theta_k^\star) = 0, \qquad \rho\big(I + \alpha\nabla^2 J_k^R(\theta_k^\star)\big) < 1, \tag{8}$$

where $\rho(\cdot)$ denotes the spectral radius (maximum absolute eigenvalue).

Assumption 2 (Constant Variance of Gradient Noise)**.** *During late-stage training ($n \geq n_{0,k}$), let $\mathcal{F}_n$ denote the filtration generated by the optimization history up to step n. The policy-gradient noise $\xi_k^{n+1} := \hat{g}_k^n - \nabla J_k^R(\theta_k^n)$ forms a martingale-difference sequence with respect to $\{\mathcal{F}_n\}$. Moreover, for some $\delta > 0$ and constant matrix $\Sigma_k \succeq 0$,*

$$\mathbb{E}[\xi_k^{n+1} \mid \mathcal{F}_n] = 0, \quad \mathbb{E}[\xi_k^{n+1}(\xi_k^{n+1})^\top \mid \mathcal{F}_n] \approx \Sigma_k, \quad \mathbb{E}[\|\xi_k^{n+1}\|^{2+\delta} \mid \mathcal{F}_n] \leq C_k. \tag{9}$$

(3) Deriving the Gaussian Surrogate. The stochastic policy-gradient update is $\theta_k^{n+1} = \theta_k^n + \alpha \hat{g}_k^n$. Let $H_k := \nabla^2 J_k^R(\theta_k^\star)$ denote the local Hessian and $M_k := I + \alpha H_k$ the effective update operator, where I is the identity matrix. Working in the error coordinates $Y_k^n = \theta_k^n - \theta_k^\star$, a first-order Taylor expansion around $\theta_k^\star$ yields the linearized local dynamics:

$$Y_k^{n+1} \approx M_k Y_k^n + W_k^{n+1}, \quad W_k^{n+1} := \alpha \xi_k^{n+1} + \mathcal{O}(\|Y_k^n\|^2). \tag{10}$$

By expanding the scalar evaluation process X_k^n around $\theta_k^\star$, we obtain $X_k^n \approx J_k^r(\theta_k^\star) + \nabla J_k^r(\theta_k^\star)^\top Y_k^n$. Crucially, because $\theta_k^\star$ is a stationary point of the shaped objective, $\nabla J_k^R(\theta_k^\star) = 0$. By applying the return-level identity from Eq. (7), the local sensitivity vector simplifies exactly to the negative gradient of the shaping function:

$$c_k := \nabla J_k^r(\theta_k^\star) = \nabla J_k^R(\theta_k^\star) - \nabla J_k^F(\theta_k^\star) = -\nabla J_k^F(\theta_k^\star). \tag{11}$$

Combining this projection with Eq. (10) yields the induced evaluation-process recursion:

$$X_k^{n+1} \approx J_k^r(\theta_k^\star) + c_k^\top M_k Y_k^n + c_k^\top W_k^{n+1}. \tag{12}$$

However, Y_k^n is high-dimensional and latent. Recent studies (Cai et al., 2025; Schneider et al., 2024; Todorov & Sabatelli, 2026) of RL optimization dynamics suggest that late-stage updates are often concentrated in a low-dimensional, and sometimes nearly rank-one, subspace that captures most of the effective performance improvement. Motivated by this structural simplification, we approximate the projected evaluation dynamics with a dominant one-dimensional contraction. We formalize this approximation in the following assumption.

Assumption 3 (Late-stage Projected Contraction)**.** *In the late stage of training, the projected evaluation dynamics approximately contract at a stable rate. Specifically, for the evaluation direction c_k, there exists $\rho_k \in (0, 1)$ such that*

$$c_k^\top M_k Y_k^n = \rho_k c_k^\top Y_k^n + \epsilon_k^n,$$

where ϵ_k^n is a sufficiently small approximation error term.

Theorem 1 (Late-stage Scalar Approximation of Evaluation Dynamics)**.** *Under Assumptions 1–3, let $\mu_k := J_k^r(\theta_k^\star)$. Then the evaluation trajectory X_k^n is well approximated by a simple first-order scalar recursion:*

$$X_k^{n+1} \approx (1 - \rho_k)\mu_k + \rho_k X_k^n + \eta_k^{n+1}, \tag{13}$$

where $\rho_k \in (0, 1)$ is the dominant contraction factor and $\eta_k^{n+1} := c_k^\top W_k^{n+1}$ is the projected innovation at step $n + 1$. (Proof in Appendix D.)

Remark 1 (Optimization time-scale)**.** The dominant contraction factor ρ_k acts as a discrete decay factor that bottlenecks the late-stage convergence speed. Physically, μ_k indicates the asymptotic performance level *where* the evaluation trajectory stabilizes, while ρ_k indicates *how fast* it settles.

Theorem 1 reduces the evaluation dynamics to a first-order autoregressive process. However, the recursion still involves an innovation term η with unspecified distribution, precluding closed-form decision models such as optimal budget allocation. To support efficient online allocation, we therefore seek a parametric characterization of each candidate's marginal mean and variance over training. Under Assumption 2, the martingale-difference structure and bounded-moment conditions allow the martingale central limit theorem to render the cumulative innovation asymptotically Gaussian. This yields a Gaussian approximation to the AR(1) process, providing the tractable form required for downstream allocation.

Proposition 1 (Gaussian Surrogate Approximation)**.** *Under the scalar approximation above, the aggregated late-stage trajectory is approximately a Gaussian process:*

$$X_k^n \sim \mathcal{N}\Big(\mathbb{E}[X_k^n],\, \mathrm{Var}(X_k^n)\Big), \tag{14}$$

$$\mathbb{E}[X_k^n] = \mu_k + (X_k^0 - \mu_k)e^{-n/\tau_k}, \tag{15}$$

$$\mathrm{Var}(X_k^n) = \sigma_{k,\infty}^2 + (\sigma_{k,0}^2 - \sigma_{k,\infty}^2)e^{-2n/\tau_k}, \tag{16}$$

where $\tau_k = -1/\ln(\rho_k)$ is the continuous time constant, $\sigma_{k,0}^2 := \mathrm{Var}(X_k^0)$, and $\sigma_{k,\infty}^2$ is the stationary variance floor. (Proof in Appendix E.)

Algorithm 1 Fast Reward Evaluation under Policy-Gradient Training

Require: Candidate set $\mathcal{F} = \{F_1, \ldots, F_K\}$, total budget B, warm-start budget b_0, per-round increment Δb

```
Initialize policy parameters {θ_k}_{k=1}^K, consumed budgets {b_k}_{k=1}^K ← 0
for k = 1, ..., K do
    Train π_{θ_k} under R_k = r + F_k for b_0 steps
    Evaluate on original reward r; estimate surrogate statistics (m̂_k, v̂_k)
    b_k ← b_0
end for
while Σ_{k=1}^K b_k < B do
    Compute budget allocation {Δb_k}_{k=1}^K by Algorithm 2
    for each candidate k with Δb_k > 0 do
        Train π_{θ_k} under R_k for Δb_k additional steps
        Evaluate updated policy under original reward r
        Update surrogate statistics (m̂_k, v̂_k) and set b_k ← b_k + Δb_k
    end for
end while
return k* = arg max_k m̂_k and corresponding policy π_{θ_{k*}}
```

3.2 Fast Reward Evaluation via Reward-Adapted OCBA

In this section, we discuss the use of the Gaussian learning-curve surrogate in Proposition 1 to guide online budget allocation across candidate rewards. Building on the OCBA framework, we adapt its allocation principle to the non-stationary, trajectory-dependent setting of RL reward evaluation. For each candidate k, the method maintains a surrogate mean $\hat{m}_k$ and variance $\hat{v}_k$, directing more training to candidates that are competitive or uncertain. Our *Fast Reward Evaluation* algorithm (Algorithm 1) first trains each candidate with a warm-start budget and evaluates it under the original reward to initialize the surrogate. At each iteration, the *Reward-Adapted OCBA Rule* (Algorithm 2 in Appendix C) allocates additional budget, favoring candidates closer to the current best or with higher uncertainty. Candidates are further trained and re-evaluated, with surrogates updated iteratively until the budget is exhausted. The candidate with the highest surrogate mean is then selected.

Next, we derive the allocation ratio underlying the *Reward-Adapted OCBA Rule*. At a given allocation round, suppose candidate k has received training budget n_k. From the Gaussian learning-curve surrogate,

$$X_k^{n_k} \sim \mathcal{N}\left(m_k(n_k), v_k(n_k)\right), \tag{17}$$

$$m_k(n_k) = \mu_k + (X_k^0 - \mu_k)e^{-n_k/\tau_k}, \quad v_k(n_k) = \sigma_{k,\infty}^2 + (\sigma_{k,0}^2 - \sigma_{k,\infty}^2)e^{-2n_k/\tau_k}. \tag{18}$$

Let $b = \arg\max_k m_k(n_k)$ be the current best candidate. We choose a threshold c between b and its closest competitor, and define the distance to this selection boundary as:

$$d_b = m_b(n_b) - c, \qquad d_i = c - m_i(n_i), \quad i \neq b. \tag{19}$$

For notational simplicity, at a given allocation epoch we write $\widetilde{X}_k(N_k)$ for the Gaussian surrogate evaluation of candidate k after receiving additional allocation N_k. Following the OCBA approximation, we use a selection boundary c to approximate the probability of correctly identifying the best reward candidate (probability of correct selection, PCS) as:

$$\begin{aligned} \mathrm{PCS} &= \mathbb{P}\{\widetilde{X}_b(N_b) \geq \widetilde{X}_i(N_i),\ \forall i \neq b\} \geq \mathbb{P}\{\widetilde{X}_b(N_b) \geq c,\ \widetilde{X}_i(N_i) \leq c,\ \forall i \neq b\} \\ &\approx \prod_{k=1}^{K} \Phi\left(d_k \sqrt{\frac{N_k}{v_k(n_k)}}\right) =: \mathrm{APCS}(N_1, \ldots, N_K), \end{aligned} \tag{20}$$

where APCS denotes the approximate PCS, N_k is the additional budget allocated to candidate k, and $\Phi(\cdot)$ is the standard normal CDF.

Theorem 2 (Asymptotic Allocation Rule)**.** *Under the Gaussian learning-curve surrogate in* (18)*, consider the approximation* $\mathrm{APCS}(N_1, \ldots, N_K)$ *in* (20) *with total additional budget* $\sum_{k=1}^{K} N_k = T$.

