LEARNING TO ALLOCATE: CRITIC TRAINING FOR TEST-TIME AGENT SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time search for LLM agents requires deciding which partial trajectories deserve further computation. Standard critic training supervises each prefix independently using continuation outcomes, although the resulting scores jointly determine how a shared search budget is allocated. We introduce RATE (Relative Allocation Training), a framework that combines pointwise continuation-value learning with relative supervision across competing prefixes. Both forms of supervision use the same sampled continuations. The joint objective preserves the ideal continuation-potential target while emphasizing relative score errors that affect allocation. At test time, the trained critic guides the allocation of continuation slots while the actor remains fixed. Across four agent environments, RATE improves mean candidate reward over Best-of-N (BoN) by 3.95–8.90 percentage points under a fixed continuation budget. The gains also extend from fixed-frontier evaluation to repeated budget allocation during online search.

1 INTRODUCTION

Test-time search gives an LLM agent several attempts at the same task (Snell et al., 2025; Wu et al., 2025). As these attempts unfold, some find promising directions while others make little progress. Continuing one leaves less computation for the rest. With a limited budget, the search procedure must decide which attempts to continue, replicate, or stop before their final outcomes are available (Figure 1). We study how to learn this allocation while keeping the underlying agent fixed.

Particle search maintains a frontier of partial trajectories and periodically resamples it to assign the next continuation slots (Zhao et al., 2024; Hwang et al., 2026). A critic scores the prefixes, and the scores jointly determine how these slots are distributed. Agentic Monte Carlo uses learned values to guide a fixed agent through this process (Hwang et al., 2026). Our focus is how to train the critic for the allocation decisions its scores induce.

A common approach fits each prefix to outcomes from sampled continuations (Wang et al., 2024; Hwang et al., 2026). Appropriately chosen pointwise losses can recover the ideal continuation potential (Zhao et al., 2024). For a critic that cannot fit every prefix exactly, the way errors affect allocation also matters. Allocation depends on both ordering and score gaps: prediction errors can

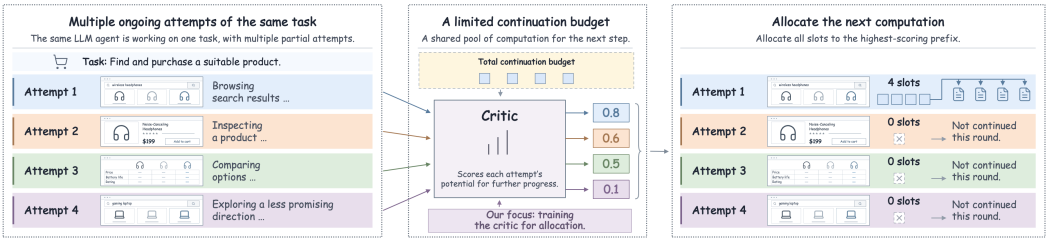

Figure 1: **Allocating a shared continuation budget.** Multiple partial attempts at the same task compete for further computation. A critic scores the current prefixes to guide which attempts are continued, replicated, or stopped.

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.

change which prefix receives the budget and whether to concentrate it at all. This motivates supervising relative scores across competing prefixes alongside their individual continuation potentials.

We introduce RATE (Relative Allocation Training), a framework for training search critics around the allocation decisions they guide. RATE combines pointwise potential learning with relative supervision through a soft distribution on each frontier. The critic still scores each prefix individually; the training objective couples prefixes that compete for the same continuation budget. Both terms use the same sampled continuations, and the trained critic guides search without updating the actor.

Continuations from the same prefix can yield different rewards, so allocation supervision needs to account for this variability. In reward-tilted search, the reference parent distribution is obtained by normalizing expected continuation weights. Directly normalizing sampled weights into training labels can instead lead to a different target. RATE therefore uses the sampled weights as coefficients in the frontier loss. For a fixed frontier, the reference parent distribution minimizes the expected loss. The joint objective retains the same continuation-potential target while placing greater emphasis on relative score errors that affect allocation. Our default allocation rule distributes slots uniformly when the predicted values are close and otherwise selects the top-scoring prefix. We connect its candidate-quality loss to the relative errors emphasized by frontier supervision.

We evaluate RATE on TextCraft, ALFWorld, WebShop, and BabyAI-Text, sharing the actor, critic backbone, training frontiers, and continuation outcomes across learned critics. RATE improves mean candidate reward over Best-of-N (BoN) by 3.95–8.90 percentage points across the four environments. Complete online rollouts in ALFWorld further evaluate repeated allocation at different search widths.