As $T \to \infty$, the continuous allocation that asymptotically maximizes APCS *satisfies:*

$$\frac{N_i}{N_j} \sim \frac{v_i(n_i)/d_i^2}{v_j(n_j)/d_j^2}, \qquad i, j = 1, \ldots, K. \tag{21}$$

Moreover, if $n_k \to \infty$ for all k, then

$$\frac{N_i}{N_j} \to \frac{\sigma_{i,\infty}^2/(d_i^\star)^2}{\sigma_{j,\infty}^2/(d_j^\star)^2}, \tag{22}$$

where $d_k^\star$ is the limiting distance between candidate k and the selection boundary.

Theorem 2 characterizes the asymptotic allocation ratio, showing that each candidate's budget is proportional to its surrogate uncertainty and inversely proportional to its squared distance from the selection boundary. Accordingly, Algorithm 2 allocates more training to candidates that are more uncertain or closer to affecting the final selection. Moreover, under Assumption 4, the allocation weights remain nondegenerate, so every candidate receives an unbounded training budget as the total budget grows. If the surrogate mean estimates are consistent and the limiting best-reward candidate is unique, the probability of correct selection converges to one, as established in Theorem 3. The proofs are provided in Appendices F and G, respectively.

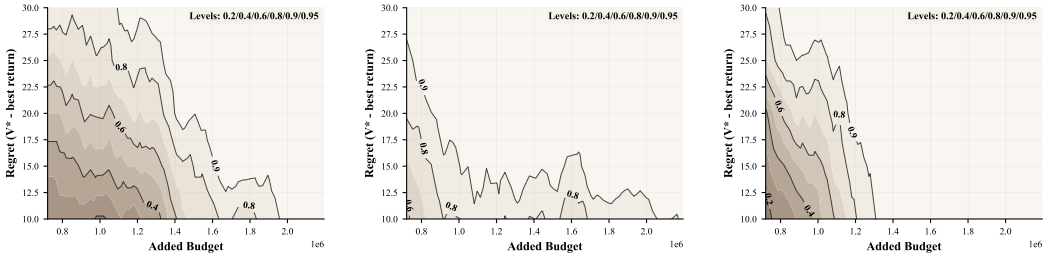
Assumption 4 (Bounded Uncertainty and Separated Candidates)**.** *The surrogate variances are bounded away from zero and infinity, i.e., $0 < \underline{v} \le \hat{v}_k \le \overline{v} < \infty$ for all $k = 1, \ldots, K$. In addition, the surrogate means are separated, i.e., $|\hat{m}_i - \hat{m}_j| \ge \underline{\Delta} > 0$ for all $i \ne j$.*

Theorem 3 (Asymptotic Consistency of Reward Selection)**.** *For Algorithm 1, suppose Assumption 4 holds. If the limiting best reward candidate is unique, then*

$$\mathrm{PCS}_B := \mathbb{P}\{\hat{k}_B = b^\star\} \to 1, \qquad B \to \infty, \tag{23}$$

where

$$b^\star = \arg\max_{1 \le k \le K} \mu_k, \qquad \hat{k}_B = \arg\max_{1 \le k \le K} \hat{m}_k(b_k(B)).$$



(a) Uniform (b) Standard OCBA (c) Reward-Adapted OCBA

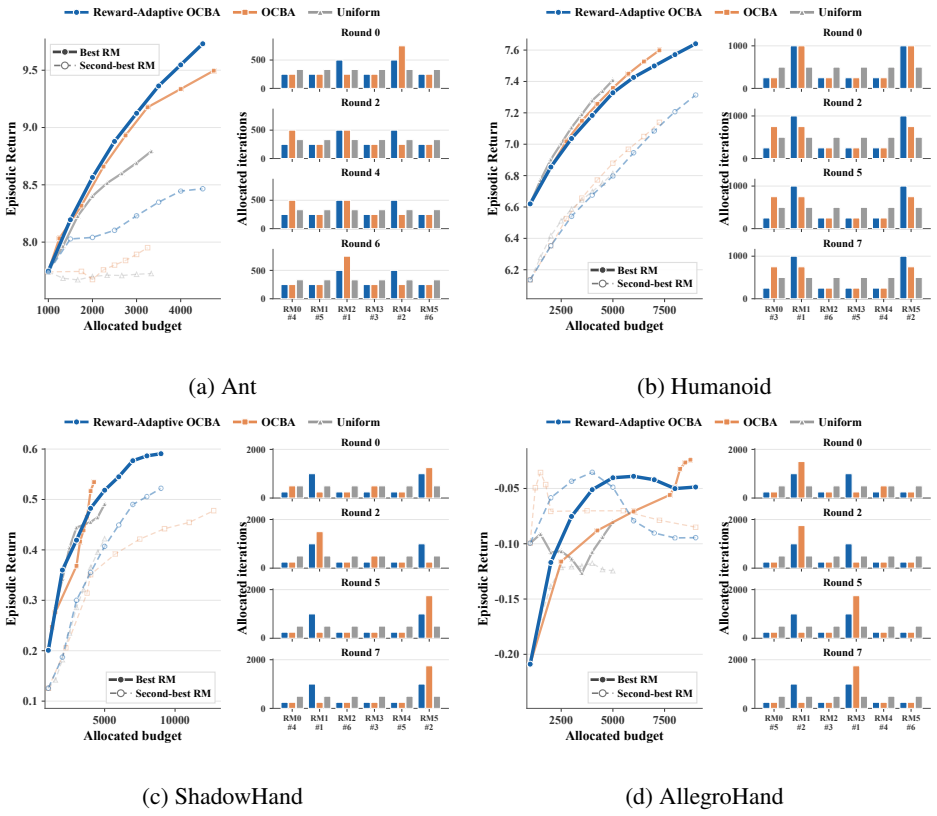
Figure 2: `MountainCar-v0` success contours: probability of reaching a target performance gap to the optimal policy within a given additional budget.

4 Experiment

The proposed method is evaluated on three types of reward-evaluation tasks to examine both generality and practical usefulness: (i) classical sparse-reward benchmarks, (ii) automatic reward generation benchmark on Eureka (Ma et al., 2024), and (iii) LLM mathematical reasoning task on GSM8K (Cobbe et al., 2021). These settings allow us to examine whether the same budget-allocation principle remains effective across diverse reward-construction pipelines and training dynamics.

4.1 Sparse-reward RL benchmark

This experiment directly validates OCBA for reward selection under a fixed budget. Specifically, we maximize the approximate probability of correct selection (Eq. (20)) using the ratio-based online allocation rule in Eq. (21), and compare it against non-adaptive baselines.



(a) Ant (b) Humanoid

(c) ShadowHand (d) AllegroHand

Figure 3: Comparison of budget allocation and reward-evaluation performance on Eureka.

We evaluate budget allocation in the classical sparse-reward environment `MountainCar-v0`. Despite its lightweight nature, the environment remains challenging due to sparse rewards, making it a suitable testbed for online training under limited budgets. Figure 2 visualizes the probability of reaching a target performance gap to the optimal policy under different computational budgets. Across target gaps and budget levels, Adaptive OCBA consistently achieves a more favorable success–cost trade-off, attaining a higher probability of approaching optimal performance under the same budget. Further experimental details and complete results are provided in Appendix I.1.

4.2 Automatic Reward Generation Benchmark

Setup. In this section, we evaluate whether *Fast Reward Evaluation* can effectively identify promising reward models and allocate training budgets among them in an automatic reward generation setting. We use Eureka to automatically generate six candidate reward functions for each task and evaluate them through policy optimization and downstream task performance. Experiments are conducted on four environments: `Ant`, `Humanoid`, `ShadowHand`, and `AllegroHand`. We compare *Uniform*, *OCBA*, and *Reward-adapted OCBA* under the same total budget. Full experimental settings are provided in Appendix I.2.

Results. Figure 3 summarizes the Eureka results. For each environment, the left panel shows the top two reward models selected by each method, while the right panel reports the stage-wise budget allocation together with the corresponding performance ranking. Across all four environments, Reward-adaptive OCBA identifies the ultimately top-ranked models after the first allocation round and maintains relatively stable preferences thereafter. In contrast, standard OCBA exhibits more frequent shifts across rounds, while Uniform allocates the budget evenly. These results indicate that, in the evaluated settings, trajectory-aware allocation can identify promising reward models at an early stage and maintain a more consistent allocation policy over subsequent rounds.

Under the same total budget, Reward-adaptive OCBA achieves the highest final best-RM return on `Ant`, `Humanoid`, and `ShadowHand`, improving over standard OCBA by 2.5%, 0.6%, and 10.5%, and over Uniform by 10.7%, 3.1%, and 20.4%, respectively. On `AllegroHand`, both adaptive methods outperform Uniform, further indicating the benefit of non-uniform allocation. These gains show that the improved allocation behavior translates into better reward selection and downstream policy performance under a fixed computational budget.

4.3 LLM Reasoning Benchmark

Setup. To evaluate the effectiveness of our method in LLM reasoning training, we conduct experiments on `GSM8K` using `Qwen2.5-1.5B-Instruct` as the policy model. We construct three reward models based on different scoring schemes for evaluating model-generated solutions, yielding reward signals with varying levels of reliability and discriminative power. Full experimental settings and hyperparameters are provided in Appendix I.3.

Results. Figure 4 shows that Reward-Adaptive OCBA achieves better overall performance than both Uniform allocation and standard OCBA under the same total training budget, with more stable optimization trajectories. Its allocation progressively shifts computation toward reward models that provide more effective training signals for the target reasoning task, whereas standard OCBA exhibits more variable preferences across rounds and Uniform allocation does not distinguish among reward models. This behavior suggests that Reward-Adaptive OCBA can better exploit differences in reward effectiveness during training, enabling the available computation to be directed toward more informative reward signals. Under the evaluated LLM reasoning setting, this adaptive allocation translates into improved computational efficiency, more stable optimization, and stronger final performance under a fixed budget.

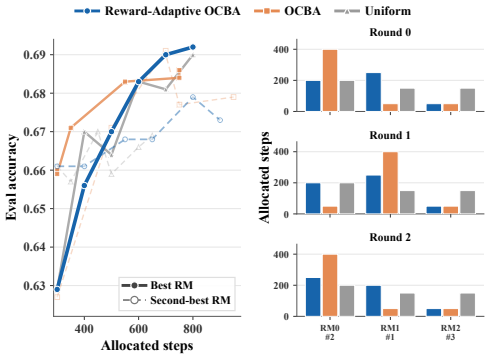


Figure 4: Experiment on GSM8K.

4.4 Ablation Study on Warm-up Budget

We further investigate the effect of the warm-up budget on adaptive allocation using `Ant` and `Humanoid`. Specifically, we compare warm-up budgets w of 500, 2,000, and 3,000 training steps while keeping all other experimental settings unchanged. The results show that a larger warm-up budget generally yields more reliable initial estimates, which in turn leads to more stable allocation decisions and learning dynamics in the subsequent adaptive stage. This suggests that sufficient initialization can help the learning process better align with the assumptions underlying our allocation strategy and improve overall performance. Detailed results are presented in Appendix I.4.

5 Conclusion

This paper reframes reward evaluation in RL as a simulation-optimization problem and proposes *Fast Reward Evaluation*, which adapts OCBA to the evolving training dynamics of RL through a lightweight Gaussian learning-curve surrogate. Experiments on sparse-reward RL benchmarks, automatic reward generation, and LLM mathematical reasoning demonstrate improved evaluation efficiency and more reliable identification of high-quality reward functions under the same computational budget. While prior work often models learning dynamics using stochastic differential equations, we adopt a lightweight Gaussian surrogate for tractable analysis. We refer readers to Appendix J for a broader discussion of limitations and future directions.

AI USE STATEMENT

In this work, we used generative AI tools for manuscript language polishing and code modification. The AI-assisted text was carefully reviewed and revised by the authors. AI-assisted code was manually inspected and tested for correctness. Generative AI was not used to generate experimental results or scientific conclusions. We take full responsibility for the final content of this work including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5055–5065, 2017.

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *COLT-23th Conference on learning theory-2010*, pp. 13–p, 2010.

Justin Boesel, Barry L Nelson, and Seong-Hee Kim. Using ranking and selection to "clean up" after simulation optimization. *Operations Research*, 51(5):814–825, 2003.

Tim Brys, Anna Harutyunyan, Matthew E Taylor, and Ann Nowé. Policy transfer using reward shaping. In *AAMAS*, pp. 181–188, 2015.

Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *International conference on Algorithmic learning theory*, pp. 23–37. Springer, 2009.

Yuchen Cai, Ding Cao, Xin Xu, Zijun Yao, Yuqing Huang, Zhenyu Tan, Benyi Zhang, Guangzhong Sun, Guiquan Liu, and Junfeng Fang. On predictability of reinforcement learning dynamics for large language models. *arXiv preprint arXiv:2510.00553*, 2025.

Chun-Hung Chen and Loo Hay Lee. *Stochastic simulation optimization: an optimal computing budget allocation*, volume 1. World scientific, 2011.

Chun-Hung Chen, Jianwu Lin, Enver Yücesan, and Stephen E Chick. Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems*, 10 (3):251–270, 2000.

Chun-Hung Chen, Donghai He, Michael Fu, and Loo Hay Lee. Efficient simulation budget allocation for selecting an optimal subset. *INFORMS Journal on Computing*, 20(4):579–595, 2008.

Chun-Hung Chen, Stephen E Chick, Loo Hay Lee, and Nugroho A Pujowidianto. Ranking and selection: Efficient simulation budget allocation. *Handbook of simulation optimization*, pp. 45–80, 2014.

Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 17, 2004.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Christoph Dann, Yishay Mansour, and Mehryar Mohri. Reinforcement learning can be more efficient with multiple rewards. In *International Conference on Machine Learning*, pp. 6948–6967. PMLR, 2023.

Yaakov Engel, Shie Mannor, and Ron Meir. Reinforcement learning with gaussian processes. In *Proceedings of the 22nd international conference on Machine learning*, pp. 201–208, 2005.

Evan Frick, Tianle Li, Connor Chen, Wei-Lin Chiang, Anastasios Nikolas Angelopoulos, Jiantao Jiao, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. How to evaluate reward models for RLHF. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cbttLtO94Q.

Michael C. Fu. *Handbook of Simulation Optimization*. Springer Publishing Company, Incorporated, 2014. ISBN 1493913832.

Adam Gleave, Michael D Dennis, Shane Legg, Stuart Russell, and Jan Leike. Quantifying differences in reward functions. In *International Conference on Learning Representations (ICLR)*, 2021.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

RISHI HAZRA, Alkis Sygkounas, Andreas Persson, Amy Loutfi, and Pedro Zuidberg Dos Martires. REvolve: Reward evolution with large language models using human feedback. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cJPUpL8mOw.

Philipp Hennig. Optimal reinforcement learning for gaussian systems. *Advances in Neural Information Processing Systems*, 24, 2011.

L Jeff Hong, Weiwei Fan, and Jun Luo. Review on ranking and selection: A new perspective. *Frontiers of Engineering Management*, 8(3):321–343, 2021.

Yujing Hu, Weixun Wang, Hangtian Jia, Yixiang Wang, Yingfeng Chen, Jianye Hao, Feng Wu, and Changjie Fan. Learning to utilize shaping rewards: A new approach of reward shaping. *Advances in Neural Information Processing Systems*, 33:15931–15941, 2020.

Yun Hua, Xiangfeng Wang, Bo Jin, Wenhao Li, Junchi Yan, Xiaofeng He, and Hongyuan Zha. Hmrl: Hyper-meta learning for sparse reward reinforcement learning problem. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 637–645, 2021.

Yun Hua, Haosheng Chen, Wenhao Li, Bo Jin, Baoxiang Wang, Hongyuan Zha, and Xiangfeng Wang. Reward translation via reward machine in semi-alignable mdps. In *Forty-second International Conference on Machine Learning*, 2025.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=10uNUgI5Kl.

Pengyi Li, HAO Jianye, Hongyao Tang, Yifu Yuan, Jinbin Qiao, Zibin Dong, and Yan Zheng. R*: Efficient reward design via reward structure evolution and parameter alignment optimization with large language models. In *Forty-second International Conference on Machine Learning*, 2025a.

Pengyi Li, Hongyao Tang, Jinbin Qiao, YAN ZHENG, and Jianye HAO. Lares: Evolutionary reinforcement learning with LLM-based adaptive reward search. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=jRjvcqtdtA.

Wenhao Li, Xiangfeng Wang, Bo Jin, Dijun Luo, and Hongyuan Zha. Structured cooperative reinforcement learning with time-varying composite action space. *IEEE transactions on pattern analysis and machine intelligence*, 44(11):8618–8634, 2021a.

Yunchuan Li, Michael C Fu, and Jie Xu. An optimal computing budget allocation tree policy for monte carlo tree search. *IEEE Transactions on Automatic Control*, 67(6):2685–2699, 2021b.

Xiaotian Liu, Yijie Peng, Gongbo Zhang, and Ruihan Zhou. An efficient node selection policy for monte carlo tree search with neural networks: Informs journal on computing meritorious paper awardee. *INFORMS Journal on Computing*, 37(4):785–807, 2025.

Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=IEduRUO55F.

Yuchun Miao, Sen Zhang, Liang Ding, Rong Bao, Lefei Zhang, and Dacheng Tao. Inform: Mitigating reward hacking in rlhf via information-theoretic reward modeling. *Advances in Neural Information Processing Systems*, 37:134387–134429, 2024.

Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pp. 663–670, 2000.

S Panchapakesan. Ranking and selection procedures. In *Handbook of the Logistic Distribution*, pp. 145–167. Dekker Columbus, OH, 1992.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Yijie Peng, Chun-Hung Chen, and Michael C Fu. A review of simulation optimization with connection to artificial intelligence. *Fundamental Research*, 2025.

Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

Ashish Rao, Bidipta Sarkar, and Tejas Narayanan. Gaussian process policy optimization. *arXiv preprint arXiv:2003.01074*, 2020.

Jan Schneider, Pierre Schumacher, Simon Guist, Le Chen, Daniel Häufle, David Ha, and Dieter Büchler. Identifying policy gradient subspaces. In *International Conference on Learning Representations*, volume 2024, pp. 4446–4467, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Junjie Sheng, Shengliang Cai, Haochuan Cui, Wenhao Li, Yun Hua, Bo Jin, Wenli Zhou, Yiqiu Hu, Lei Zhu, Qian Peng, Hongyuan Zha, and Xiangfeng Wang. Vmagent: A practical virtual machine scheduling platform. In Lud De Raedt (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 5944–5947. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/860. URL https://doi.org/10.24963/ijcai.2022/860. Demo Track.

Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial intelligence review*, 55(2):945–990, 2022.

Satinder Singh, Richard L Lewis, and Andrew G Barto. Where do rewards come from. In *Proceedings of the annual conference of the cognitive science society*, pp. 2601–2606, 2009.

Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward gaming. *Advances in Neural Information Processing Systems*, 35:9460–9471, 2022.

Joar Max Viktor Skalse, Lucy Farnik, Sumeet Ramesh Motwani, Erik Jenner, Adam Gleave, and Alessandro Abate. STARC: A general framework for quantifying differences between reward functions. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wPhbtwlCDa.

Jonathan Sorg, Richard L Lewis, and Satinder Singh. Reward design via online gradient ascent. *Advances in Neural Information Processing Systems*, 23, 2010.

Jonathan Sorg, Satinder Singh, and Richard Lewis. Optimal rewards versus leaf-evaluation heuristics in planning agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pp. 465–470, 2011.

Bradly Stadie, Lunjun Zhang, and Jimmy Ba. Learning intrinsic rewards as a bi-level optimization problem. In *Conference on uncertainty in artificial intelligence*, pp. 111–120. PMLR, 2020.

Aleksandar Todorov and Matthia Sabatelli. Learning in low-dimensional subspaces: Orthogonal bottlenecks for reinforcement learning. *arXiv preprint arXiv:2605.26012*, 2026.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

Blake Wulfe, Logan Michael Ellis, Jean Mercat, Rowan Thomas McAllister, and Adrien Gaidon. Dynamics-aware comparison of learned reward functions. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=CALFyKVs87.

Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tUM39YTRxH.

Deheng Ye, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, Zhao Liu, Fuhao Qiu, Hongsheng Yu, et al. Towards playing full moba games with deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:621–632, 2020.