Our contributions are as follows.

1. We formulate critic learning for allocating a shared continuation budget among competing prefixes, with mean candidate reward as the direct measure of allocation quality.
2. We introduce RATE, a joint pointwise-and-frontier training framework that learns continuation values and relative scores from sampled outcomes while preserving the continuation-potential target.
3. We characterize the allocation-relevant score errors emphasized by RATE, connect them to the candidate-quality loss of budget allocation, and evaluate the framework across four agent environments and repeated online search.

2 Particle Search and Resampling Targets

2.1 Reward-tilted particle inference

Let $\pi(\tau)$ be the trajectory distribution of a fixed sequence model or policy, and let $R(\tau)$ be its terminal reward. Reward-tilted inference targets $q^\star_\lambda(\tau) = Z_\lambda^{-1}\pi(\tau)e^{\lambda R(\tau)}$ for $\lambda > 0$. Particle methods approximate this distribution with N trajectories that are extended and periodically resampled. The resampling weights combine information from each observed prefix with an estimate of its future reward (Heng et al., 2020; Zhao et al., 2024).

At a resampling step, the frontier $\mathcal{F} = \{(s_i, b_i)\}_{i=1}^N$ records the information available before future continuations are sampled. Here s_i is a trajectory prefix, and b_i collects the log-weight terms fixed by that prefix. Each continuation follows π from its own prefix, so conditioning on the frontier leaves its marginal continuation law unchanged. For the resulting future reward G_i, the ideal log continuation potential is $V_i^\star = \log \mathbb{E}_\pi[e^{\lambda G_i} \mid s_i]$. We assume $0 < \mathbb{E}_\pi[e^{\lambda G_i} \mid s_i] < \infty$ and finite prefix log weights b_i. The ideal resampling distribution is

$$p_i^\star = \frac{e^{b_i}\mathbb{E}_\pi[e^{\lambda G_i} \mid s_i]}{\sum_j e^{b_j}\mathbb{E}_\pi[e^{\lambda G_j} \mid s_j]} = \mathrm{softmax}_i(b_i + V_i^\star). \tag{1}$$

This soft reference supplies RATE's training target. The learned values also guide the discrete budget decisions in Section 3.4, where the objective is the mean reward of generated candidates.

2.2 POINTWISE POPULATION TARGETS

To identify pointwise supervision targets, write a sampled continuation weight as $Y = e^{\lambda G} > 0$. When the corresponding conditional second moments are finite, raw-return regression with loss $(r - G)^2$ targets $r^\star(s) = \mathbb{E}_\pi[G \mid s]$, while regression on the transformed reward targets $h^\star(s) = \mathbb{E}_\pi[Y \mid s]$. We call the latter loss, $(h - Y)^2$, Potential-MSE.

RATE parameterizes this positive potential through its logarithm v and uses the pointwise loss

$$\ell_{\text{potential}}(v; Y) = e^v - Yv. \tag{2}$$

For a fixed prefix, its conditional risk is $\mathcal{R}_{\text{potential}}(v \mid s) = e^v - \mathbb{E}_\pi[Y \mid s]v$, with first-order condition $\partial_v \mathcal{R}_{\text{potential}} = e^v - \mathbb{E}_\pi[Y \mid s] = 0$. Since $\partial_v^2 \mathcal{R}_{\text{potential}} = e^v > 0$, the unique optimum is $v^\star(s) = \log \mathbb{E}_\pi[Y \mid s] = V^\star(s, \lambda)$. This standard Bregman mean loss under a log link (Banerjee et al., 2005) lets the critic learn the log potential directly. Potential-MSE and Equation (2) therefore recover the same potential in their respective parameterizations.

A known upper bound $G \le c(s)$ permits the shifted parameterization $\widetilde{Y} = e^{\lambda(G - c(s))}$ and $\widetilde{v} = v - \lambda c(s)$. This bounds the sampled target by one and preserves the conditional optimum after restoring the known offset.

2.3 EXPECTATION AND NORMALIZATION

Equation (1) averages each exponential continuation weight before normalizing across the frontier. The order of these operations matters.

Raw-return regression instead averages the reward before exponentiation, inducing $p_i^{\text{Return}} = \text{softmax}_i\big(b_i + \lambda \mathbb{E}[G_i \mid s_i]\big)$, whereas Jensen's inequality gives $e^{\lambda \mathbb{E}[G_i \mid s_i]} \le \mathbb{E}[e^{\lambda G_i} \mid s_i]$. For $\lambda > 0$, equality requires G_i to be conditionally constant almost surely. Thus, stochastic future rewards separate the mean-return target from the continuation potential.

Normalizing sampled weights into soft labels before taking their expectation need not recover $p^\star$. RATE retains the sampled weights in its frontier loss so that its optimum is $p^\star$. Section 3 presents the joint training objective and derives the targets learned with and without this weighting.

3 RATE

RATE uses each frontier as a unit of supervision for the critic that allocates its continuation budget. Figure 2 shows how sampled outcomes supply both pointwise and frontier supervision, and how the trained critic guides allocation during search.

3.1 ALLOCATION AND TRAINING OBJECTIVE

The critic evaluates each prefix through $V_\theta(s_i, \lambda)$. On a frontier, its scores induce the soft training distribution $p_{\theta,i} = \text{softmax}_i\big(b_i + V_\theta(s_i, \lambda)\big)$. Training couples these predictions through this distribution. For sampled continuation weights $Y_i = e^{\lambda G_i}$, RATE assigns the coefficient $e^{b_i} Y_i$ to each negative log probability. With a finite positive scale $a(\mathcal{F})$ fixed before continuation sampling, the frontier loss is

$$\mathcal{L}_{\text{frontier}}(\theta) = -\mathbb{E}\left[\frac{1}{a(\mathcal{F})} \sum_{i=1}^{N} e^{b_i} Y_i \log p_{\theta,i}\right]. \tag{3}$$

We combine this loss with the pointwise potential loss in Equation (2):

$$\mathcal{L}_{\text{RATE}}(\theta) = \mathcal{L}_{\text{frontier}}(\theta) + \gamma\, \mathbb{E}\left[\ell_{\text{potential}}\big(V_\theta(s, \lambda); Y\big)\right], \tag{4}$$

with $\gamma \ge 0$. Both terms use the same continuation samples. The frontier term supervises how prefixes share the budget. For $\gamma > 0$, the pointwise term anchors the absolute log potential, resolving the common-shift ambiguity of the frontier softmax. In the fixed-frontier evaluation, the values determine candidate-slot allocations through Section 3.4.

For bounded rewards $G_i \le c_i$, choosing $a(\mathcal{F}) = \sum_j e^{b_j + \lambda c_j}$ makes the sampled coefficients sum to at most one.

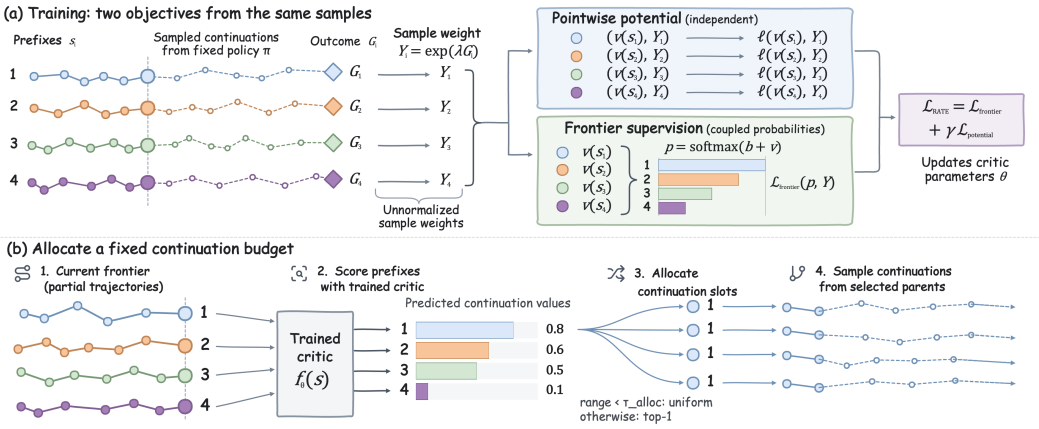

Figure 2: **Training a critic for allocation with RATE.** (a) Continuations sampled from a fixed policy provide weights $Y_i = e^{\lambda G_i}$ for both pointwise potential learning and frontier supervision. The joint objective updates the critic. (b) The trained critic scores competing prefixes to guide the redistribution of a fixed continuation budget.