Rui Yu, Shenghua Wan, Yucen Wang, Chen-Xiao Gao, Le Gan, Zongzhang Zhang, and De-Chuan Zhan. Reward models in deep reinforcement learning: a survey. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pp. 10807–10816, 2025.

Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, pp. 537–546. PMLR, 2022.

Haoting Zhang, Jinghai He, Rhonda Righter, and Zeyu Zheng. Language model prompt selection via simulation optimization. *arXiv preprint arXiv:2404.08164*, 2024.

A Related Works

This work lies at the intersection of reward design in reinforcement learning and simulation optimization for efficient evaluation. The central challenge is that advances in artificial intelligence enable the generation of large sets of candidate rewards, whereas reliably evaluating these candidates remains computationally prohibitive. Prior works including optimal reward learning framework, reward evaluation through reward difference and simulation optimization in artificial intelligence is related to this paper.

A.1 Reward Design and the Optimal Reward Problem

In reinforcement learning, the reward function defines the objective that an agent optimizes and is a critical determinant of both learning efficiency and final performance. However, rewards naturally provided by the environment—such as win/loss signals in games—are often too sparse to support effective learning. In practice, one therefore resorts to hand-designed dense reward functions, which raises a long-standing question in the RL community: given a space of candidate reward functions, how can we identify the one that best serves the agent's learning objective?

Early work by Chentanez et al. (Chentanez et al., 2004) formalized the now-standard decomposition of the agent's learning signal into an internally designed reward combined with the external reward provided by the environment. Building on this foundation, Singh et al. (Singh et al., 2009), in their seminal paper *"Where Do Rewards Come From?"*, adopted an evolutionary perspective and argued that rewards should be treated as *optimizable objects* rather than fixed exogenous signals—giving rise to what is now known as the Optimal Reward Problem (ORP).

Sorg et al. (2011) provided the first systematic treatment of the ORP, consolidating its formal definition, theoretical foundations, and algorithmic principles. Among his key contributions, Sorg et al. (Sorg et al., 2010) introduced the Policy Gradient for Reward Design (PGRD) algorithm, which approximately solves the ORP online with convergence guarantees, while (Sorg et al., 2011) further demonstrated the advantages of optimal rewards over traditional heuristics in planning agents.

In recent years, the ORP has evolved into a general framework that accommodates a variety of reward-learning methods. For instance, Hu et al. (Hu et al., 2020) cast it as a bi-level optimization problem and employed meta-gradient descent to solve for parameterized optimal reward-shaping. More recently, the rise of large language models (LLMs) has opened a new paradigm: by combining the code-generation capabilities of LLMs with evolutionary search, recent works let LLMs iteratively propose and refine reward functions. Representative examples include Eureka (Ma et al., 2024), REvolve (HAZRA et al., 2025), and R* (Li et al., 2025a). However, this massive generation capacity exacerbates a severe computational bottleneck in *reward evaluation*: accurately assessing a candidate reward's true utility requires full downstream policy training, which quickly becomes prohibitive at scale. Addressing this evaluation bottleneck—how to efficiently and reliably identify high-quality rewards from a large pool of LLM-generated candidates under a limited computational budget—is precisely the problem that our work seeks to tackle.

A.2 Reward difference

A prominent line of research seeks to evaluate and compare candidate reward functions without incurring the full cost of inner-loop policy optimization. Rather than training a policy to assess each reward, these methods directly quantify differences between reward functions in a way that is invariant to transformations—such as potential-based shaping and positive affine scaling—that provably preserve the optimal policy.

Gleave et al. (Gleave et al., 2021) pioneered this direction with the *Equivalent-Policy Invariant Comparison* (EPIC) distance, a pseudometric that first canonicalizes rewards under potential shaping and then computes a Pearson-distance-based measure over a chosen coverage distribution. EPIC provides theoretical regret bounds linking reward distance to policy-performance gaps and enables evaluation that is orders of magnitude faster than training policies to completion. Building on this foundation, Wulfe et al. (Wulfe et al., 2022) proposed the *Dynamics-Aware Reward Distance* (DARD), which incorporates environment transition dynamics by evaluating reward differences only on realizable state–action–next-state tuples, rendering it more predictive of downstream policy

behavior than purely static metrics. More recently, Skalse et al. (Skalse et al., 2024) introduced *STARC* (STAndardised Reward Comparison), a general family of pseudometrics that subsumes and refines both EPIC and DARD. STARC offers tighter theoretical guarantees—inducing both upper and lower bounds on worst-case regret—and exhibits stronger empirical correlation with actual policy performance across diverse environments.

Despite their appeal, these direct reward-distance approaches share a common limitation: they rely on a sufficiently representative state–action distribution for canonicalization and comparison. In online RL—particularly under sparse extrinsic rewards—such well-covered distributions are difficult to obtain, since agents typically struggle with exploration in high-dimensional or long-horizon tasks. Consequently, reward-distance metrics generally cannot serve as a complete substitute for actual policy training when evaluating candidate rewards, motivating the need for evaluation strategies that remain reliable under limited and biased data coverage.

A.3 Simulation Optimization in Reinforcement Learning

Integrating the methodology of simulation optimization into artificial intelligence research has emerged as an important direction in the simulation optimization community in recent years. Peng et al. (Peng et al., 2025) systematically surveyed the potential connections between artificial intelligence and two major branches of simulation optimization: *gradient estimation* and *ranking and selection* (R&S). Broadly speaking, gradient estimation has already formed a natural connection with stochastic gradient methods in deep learning and reinforcement learning. In contrast, the direct application of R&S to reinforcement learning remains relatively limited and has yet to develop into a systematic research direction.

The most clearly established line of work in this area centers on node selection and simulation budget allocation in Monte Carlo Tree Search (MCTS). Li et al. (Li et al., 2021b) incorporated the idea of Optimal Computing Budget Allocation (OCBA) into MCTS, designing an R&S-based tree policy that allocates simulation resources more effectively under a limited rollout budget and thereby improves the probability of selecting the optimal action. Building on this, Liu et al. (Liu et al., 2025) extended the Asymptotically Optimal Allocation Policy (AOAP) from R&S to tree-structured settings, proposing the AOAT method and combining it with neural networks to improve node-selection efficiency in MCTS. These works demonstrate that R&S can provide effective tools for budget-constrained decision making in AI search algorithms. For more general reinforcement learning problems, however, the application of R&S methods remains in an early stage, leaving substantial room for further exploration.

In a related direction, Zhang et al. (Zhang et al., 2024) proposed a two-stage simulation optimization framework for language model prompt selection that first identifies a feasible set of prompts encoded as moderate-dimensional vectors and then sequentially evaluates them via a constructed surrogate model to maximize a pre-defined score, establishing consistency guarantees for the sequential evaluation procedure.

B Discussion of Stability Assumptions

Reinforcement learning justification of Assumption 1. In deep RL, mini-batch and environmental stochasticity prevent strict point-wise convergence; parameters instead fluctuate within a bounded neighborhood of a local optimum once the policy stabilizes after an initial warm-up phase. This assumption formalizes exactly such late-stage local stability under the shaped objective, without requiring global convergence. Consequently, the dynamics admit a valid first-order linear approximation, which forms the basis for the Gaussian learning-curve surrogate.

Reinforcement learning justification of Assumption 2. During late-stage RL training, the policy and its state-action distribution stabilize, so mini-batch and exploration noise settle into an approximately constant covariance. The bounded higher-order moment rules out extreme single-step gradients, which is the statistical prerequisite for approximating the aggregated evaluation noise as Gaussian via the martingale central limit theorem.

Algorithm 2 Reward-Adapted OCBA Rule for Fast Reward Evaluation

Require: Surrogate means $\{\hat{m}_k\}_{k=1}^K$, variances $\{\hat{s}_k^2\}_{k=1}^K$, consumed budgets $\{b_k\}_{k=1}^K$, total budget B, increment Δb

Ensure: Additional budgets $\{\Delta b_k\}_{k=1}^K$

Boundary construction.

$b \leftarrow \arg\max_k \hat{m}_k$; let (2) denote the second-best candidate

$c \leftarrow (\hat{s}_{(2)}\hat{m}_b + \hat{s}_b\hat{m}_{(2)})/(\hat{s}_b + \hat{s}_{(2)})$

$\hat{d}_b \leftarrow \hat{m}_b - c$; $\hat{d}_k \leftarrow c - \hat{m}_k$ for all $k \neq b$

OCBA weights calculation.

$w_k \leftarrow \hat{s}_k^2/\hat{d}_k^2, \quad k = 1, \ldots, K$

$B^+ \leftarrow \min\{B, \sum_{k=1}^K b_k + \Delta b\}$

Budget normalization.

$B_k^{\mathrm{tar}} \leftarrow \left(w_k/\sum_{\ell=1}^K w_\ell\right) B^+, \quad k = 1, \ldots, K$

$\widetilde{\Delta b}_k \leftarrow \max\{B_k^{\mathrm{tar}} - b_k, 0\}, \quad k = 1, \ldots, K$

$\Delta b_k \leftarrow \left(\widetilde{\Delta b}_k/\sum_{\ell=1}^K \widetilde{\Delta b}_\ell\right)\left(B^+ - \sum_{\ell=1}^K b_\ell\right), \quad k = 1, \ldots, K$

return $\{\Delta b_k\}_{k=1}^K$

C ALGORITHMS

D PROOF OF THEOREM 1

For readability, we drop subscript k in this appendix and write

$$Y_{n+1} = MY_n + W_{n+1}, \qquad \widetilde{X}_n := X_n - J^r(\theta^\star) = c^\top Y_n.$$

That is, $d := d_k$, $m := m_k$, and $\rho := \rho_k$ in the notation of Theorem 1. This is exactly the local linearized policy-dynamics model in the main text.

Step 1: State-space form and latent-state elimination target. The vector process $Y_n \in \mathbb{R}^d$ is a VAR(1)-type recursion. The scalar evaluation process $\widetilde{X}_n$ is a linear readout of the latent state. The goal is to eliminate Y_n and obtain a closed scalar recursion in $\widetilde{X}_n$.