3.2 Allocation supervision from sampled continuations

The frontier loss is linear in each sampled weight Y_i. On a fixed frontier, $\mathbb{E}[Y_i \mid \mathcal{F}] = \mathbb{E}_\pi[Y_i \mid s_i]$, so its conditional risk is

$$\begin{aligned}\mathcal{R}_{\mathrm{frontier}}(p \mid \mathcal{F}) &= -\frac{1}{a(\mathcal{F})} \sum_i e^{b_i} \mathbb{E}_\pi[Y_i \mid s_i] \log p_i \\ &= -\frac{\sum_j e^{b_j} \mathbb{E}_\pi[Y_j \mid s_j]}{a(\mathcal{F})} \sum_i p_i^\star \log p_i. \end{aligned} \tag{5}$$

The result is cross-entropy with $p^\star$, scaled by a positive factor independent of p. Subtracting its value at $p^\star$ gives a direct characterization of the excess risk.

Proposition 1 (Population target of the frontier loss)**.** *Fix a frontier with $0 < \mathbb{E}_\pi[Y_i \mid s_i] < \infty$ and finite b_i. Let $0 < a(\mathcal{F}) < \infty$ be determined before continuation sampling. For every distribution p with positive entries,*

$$\begin{aligned}&\mathcal{R}_{\mathrm{frontier}}(p \mid \mathcal{F}) - \mathcal{R}_{\mathrm{frontier}}(p^\star \mid \mathcal{F}) \\ &\qquad = \frac{\sum_j e^{b_j} \mathbb{E}_\pi[Y_j \mid s_j]}{a(\mathcal{F})} D_{\mathrm{KL}}(p^\star \,\|\, p). \end{aligned} \tag{6}$$

Consequently, $p^\star$ is the unique minimizer over the probability simplex.

The identity follows directly from Equation (5). Since $p^\star$ has full support, boundary distributions have infinite risk. A single continuation per prefix supplies an unbiased estimate of the conditional risk.

To compare with normalized-label training, rewrite the sample loss in Equation (3) as

$$-\frac{\sum_j e^{b_j} Y_j}{a(\mathcal{F})} \sum_i \frac{e^{b_i} Y_i}{\sum_j e^{b_j} Y_j} \log p_{\theta,i}. \tag{7}$$

Thus, normalized labels recover the same loss when weighted by their total sampled mass.

If the total weight is discarded, cross-entropy with the normalized sampled weights gives

$$\mathcal{R}_{\mathrm{Norm}}(p \mid \mathcal{F}) = -\sum_i \mathbb{E}\left[\frac{e^{b_i} Y_i}{\sum_j e^{b_j} Y_j} \middle| \mathcal{F}\right] \log p_i, \tag{8}$$

$$p_i^{\mathrm{Norm}} = \mathbb{E}\left[\frac{e^{b_i} Y_i}{\sum_j e^{b_j} Y_j} \middle| \mathcal{F}\right]. \tag{9}$$

Table 1: Population targets of the training losses.

Method	Sample loss	Population optimum
Return-MSE	$(r-G)^2$	$r^\star = \mathbb{E}_\pi[G \mid s]$
Potential-MSE	$(h-Y)^2$	$h^\star = \mathbb{E}_\pi[Y \mid s]$
Pointwise potential	$e^v - Yv$	$v^\star = \log \mathbb{E}_\pi[Y \mid s]$
Norm-CE	$-\sum_i \frac{e^{b_i} Y_i}{\sum_j e^{b_j} Y_j} \log p_{\theta,i}$	$p_i^{\mathrm{Norm}} = \mathbb{E}\left[\frac{e^{b_i} Y_i}{\sum_j e^{b_j} Y_j} \middle\vert \mathcal{F}\right]$
Resampling-aligned	$-\sum_i e^{b_i} Y_i \log p_{\theta,i}/a(\mathcal{F})$	$p^\star$

The conditional risk is cross-entropy with p^{Norm}, so this distribution is its unique minimizer. The random denominator means that averaging the normalized weights need not recover $p_i^\star$, a standard ratio bias (Agapiou et al., 2017). Its second-order expansion involves the variance of the total sampled weight and its covariance with each particle's weight (Appendix A.1). The targets coincide when the weights are deterministic or share a common positive random factor that cancels on normalization. Retaining the total sampled weight in RATE avoids this change in the supervision target.

3.3 Population target and allocation errors

Both terms admit the ideal potential when it is representable. To see how they weight approximation errors, write $\delta_i = V_\theta(s_i, \lambda) - V_i^\star$. Substituting $p_{\theta,i} = p_i^\star e^{\delta_i} / \sum_j p_j^\star e^{\delta_j}$ into the KL divergence gives

$$D_{\mathrm{KL}}(p^\star \,\|\, p_\theta) = \log \mathbb{E}_{i\sim p^\star}[e^{\delta_i}] - \mathbb{E}_{i\sim p^\star}[\delta_i]. \tag{10}$$

A common shift in all δ_i cancels exactly. When the errors are close to their $p^\star$-weighted mean, a second-order expansion gives

$$D_{\mathrm{KL}}(p^\star \,\|\, p_\theta) \approx \frac{1}{2} \operatorname{Var}_{i\sim p^\star}(\delta_i). \tag{11}$$

Locally, the frontier term penalizes the $p^\star$-weighted variance of score errors across competing prefixes. By comparison, the pointwise excess risk at a prefix with error $\delta = v - V^\star(s, \lambda)$ is

$$\mathcal{R}_{\mathrm{potential}}(v \mid s) - \mathcal{R}_{\mathrm{potential}}(V^\star \mid s) = \mathbb{E}_\pi[Y \mid s]\big(e^\delta - 1 - \delta\big), \tag{12}$$

which is locally $\mathbb{E}_\pi[Y \mid s]\delta^2/2$ and penalizes absolute score error. Appendix A.3 gives the expansions. Thus RATE preserves the potential target while adding a penalty on the relative errors that redistribute computation. For a restricted critic, the joint objective changes which approximation errors receive more weight. Table 1 summarizes the population targets.

3.4 From relative supervision to budget allocation

The soft distribution p_θ provides relative supervision during training. The learned scores guide candidate allocation, and the bound below relates its mean reward to relative score errors on a fixed frontier.

Consider a fixed frontier with equal prefix weights ($b_i = 0$). Let A contain n eligible parents, with mean continuation returns $\mu_i = \mathbb{E}_\pi[G_i \mid s_i]$ and predicted values h_i in return units. An allocation q assigns a fraction q_i of the candidate budget to parent i, giving expected mean reward

$$J_{\mathcal{F}}(q) = \sum_{i\in A} q_i \mu_i. \tag{13}$$

We use a simple thresholded greedy rule. Let $d_{\mathcal{F}}$ denote the range of predicted values across the frontier and q^{top1} the allocation to its highest-scoring eligible parents. Then

$$q_{\tau,i} = \begin{cases} 1/n, & d_{\mathcal{F}} < \tau_{\mathrm{alloc}}, \\ q_i^{\mathrm{top1}}, & d_{\mathcal{F}} \geq \tau_{\mathrm{alloc}}, \end{cases}. \tag{14}$$

When predicted values are close, uniform allocation forgoes little predicted gain while retaining competing branches. Otherwise, the budget concentrates on the most promising parents. The threshold is in return units; its value and the full protocol are given in Appendix B.5.

Proposition 2 (Local allocation regret). *Fix a frontier with $n \geq 1$, rewards $G_i \in [0,1]$, and $\lambda > 0$. For $f_i^\star = V_i^\star/\lambda$, $e_i = h_i - f_i^\star$, and $\operatorname{span}_A(e) = \max_{i\in A} e_i - \min_{i\in A} e_i$,*

$$\max_{i\in A} \mu_i - J_{\mathcal{F}}(q_\tau) \leq (1 - 1/n)\tau_{\text{alloc}}\mathbf{1}\{d_{\mathcal{F}} < \tau_{\text{alloc}}\} + \operatorname{span}_A(e) + \lambda/8. \tag{15}$$

The terms separate the predicted gain forgone by uniform allocation, relative prediction error, and the difference between the potential target and mean return. Relative errors influence both whether to concentrate the budget and which parent receives it. RATE's frontier term emphasizes these differences, while pointwise learning anchors the values. Appendices A.4 and A.5 prove the bound and connect it to the training objective's excess risk.

4 Experiments

We first compare critic objectives on shared frontiers and continuation outcomes, so each method allocates the same continuation opportunities. We then evaluate repeated allocation in complete ALFWorld searches, varying search width and testing final selection under a common compute cap. Finally, we fine-tune the actor on TextCraft trajectories collected by RATE and evaluate it without search.

Environments and models. We evaluate critic training on TextCraft (Prasad et al., 2024), ALFWorld (Shridhar et al., 2021), WebShop (Yao et al., 2022), and BabyAI-Text, a text interface to BabyAI (Chevalier-Boisvert et al., 2019). Qwen3-8B (Qwen Team, 2025) serves as the fixed actor. All trained critics use Qwen3-4B-Instruct-2507 with a scalar prediction head and LoRA adapters (Hu et al., 2022).