Step 2: Cayley–Hamilton reduction to finite-order ARMA. Let the characteristic polynomial of M be

$$p_M(\lambda) = \lambda^d - a_1\lambda^{d-1} - \cdots - a_d.$$

By the Cayley–Hamilton theorem,

$$M^d = a_1M^{d-1} + \cdots + a_dI.$$

Applying this identity to Y_{n-d} and left-multiplying by $c^\top$ gives

$$c^\top M^d Y_{n-d} - a_1c^\top M^{d-1}Y_{n-d} - \cdots - a_dc^\top Y_{n-d} = 0.$$

Using $c^\top Y_n = \widetilde{X}_n$ and recursively substituting $Y_{n+1} = MY_n + W_{n+1}$ yields

$$\widetilde{X}_n = a_1\widetilde{X}_{n-1} + \cdots + a_d\widetilde{X}_{n-d} + u_n, \tag{24}$$

where u_n is a linear combination of $W_n, W_{n-1}, \ldots, W_{n-d+1}$. Therefore, the scalar evaluation process $\widetilde{X}_n$ generally admits an ARMA representation. However, under a dominant-mode approximation where the slowest eigenmode dictates the late-stage convergence bottleneck, this reduces to an effective AR(1) model.

Step 3: Projected-subspace reduction in overparameterized networks. In highly parameterized neural networks, the Hessian near $\theta^\star$ is often degenerate, with a dense spectrum near zero and no clean spectral gap. Our observable process is scalar:

$$\widetilde{X}_n = c^\top Y_n,$$

which depends only on the projection along $c = \nabla J^r(\theta^\star)$. Therefore, instead of requiring a unique dominant eigenvalue of M, we model the c-projected late-stage dynamics directly by an effective low-order contraction.

Step 4: Late-stage effective contraction in projected subspace. Define $\rho \in (0, 1)$ as the effective contraction coefficient in the projected subspace, satisfying

$$\mathbb{E}[\widetilde{X}_{n+1} \mid \widetilde{X}_n] \approx \rho\, \widetilde{X}_n$$

in the late-stage regime, and aggregate projected innovation into

$$\eta_{n+1} := c^\top W_{n+1}.$$

Collecting transverse and higher-order effects into a residual δ_{n+1}, we obtain

$$\widetilde{X}_{n+1} = \rho\, \widetilde{X}_n + \eta_{n+1} + \delta_{n+1}.$$

Neglecting δ_n in the late stage gives

$$\widetilde{X}_{n+1} \approx \rho\, \widetilde{X}_n + \eta_{n+1}.$$

Returning to $X_n = \widetilde{X}_n + J^r(\theta^\star)$ yields

$$X_{n+1} \approx (1-\rho) J^r(\theta^\star) + \rho X_n + \eta_{n+1},$$

which is exactly Eq. (13) in the main text.

E Proof of Proposition 1

In this section, we provide the moment-matching derivation that bridges the discrete linear recursion in Theorem 1 with the Gaussian trajectory model in Proposition 1.

From Theorem 1,

$$X_k^{n+1} = (1-\rho_k)\mu_k + \rho_k X_k^n + \eta_k^{n+1}, \qquad \mu_k = J_k^r(\theta_k^\star).$$

Subtracting μ_k from both sides gives

$$X_k^{n+1} - \mu_k = \rho_k (X_k^n - \mu_k) + \eta_k^{n+1}.$$

Step 1: Telescopic Unrolling and Mean Trajectory. To characterize the multi-step trajectory, define the shifted process $\widetilde{X}_k^n := X_k^n - \mu_k$. Starting from $n = 0$:

$$\widetilde{X}_k^1 = \rho_k \widetilde{X}_k^0 + \eta_k^1, \qquad \widetilde{X}_k^2 = \rho_k(\rho_k \widetilde{X}_k^0 + \eta_k^1) + \eta_k^2 = \rho_k^2 \widetilde{X}_k^0 + \rho_k \eta_k^1 + \eta_k^2.$$

By induction, unrolling over n steps gives

$$X_k^n = \mu_k + \rho_k^n (X_k^0 - \mu_k) + \sum_{j=0}^{n-1} \rho_k^j \eta_k^{n-j}. \tag{25}$$

To connect this discrete recursion to a continuous learning curve, define $\tau_k := -1/\ln(\rho_k)$, so $\rho_k^n = e^{-n/\tau_k}$. Under Assumption 2, η_k is a martingale-difference sequence and $\mathbb{E}[\eta_k^{n-j}] = 0$. Taking expectation in (25) yields

$$\mathbb{E}[X_k^n] = \mu_k + b_k e^{-n/\tau_k}, \qquad b_k := X_k^0 - \mu_k.$$

Step 2: Variance Evolution and the Noise Floor. To derive the variance of the evaluation process, apply the variance operator to Eq. (25). Since μ_k is deterministic,

$$\mathrm{Var}(X_k^n) = \mathrm{Var}\left(\rho_k^n X_k^0 + \sum_{j=0}^{n-1} \rho_k^j \eta_k^{n-j} \right).$$

First, under Assumption 2, η_k is a martingale-difference sequence, so $\mathrm{Cov}(\eta_k^i, \eta_k^j) = 0$ for $i \neq j$, and future noise is uncorrelated with the initial-state term. Hence

$$\mathrm{Var}(X_k^n) = \mathrm{Var}(\rho_k^n X_k^0) + \sum_{j=0}^{n-1} \mathrm{Var}(\rho_k^j \eta_k^{n-j}).$$

Second, pulling deterministic scales outside the variance gives (with $\sigma_{k,0}^2 := \mathrm{Var}(X_k^0)$, $\sigma_{\eta,k}^2 := \mathrm{Var}(\eta_k)$)

$$\mathrm{Var}(X_k^n) = \rho_k^{2n}\sigma_{k,0}^2 + \sum_{j=0}^{n-1} \rho_k^{2j}\sigma_{\eta,k}^2 = \rho_k^{2n}\sigma_{k,0}^2 + \sigma_{\eta,k}^2 \frac{1-\rho_k^{2n}}{1-\rho_k^2}.$$

Next, using $\sum_{j=0}^{n-1}(\rho_k^2)^j = \frac{1-\rho_k^{2n}}{1-\rho_k^2}$, we obtain the finite-time variance:

$$\mathrm{Var}(X_k^n) = \rho_k^{2n}\sigma_{k,0}^2 + \sigma_{\eta,k}^2 \left(\frac{1-\rho_k^{2n}}{1-\rho_k^2}\right). \tag{26}$$

As $n \to \infty$, $\rho_k^{2n} \to 0$, and the variance approaches the stationary noise floor:

$$\sigma_{k,\infty}^2 := \frac{\sigma_{\eta,k}^2}{1-\rho_k^2}.$$

Finally, substituting $\sigma_{k,\infty}^2$ back and then using $\rho_k^{2n} = e^{-2n/\tau_k}$ from Step 1 gives

$$\mathrm{Var}(X_k^n) = \sigma_{k,\infty}^2 + (\sigma_{k,0}^2 - \sigma_{k,\infty}^2)\rho_k^{2n} = \sigma_{k,\infty}^2 + (\sigma_{k,0}^2 - \sigma_{k,\infty}^2)e^{-2n/\tau_k}.$$

Step 3: Gaussian Approximation via Accumulated History. The final step is to establish the distributional form of X_k^n in the late stage. Isolate the zero-mean stochastic fluctuation:

$$Z_k^n := X_k^n - \mathbb{E}[X_k^n] = \sum_{j=0}^{n-1} \rho_k^j \eta_k^{n-j}.$$

First, in deep RL, single-step projected gradient noise η_k can be non-Gaussian and heavy-tailed. Under Assumption 2, however, $\{\eta_k^n\}$ is a martingale-difference array with bounded $2+\delta$ moments:

$$\mathbb{E}\big[|\eta_k^n|^{2+\delta}\big] \le C_k, \qquad \delta > 0.$$

Second, the weights $w_j = \rho_k^j$ define an exponentially weighted moving average. Its effective memory length scales as $L_{\text{eff}} \asymp 1/(1-\rho_k)$. In late-stage training, ρ_k is typically close to 1, so Z_k^n aggregates many historical perturbations rather than being dominated by a single update.

Next, verify a Lyapunov-type condition for the weighted sum:

$$\frac{\sum_{j=0}^{n-1} \mathbb{E}\Big[|\rho_k^j \eta_k^{n-j}|^{2+\delta}\Big]}{\Big(\sum_{j=0}^{n-1} \rho_k^{2j}\sigma_{\eta,k}^2\Big)^{1+\delta/2}} \le \frac{C_k \sum_{j=0}^{n-1} \rho_k^{j(2+\delta)}}{\sigma_{\eta,k}^{2+\delta}\Big(\sum_{j=0}^{n-1} \rho_k^{2j}\Big)^{1+\delta/2}}.$$

When n is large and $\rho_k \to 1$, the denominator grows faster than the numerator, so the ratio vanishes; no single weighted increment dominates total variance.

Therefore, because the Lyapunov condition holds in the late-stage regime, the martingale CLT applies. In particular, the aggregated zero-mean fluctuation converges in distribution as

$$Z_k^n \xrightarrow{d} \mathcal{N}\big(0, \mathrm{Var}(X_k^n)\big).$$

By construction, $X_k^n = \mathbb{E}[X_k^n] + Z_k^n$. Adding a deterministic term shifts only the mean, so

$$X_k^n \sim \mathcal{N}\big(\mathbb{E}[X_k^n], \mathrm{Var}(X_k^n)\big).$$

Finally, substituting the closed-form mean from Step 1 and variance from Step 2 gives

$$X_k^n \sim \mathcal{N}\Big(\mu_k + (X_k^0 - \mu_k)e^{-n/\tau_k},\ \sigma_{k,\infty}^2 + (\sigma_{k,0}^2 - \sigma_{k,\infty}^2)e^{-2n/\tau_k}\Big),$$

which yields the approximately Gaussian surrogate with matched first two moments stated in Proposition 1. This closes the bridge from non-stationary policy-gradient optimization to a tractable Gaussian surrogate and provides the statistical prerequisite for OCBA-based allocation.

F Proof of Theorem 2

Proof. Consider the continuous relaxation of maximizing (20) subject to $\sum_{k=1}^{K} N_k = T$ and $N_k > 0$. At an interior optimum, the KKT conditions imply that, for any i and j,

$$\frac{\partial \mathrm{APCS}}{\partial N_i} = \frac{\partial \mathrm{APCS}}{\partial N_j}.$$

Using

$$\frac{\partial}{\partial N_i}\Phi\left(d_i\sqrt{\frac{N_i}{v_i(n_i)}}\right) = \phi\left(d_i\sqrt{\frac{N_i}{v_i(n_i)}}\right)\frac{d_i}{2\sqrt{v_i(n_i)}\sqrt{N_i}},$$

where $\phi(\cdot)$ is the standard normal density, and setting $N_k = \alpha_k T$, the leading exponential terms in the KKT conditions yield

$$\frac{d_i^2}{v_i(n_i)}\alpha_i = \frac{d_j^2}{v_j(n_j)}\alpha_j.$$

Therefore,

$$\frac{N_i}{N_j} = \frac{\alpha_i}{\alpha_j} \sim \frac{v_i(n_i)/d_i^2}{v_j(n_j)/d_j^2}.$$

Finally, by (18),

$$m_k(n_k) \to \mu_k, \qquad v_k(n_k) \to \sigma_{k,\infty}^2 \quad \text{as } n_k \to \infty.$$

Substituting these limits into (21) gives the limiting allocation ratio. □

G Proof of Theorem 3

Proof. We first show that every candidate receives unbounded budget. Under Assumption 4, the boundary distances $\hat{d}_k$ in Algorithm 2 are bounded away from zero and infinity. Thus the OCBA weights

$$w_k = \frac{\hat{v}_k}{\hat{d}_k^2}$$

are also bounded away from zero and infinity. Hence the target proportions

$$p_k = \frac{w_k}{\sum_{\ell=1}^{K} w_\ell}$$

satisfy $p_k \geq \eta > 0$ for all k and all sufficiently large allocation epochs.

Suppose, for contradiction, that some candidate k receives only bounded budget, say $b_k(B) \leq M$. Since its target budget is at least ηB, we have

$$p_k B - b_k(B) \to \infty \qquad \text{as } B \to \infty.$$

Therefore, the OCBA update assigns candidate k positive budget increments infinitely often, contradicting the boundedness of $b_k(B)$. Hence

$$b_k(B) \to \infty, \qquad k = 1, \ldots, K.$$

By the consistency of the surrogate mean estimate,

$$\hat{m}_k(b_k(B)) \xrightarrow{p} \mu_k, \qquad k = 1, \ldots, K.$$

Let

$$\Delta_{\min} := \min_{i \neq b^\star}(\mu_{b^\star} - \mu_i) > 0,$$

where strict positivity follows from the uniqueness of the limiting best candidate. Then, with probability tending to one,

$$\max_k |\hat{m}_k(b_k(B)) - \mu_k| < \Delta_{\min}/2.$$

On this event,

$$\hat{m}_{b^\star}(b_{b^\star}(B)) > \hat{m}_i(b_i(B)), \qquad i \neq b^\star.$$

Therefore $\hat{k}_B = b^\star$ with probability tending to one, which proves

$$\mathrm{PCS}_B \to 1.$$

□

H EMPIRICAL DIAGNOSTICS OF THE GAUSSIAN SURROGATE

We examine whether the observed policy-learning dynamics are locally consistent with the Gaussian AR(1) surrogate used in our analysis. To reduce short-term fluctuations, we average the logged ground-truth reward over non-overlapping blocks of 10 PPO updates and select a local window containing 200 block-averaged observations. The surrogate is fitted using the first 70% of this window (140 observations), while the remaining 30% (60 observations) is held out for evaluation. We assess the approximation through two complementary diagnostics: its ability to predict the held-out learning trajectory and the agreement between its standardized innovations and a Gaussian distribution.

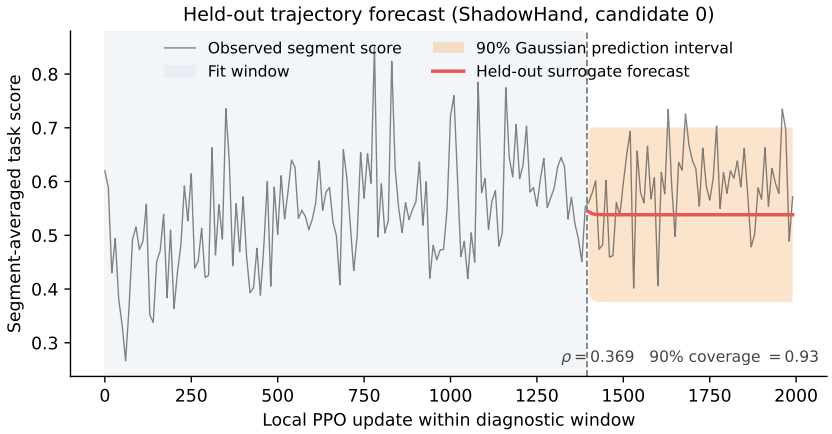


Figure 5: Held-out trajectory forecast for a representative `ShadowHand` reward candidate. The AR(1) surrogate is fitted using only the first 70% of the local window. The red line and shaded orange region show the predictive mean and the corresponding 90% Gaussian prediction interval over the held-out portion. The interval covers 93% of the held-out block-averaged observations, close to its nominal coverage.

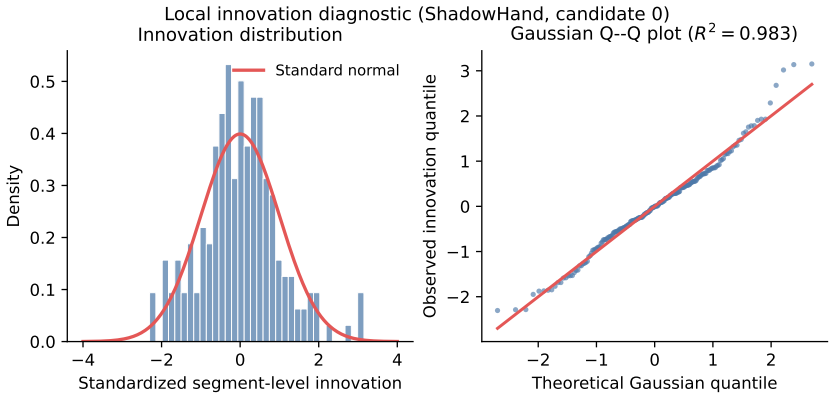
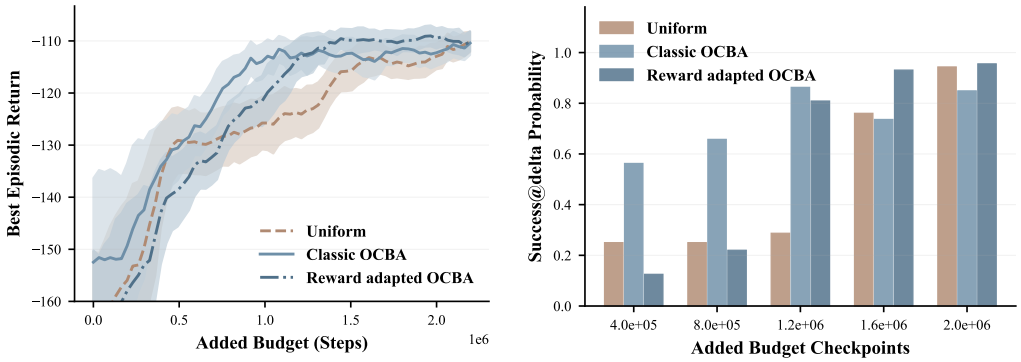


Figure 6: Innovation diagnostics for the same local surrogate fit. The histogram compares the standardized innovations with the standard-normal density, while the Q–Q plot compares their empirical quantiles with the corresponding Gaussian quantiles. The close alignment in the Q–Q plot ($R^2 = 0.983$) supports the Gaussian approximation over most of the distribution, although deviations remain in the extreme upper tail.

For the examined learning trajectory, the surrogate provides an accurate held-out description of the local training dynamics: the nominal 90% prediction interval covers 93% of the unseen observations, and the standardized innovations closely follow the Gaussian reference ($R^2 = 0.983$ in the Q–Q

plot). This evidence suggests that although the Gaussian AR(1) surrogate should not be interpreted as an exact global model of RL training, it is a reasonable approximation in the local training regime.



(a) Learning curve: best true-reward return vs. budget. (b) Success@δ probability vs. additional budget cost.

Figure 7: `MountainCar-v0`: best return and success probability vs. budget.

I Experiments Details

I.1 Sparse-reward benchmark environments

The first experiment includes the canonical sparse-reward RL benchmark shown in Figure 8. In `MountainCar-v0`, the agent receives a sparse terminal success signal only after reaching the hilltop target, while early training trajectories mostly observe weak task feedback.

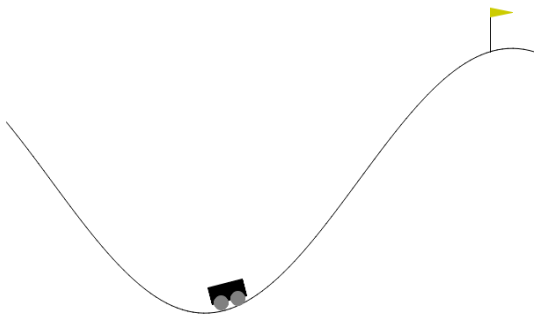


Figure 8: Sparse-reward RL benchmark used in experiment. `MountainCar-v0`: a hill-climbing control task with sparse terminal success feedback.

I.1.1 Detailed `MountainCar-v0` Reward-Shaping Settings

For `MountainCar-v0`, we use potential-based shaping

$$F_k(s_t, a_t, s_{t+1}) = \beta_k(\gamma\Phi_k(s_{t+1}) - \Phi_k(s_t)), \tag{27}$$

where $s = (\text{pos}, \text{vel})$ and Φ_k is built from normalized position/velocity features. The state features used by all candidates are

$$\text{pos}_{\text{norm}} = \frac{\text{pos} + 1.2}{1.8}, \quad |\text{vel}|_{\text{norm}} = \frac{|\text{vel}|}{0.07}, \quad \text{vel}^2_{\text{norm}} = \frac{\text{vel}^2}{0.07^2}. \tag{28}$$

The candidate set includes neutral, helpful, and deceptive shaping functions to test robustness under a fixed budget. Detailed candidate definitions are listed in Table 1.

Table 1: Detailed reward-shaping candidates for `MountainCar-v0`.