Baselines and ablations. Best-of-N (BoN) generates one continuation per original prefix; we evaluate the resulting candidate pool. The trained baselines are Return-MSE and Return-MSE + RankNet. The latter adds RankNet's pairwise logistic loss (Burges et al., 2005) to Return-MSE, ranking nonterminal prefixes within each frontier by their average sampled returns. We ablate RATE in two ways. Pointwise CGF removes the frontier term, using only the potential loss in Equation (2). Norm-CE retains the pointwise loss but replaces the sample-dependent weight on normalized-label cross-entropy with a fixed coefficient. Within each environment, learned critics share training frontiers, continuation labels, and the number of training updates. We also evaluate Qwen 3.7 as a critic, scoring the same prefixes without task-specific training. Appendix B specifies the objectives, hyperparameters, and selection protocols.

Allocation and evaluation. All critics use Equation (14) with a shared threshold, and checkpoints are selected by validation Reward. Mean candidate Reward averages the resulting allocation over the shared continuation outcomes. Appendix B specifies the evaluation protocol. Soft-distribution KL diagnostics appear in Appendix B.10.

Main results. RATE achieves the highest mean candidate Reward in all four environments, improving on BoN by 3.95–8.90 percentage points (Table 2, Figure 3). It also outperforms Return-MSE and Return-MSE + RankNet with the same actor, critic backbone, training frontiers, and continuation labels. The comparison with Return-MSE + RankNet shows the value of supervising allocation beyond adding pairwise ranking to return regression. The 4B RATE critic exceeds Qwen 3.7 in every environment.

Ablations. Frontier supervision improves mean Reward over Pointwise CGF in all four environments, with the largest gains on TextCraft and ALFWorld (+2.01 and +2.99 percentage points; Figure 3). This comparison supports joint relative supervision of competing prefixes. RATE also improves on the fixed-weight Norm-CE variant in every environment, connecting the target-preserving weighting in Section 3.2 to candidate quality.

Scaling online search. To test repeated allocation during online search, we run complete ALFWorld rollouts on all 134 tasks at widths $N \in \{1, 2, 4, 8\}$ with a fixed actor. The learned critics share a resampling procedure that periodically reallocates unfinished trajectories, balancing uniform

Table 2: **Main results and RATE ablations.** Mean candidate Reward (%, ↑). Critic scores use the shared thresholded greedy rule. Parentheses give gains over Best-of-N (BoN), in percentage points. Bold/underlined values mark the best/second-best results among trained critics.

Method	TextCraft	ALFWorld	WebShop	BabyAI-Text
Baselines and reference				
Best-of-N	66.98	71.11	46.56	21.60
Qwen 3.7	75.53	74.01	51.31	25.42
Return-MSE	73.30 (+6.33)	73.74 (+2.64)	51.39 (+4.83)	24.44 (+2.84)
Return-MSE + RankNet	<u>75.06</u> (+8.09)	75.12 (+4.01)	51.45 (+4.90)	24.63 (+3.03)
RATE and ablations				
Pointwise CGF (no frontier term)	73.86 (+6.89)	73.65 (+2.54)	<u>51.90</u> (+5.34)	24.98 (+3.37)
Norm-CE (fixed frontier weight)	75.00 (+8.03)	<u>76.10</u> (+4.99)	51.74 (+5.18)	<u>25.32</u> (+3.72)
RATE	**75.88** (+8.90)	**76.63** (+5.53)	**52.44** (+5.89)	**25.56** (+3.95)

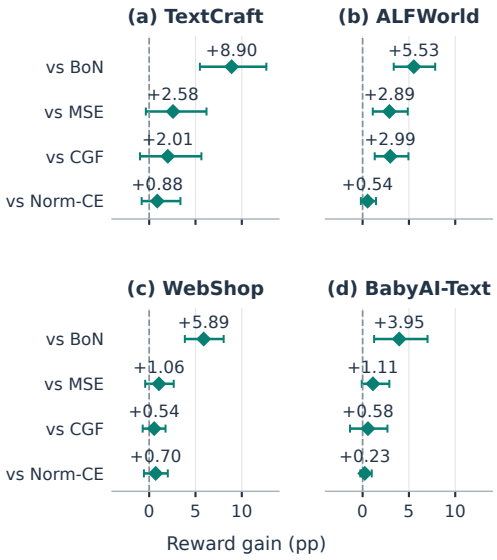

Figure 3: **Reward gains and ablations.** RATE minus BoN, MSE (Return-MSE), and two ablations, CGF (Pointwise CGF) and Norm-CE (percentage points), with 95% paired task-bootstrap intervals.