Candidate	β_k	Definition of $\Phi_k(s)$ / extra rule
baseline	0.00	$\Phi(s) = 0$
assist_pos_vel	0.14	$\Phi(s) = \text{pos}_{\text{norm}} + 0.45\lvert\text{vel}\rvert_{\text{norm}}$
assist_energy_gate	0.18	$\Phi(s) = \text{pos}_{\text{norm}} + 0.55\,\text{vel}^2_{\text{norm}} + 0.35\tanh(7.5(\text{pos} - 0.15))$
deceptive_left	0.20	$\Phi(s) = 0.85\tanh(8.0(-\text{pos} - 0.25)) - 0.20\,\text{pos}_{\text{norm}}$; if action$= 2$, add -0.03

I.1.2 Additional MountainCar-v0 figures

On `MountainCar-v0`, the learning-curve comparison in Figure 7 shows that **Adaptive OCBA** reaches strong true-reward performance faster than the two baselines under matched budget. To more directly quantify efficiency, Figure 7 also reports **Success@δ Probability** against cost, i.e., the probability of entering a target near-optimal interval versus additional budget, where Adaptive OCBA achieves better success–cost trade-offs.

Table 2: Training and allocation settings for `MountainCar-v0`.

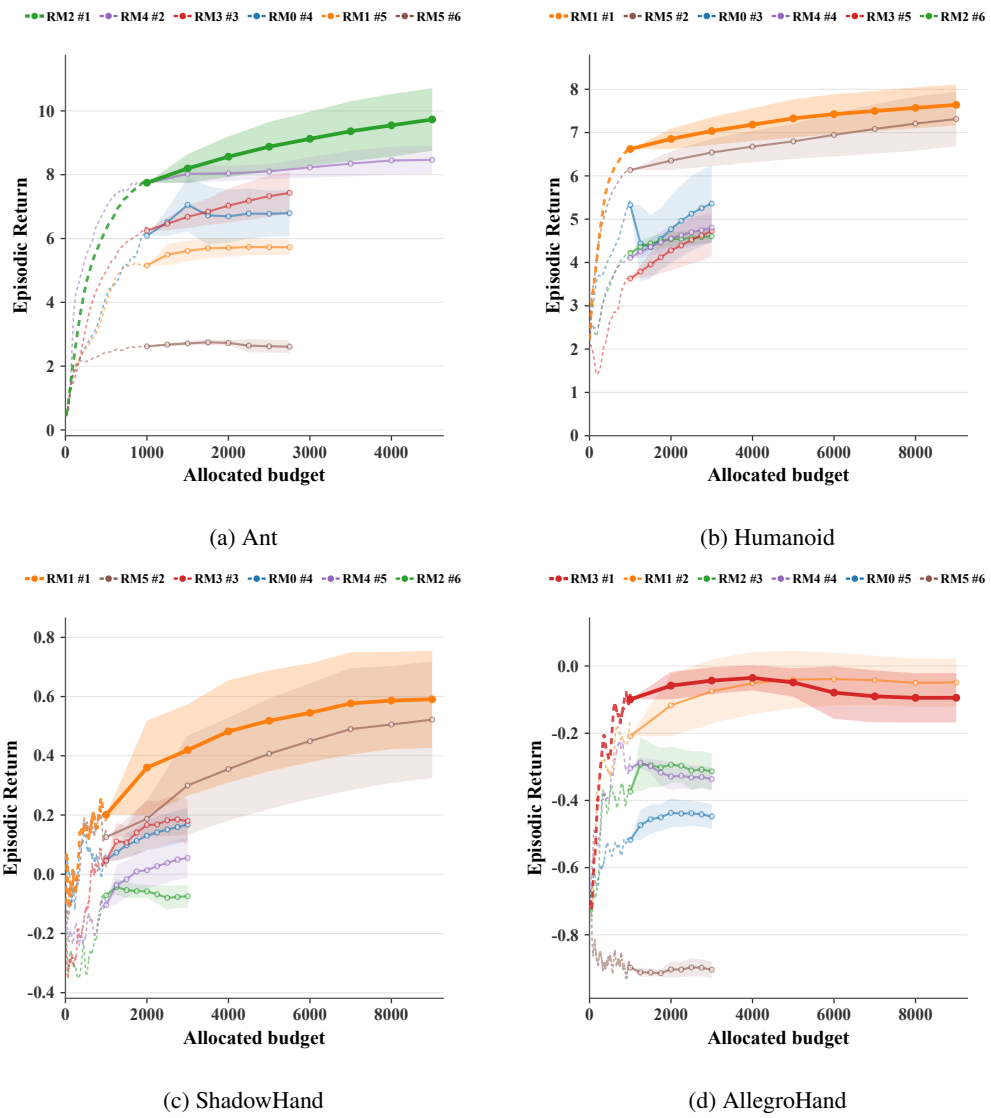
Setting	Value
Total budget B	3,600,000
Warm-up budget per candidate	200,000
Per-round incremental budget Δb	32,768
n_{envs}	16
n_{steps}	16
Update unit ($n_{\text{envs}} \times n_{\text{steps}}$)	256
γ	0.99
GAE λ	0.98
PPO epochs	4
Learning rate	3×10^{-4}
Batch size	256
Entropy coefficient	0.0
True-reward eval episodes	30
Shaped-reward eval episodes	20

I.2 Automatic Reward Generation Benchmark

Eureka is an LLM-based reward design framework that automatically generates candidate reward functions for reinforcement learning tasks and evaluates them through policy training. In our experiments, Eureka is used to generate multiple reward candidates for each task, which are subsequently trained and compared under different budget-allocation strategies. We compare *Uniform*, *OCBA*, and *Reward-adapted OCBA* under the same total budget.

I.2.1 Detailed Eureka Settings

For the Eureka experiments, we consider four continuous-control tasks in Isaac Gym: `Ant`, `Humanoid`, `ShadowHand`, and `AllegroHand`. For each task, the LLM (DeepSeek-V4-Flash, temperature $= 1.0$) generates six candidate reward functions, which are shared across all budget-allocation strategies. All candidates are trained and evaluated using three random seeds. Each candidate is first trained for a common warm-up of 1,000 PPO iterations. After warm-up, the remaining training budget is allocated over multiple rounds using Uniform allocation, OCBA, or Reward-Adapted OCBA. The three strategies therefore use the same candidate reward functions and the same total training budget, differing only in how the post warm-up budget is distributed among candidates. Candidate quality is measured using the environment ground-truth reward; after each training segment, we average the final 10 percent of the recorded metric values, with a minimum of five observations, to obtain the performance estimate used for subsequent allocation. For non-uniform allocation methods, each candidate receives a minimum allocation of 250 PPO iterations whenever selected, and all allocations are discretized in units of 250 iterations. PPO training is implemented



(a) Ant

(b) Humanoid

(c) ShadowHand

(d) AllegroHand

Figure 9: Complete learning curves of Reward-Adaptive OCBA on all four Eureka tasks.

with `rl_games` using a continuous actor–critic policy with a shared MLP. The main training and allocation parameters are summarized in Table 3. .

I.2.2 Additional Eureka figures

We additionally report the complete learning curves of Reward-Adaptive OCBA on all four Eureka tasks in Figure 9. These results provide a more detailed view of the optimization dynamics throughout training, complementing the main-text comparisons based on aggregated performance. Across `Ant`, `Humanoid`, `ShadowHand`, and `AllegroHand`, the full curves illustrate how Reward-Adaptive OCBA progressively reallocates training budget among candidate reward functions and tracks the resulting policy performance over the entire allocation process.

The environment-specific PPO settings are summarized in Table 4.

Table 3: Training and budget-allocation settings for the Eureka experiments.

Setting	Value
Environments	Ant, Humanoid, ShadowHand, AllegroHand
Number of reward candidates	6
Reward generator	DeepSeek-V4-Flash
LLM temperature	1.0
Allocation strategies	Uniform / OCBA / Reward-Adapted OCBA
Warm-up budget per candidate	1,000
Total warm-up budget	6,000
Allocation metric	`gt_reward`
Segment score	Mean of final 10% (minimum 5 points)
Allocation unit	250 iterations
Minimum allocation per RM	250
Candidate concurrency	3
Ant total budget	20,000
Ant per-round budget Δb	2,000
Ant allocation rounds	7
Others (Humanoid / ShadowHand / AllegroHand) total budget	30,000
Others per-round budget Δb	3,000
Others allocation rounds	8
PPO clip ϵ	0.2
Discount factor γ	0.99
GAE λ (τ)	0.95
Entropy coefficient	0.0
Gradient norm	1.0
Reward-shaper scale	0.01
Learning-rate schedule	Adaptive
Value/input normalization	Enabled

Table 4: Environment-specific PPO configurations used in the Eureka experiments.

Environment	LR	Horizon	Epochs	Minibatch	Critic coef.
Ant	3×10^{-4}	16	4	32,768	2
Humanoid	5×10^{-4}	32	5	32,768	4
ShadowHand	5×10^{-4}	8	5	32,768	4
AllegroHand	5×10^{-4}	8	5	32,768	4

I.3 LLM REASONING BENCHMARK

I.3.1 DETAILED GSM8K SETTINGS

We evaluate our budget-allocation strategies on the `GSM8K` mathematical reasoning benchmark. The policy model is `Qwen2.5-1.5B-Instruct`, trained with PPO using OpenRLHF. The training prompt pool contains 7,473 examples and is reused across allocation rounds, while external evaluation is performed on a fixed set consisting of 1,000 examples. Model outputs are evaluated using a rule-based `Gsm8kGrader`, which robustly extracts the final answer from common answer formats, and compares it with the reference answer using binary correctness. All candidates are trained and evaluated using three random seeds. The reported evaluation metric is the resulting accuracy on this fixed evaluation set.

We construct three candidate reward functions with different levels of alignment to the external evaluation signal. RM0 combines answer correctness with a dense partial score for reasoning quality. RM1 uses a sparse binary correctness reward derived from the same grading rule as the external evaluator. RM2 checks whether the model output follows a valid answer format and does not evaluate answer correctness. All three budget-allocation strategies use the same set of reward candidates and the same total PPO training budget, differing only in how the budget is distributed among the reward functions.

Table 5: Training and budget-allocation settings for the `GSM8K` experiments.