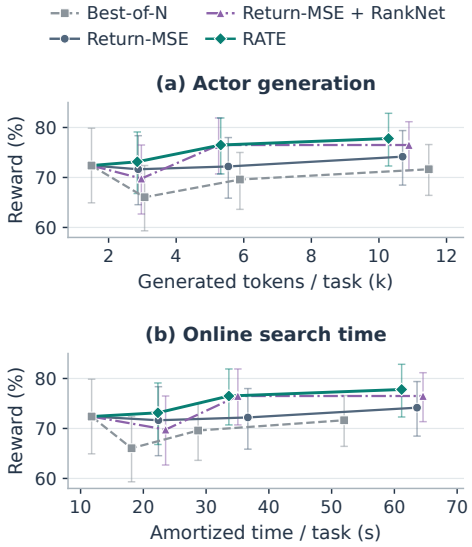

Figure 4: **Online search on ALFWorld.** Mean candidate Reward versus generation and time, including critic and environment overhead. Points follow $N = 1, 2, 4, 8$; bars show 95% task-bootstrap intervals.

continuation with greedy selection (Appendix E.1). Table 3 reports mean candidate Reward; Figure 4 reports actor generation and elapsed search time.

RATE's Reward rises from 72.39% at $N = 1$ to 77.80% at $N = 8$ and exceeds BoN for every $N > 1$. RATE leads at $N = 2, 8$ and ties with Return-MSE + RankNet at $N = 4$; at $N = 8$, its gain over BoN is 6.16 percentage points. The improvement thus extends to successive allocation decisions along complete agent trajectories. At matched widths $N > 1$, RATE uses fewer actor tokens than BoN, with longer elapsed search times once critic and resampling costs are included (Figure 4).

Closed-loop search under a compute budget. We next evaluate final trajectory selection on all 134 ALFWorld tasks with eight particles, horizon $H = 100$, and a per-task cap of 15 PFLOPs covering actor generation, critic evaluation, and final selection. Each method selects one candidate

Table 3: **ALFWorld search at different widths.** Mean candidate Reward (%). $N = 1$ is a shared actor-only reference.

Method	$N = 1$	$N = 2$	$N = 4$	$N = 8$
BoN	72.39	66.04	69.59	71.64
Return-MSE	72.39	<u>71.64</u>	<u>72.20</u>	74.16
Return-MSE + RankNet	72.39	69.78	**76.49**	<u>76.49</u>
RATE	72.39	**73.13**	**76.49**	**77.80**

Table 4: **TextCraft SFT.** Actor success (%). Each sampler collects 10^6 generated tokens.

Training data	Success
Initial actor	64.2
Uniform (all)	66.8
RATE (all)	**68.4**
Uniform (matched)	66.8
RATE (matched)	**71.2**

Table 5: **Budgeted closed-loop search on ALFWorld.** All 134 tasks, $N = 8$, and $H = 100$, with a per-task cap of 15 PFLOPs. Success is measured after critic selection; BoN uses RATE's selector. Computation is analytically modeled and averaged over tasks.

Method	Reward (%)	Selected success (%)	PFLOPs/task
Best-of-N	71.64	83.58	9.309
Return-MSE	71.55	81.34	8.864
Pointwise CGF	74.63	84.33	9.022
RATE	**76.12**	**85.07**	8.178

using its critic and the visible interaction history; the independent BoN pool uses the same final selector as RATE. RATE raises candidate Reward from 71.64% to 76.12% and selected success from 83.58% to 85.07% relative to BoN (Table 5). Average modeled computation falls from 9.309 to 8.178 PFLOPs per task, a 12.1% reduction. The better candidate pool thus translates into higher final success under the shared compute cap. Appendix B.9 gives the budget and selection protocol.

Collecting data for SFT. Trajectories collected by RATE improve TextCraft actor performance after supervised fine-tuning (SFT). RATE and uniform sampling each receive one million generated tokens across 200 training tasks. After SFT on the usable successful trajectories, the resulting actors reach 68.4% and 66.8% success, respectively (Table 4). Matching the training-set sizes gives 71.2% versus 66.8%, a 4.4-point gain with a paired 95% interval of $[1.4, 7.4]$ points. The critic is used only during collection; these gains are measured in ordinary actor rollouts without search. Appendix B.8 gives the experimental details.