Setting	Value
Policy model	Qwen2.5-1.5B-Instruct
Training framework	OpenRLHF PPO
Training prompt pool	7,473
Evaluation prompts	1,000
Number of reward candidates	3
Warm-up budget per candidate	300
Total warm-up budget	900
Total budget	2,100
Per-round budget Δb	500
Allocation unit	50
Minimum allocation per RM	50
Actor learning rate	5×10^{-7}
Critic learning rate	3×10^{-6}
Rollout batch size	16
Micro train batch size	2
Initial KL coefficient	0.05
Reward normalization	Enabled
Reward clipping	$[-5, 5]$
Stop-properly penalty coef.	0.5
Max new tokens	384
Eval max new tokens	256
Overlong buffer / penalty	64 / 1.0
Save steps	50
DeepSpeed ZeRO stage	2
vLLM GPU memory utilization	0.40
LoRA rank	0

I.3.2 ADDITIONAL GSM8K RESULTS

As an additional analysis, we present the full training trajectory for Reward-Adaptive OCBA on `GSM8K` in Figure 10. This result provides a more detailed view of the allocation process and the corresponding changes in evaluation accuracy over the course of PPO training.

We define the reward model for GSM8K as follows:

```
def reward_func_0(queries, prompts, labels, **kwargs):
    rewards = []
```

```
    for query, prompt, label in zip(queries, prompts, labels):
        response = query[len(prompt):] if prompt and query.
    ↪ startswith(prompt) else query
        pred = extract_answer_from_response(response)
        if is_equiv_match(pred, label):
            rewards.append(1.0)
            continue
        reasoning = has_reasoning_steps(response)
        formatted = has_final_answer_line(response)
        if reasoning and formatted:
            rewards.append(0.6)
        elif reasoning:
            rewards.append(0.4)
        elif formatted:
            rewards.append(0.2)
        else:
            rewards.append(0.0)
    return reward_dict(rewards, {})
```

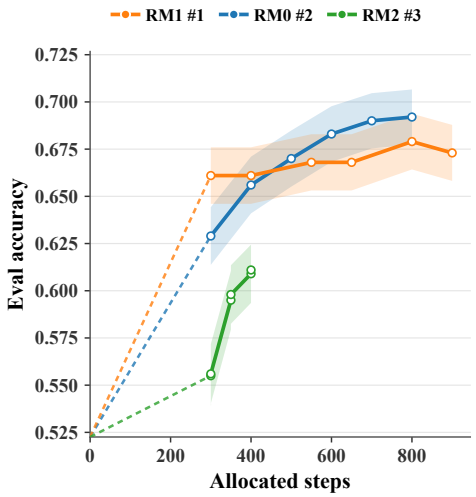
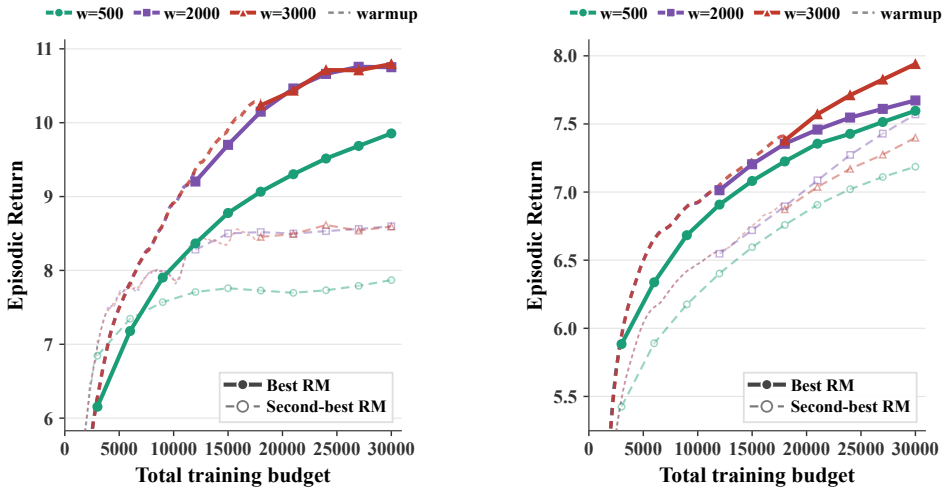


Figure 10: Complete learning curves of GSM8K.

```
def reward_func_1(queries, prompts, labels, **kwargs):
    rewards = []
    for query, prompt, label in zip(queries, prompts, labels):
        response = query[len(prompt):] if prompt and query.
    ↪ startswith(prompt) else query
        pred = extract_answer_from_response(response)
        rewards.append(1.0 if is_equiv_match(pred, label) else 0.0)
    return reward_dict(rewards, {})
```

```
def reward_func_2(queries, prompts, labels, **kwargs):
    rewards = []
    for query, prompt, _label in zip(queries, prompts, labels):
        response = query[len(prompt):] if prompt and query.
    ↪ startswith(prompt) else query
        rewards.append(1.0 if has_answer_like_format(response) else
    ↪ 0.0)
    return reward_dict(rewards, {})
```



(a) Ant (b) Humanoid

Figure 11: Ablation study on warm-up phase.

I.4 Ablation Study on Warm-up Budget

To better understand the role of the initialization phase in adaptive budget allocation, we study the effect of the warm-up budget on allocation quality and downstream performance. The purpose of this ablation is to examine whether a longer warm-up phase can provide more reliable early-stage estimates, thereby making the subsequent learning dynamics better aligned with the assumptions of our allocation method.

We conduct experiments on `Ant` and `Humanoid` in Figure 11, comparing warm-up lengths(w) of 500, 2,000, and 3,000 training steps while keeping the remaining experimental settings unchanged. We observe that a larger warm-up budget generally leads to more stable learning dynamics in the subsequent adaptive allocation stage. With more reliable initial estimates, the learning process better satisfies the assumptions underlying our allocation strategy, enabling more accurate budget allocation across candidate rewards. As a result, longer warm-up phases often lead to more stable and improved overall performance.

J Discussion and Future Directions

J.1 Revisiting the Core Idea

At its heart, this work reframes reward evaluation as a *simulation optimization* problem. The overarching observation is straightforward: modern tools especially large language models, can generate reward-function candidates at scale with relative ease, yet reliably evaluating those candidates remains computationally prohibitive because each assessment requires a full inner-loop policy-training run. We therefore ask a practical question: given a fixed training budget, how should we dynamically distribute environment interactions across candidates so as to maximize the chance of identifying the best one?

The central technical challenge is that classical simulation-optimization methods are designed for settings where each alternative can be evaluated through independent, statistically well-behaved simulation replications. Reinforcement-learning training does not fit this mood. The evaluation trajectory of any candidate reward is a complex, non-stationary, path-dependent dynamical process: as training proceeds, the policy evolves, the state–action distribution shifts, and the observed returns are serially correlated. Consequently, one cannot directly apply off-the-shelf ranking-and-selection methods without first bridging this modeling gap.

Our response is to approximate the late-stage evaluation dynamics with a lightweight Gaussian learning-curve surrogate. This approximation is motivated by the empirical observation that, after an initial transient, policy training often enters a regime of stable, approximately unimodal convergence. In this regime, the scalar evaluation process can be captured by a simple first-order autoregressive model, whose parameters encode the asymptotic performance level, the effective learning speed, and the residual noise floor. We do not claim that this surrogate is universally exact; rather, it offers a pragmatically tractable abstraction that furnishes the mean and variance statistics required by Optimal Computing Budget Allocation (OCBA). By embedding this surrogate into the OCBA framework, we obtain a *reward-adapted* allocation rule that directs additional training steps toward candidates that are either competitive or insufficiently resolved, while deprioritizing those that are clearly inferior or already converged. Theoretical analysis shows that this rule enjoys asymptotic consistency guarantees under mild regularity conditions, and the empirical results confirm that it improves evaluation efficiency across diverse tasks.

J.2 Limitations and Surprises

The most important limitation of the present work lies in the Gaussian surrogate itself. The model is ultimately an empirical construct: its functional form is suggested by qualitative training behavior rather than derived from first principles of policy optimization, and its validity depends on the late-stage training regime being sufficiently regular. In environments where learning dynamics are multimodal, highly non-stationary, or governed by sharp phase transitions, the approximation can break down.

This limitation manifests concretely in our experiments. On `Humanoid`, for instance, the reward-adapted OCBA strategy does not consistently outperform standard OCBA. We attribute this to the fact that the surrogate fails to fully capture the particular noise structure and exploration dynamics of that task, so the additional modeling effort does not translate into a reliable advantage.

Yet this observation also reveals a surprising and instructive finding: *even vanilla OCBA delivers meaningful gains over uniform allocation*. The intuition is transparent. OCBA allocates more budget to alternatives that are promising but uncertain—those with high variance or small performance gaps, and less to alternatives that are either already converged or performing poorly. During the exploratory phase of RL, the sampling variances across candidates are often comparable; consequently, standard OCBA naturally concentrates resources on candidates that exhibit the greatest room for improvement. This behavior aligns closely with the heuristic rationale behind early stopping and progressive allocation, suggesting that the basic principle of "allocate where information is most valuable" is robust even when the dynamical surrogate is imperfect.

Another practical issue deserving attention is the choice of warm-up budget (the so-called first-stage sampling in simulation-optimization parlance). In the present experiments this budget was chosen in a relatively ad-hoc manner, yet prior work in simulation optimization has long recognized that first-stage sample sizes can substantially influence final selection performance. A systematic analysis of how warm-up budget should be set in the RL context, balancing the need for reliable surrogate initialization against the cost of prematurely committing resources, remains an open question that we intend to pursue.

J.3 Future Directions

The intersection of simulation optimization and reinforcement learning is, in our view, a fertile and underexplored territory. Classical RL can itself be seen as a process of repeated simulation and learning; the two fields are therefore natural allies. This paper offers only a first step, and several promising directions lie ahead.

First, if one is willing to trade some theoretical tidiness for broader applicability, the simulation-optimization perspective can be extended well beyond reward evaluation. Hyperparameter tuning, algorithm selection, and even architecture search in RL all involve repeatedly evaluating competing alternatives under a fixed computational budget. The online budget-allocation philosophy developed here could, in principle, be adapted to these settings, potentially improving efficiency wherever candidate options must be compared through partial training runs.

Second, and more fundamentally, there is ample room to develop richer surrogate models of the RL training process. The Gaussian learning-curve approximation used here is intentionally minimal; more expressive models whether drawn from Bayesian optimization, neural temporal processes, or mechanistic theories of deep-learning dynamics, could yield tighter uncertainty estimates and sharper allocation rules. Extending such surrogates to the large-scale, non-stationary training dynamics characteristic of modern LLM alignment is a particularly exciting challenge. We hope that the conceptual framing introduced in this work encourages further cross-pollination between the simulation-optimization and reinforcement-learning communities.