Training dynamics. RATE improves validation Reward early in training across all four environments (Figure 5). Appendix B.10 gives the corresponding KL curves.

5 Related Work

Test-time search and process supervision. Tree of Thoughts and Language Agent Tree Search explore partial reasoning and action trajectories through search (Yao et al., 2023; Zhou et al., 2024). REBASE makes compute allocation explicit by distributing node expansions according to normalized process-reward scores (Wu et al., 2025). Process supervision can come from human step annotations (Lightman et al., 2024) or sampled completions (Wang et al., 2024). Rewarding Progress uses process advantages to measure progress along a trajectory and guide search (Setlur et al., 2025). RATE uses sampled outcomes to supervise relative scores among prefixes sharing a continuation budget.

Learned particle guidance. Controlled SMC learns control functions for particle proposals (Heng et al., 2020). Twisted SMC learns intermediate potentials through contrastive distribution matching or regression on sampled terminal potentials (Zhao et al., 2024), and has been applied to mathematical reasoning (Feng et al., 2025). Agentic Monte Carlo guides a fixed agent through sequential Monte Carlo (Hwang et al., 2026). It identifies the log expected exponential reward as the ideal value and uses return regression in its practical single-continuation approximation. RATE retains the continuation-potential target and adds joint supervision of competing prefixes.

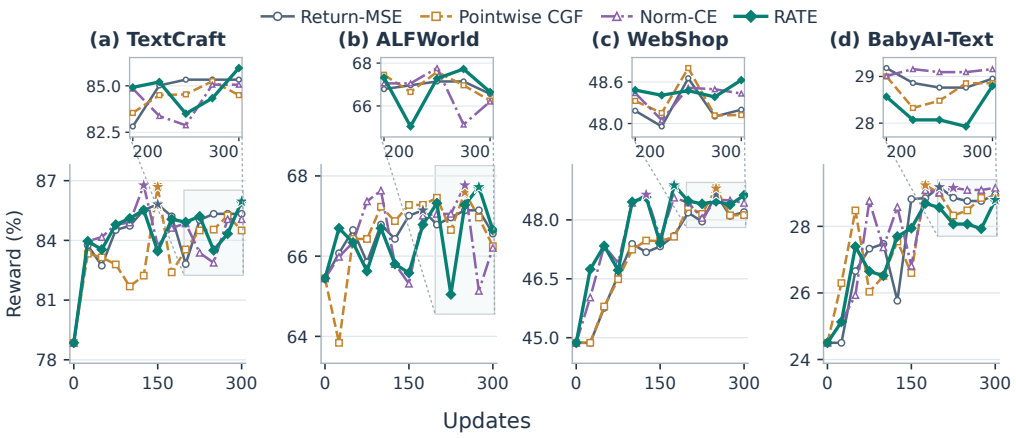

Figure 5: **Reward during training.** Curves use the validation splits. Stars mark the checkpoints selected by validation Reward and used in Table 2; insets enlarge updates 200–300.

Listwise and decision-focused learning. ListNet learns scores jointly over competing candidates (Cao et al., 2007), while decision-focused learning trains predictors for their downstream optimization use (Wilder et al., 2019). This need not change the population target: under its assumptions, SPO+ shares the conditional-mean optimum of least squares while allowing different approximation trade-offs in restricted models (Elmachtoub & Grigas, 2022). RATE applies this perspective to allocation: its joint loss preserves the continuation-potential target while emphasizing relative errors. Retaining the total sampled weight avoids the target change induced by unweighted normalized-label training.

Normalization and finite-particle effects. Bias from random normalization is well established in self-normalized importance sampling (Agapiou et al., 2017), as is the loss of distinct particles under repeated resampling (Del Moral et al., 2012). Entropic particle filtering mitigates premature exploitation through adaptive temperature and look-ahead (Giannone et al., 2026). Recent theory connects reward-model error to guarantees for particle sampling, while experiments show that sampling fidelity need not predict task accuracy (Golowich et al., 2026). RATE connects relative score errors to candidate-quality loss; the appendix also analyzes coupling and retention under resampling.

6 Conclusion

We introduced RATE, a framework for training critics around the allocation decisions in test-time agent search. RATE combines pointwise potential learning with relative supervision over competing trajectory prefixes. Our analysis shows that the joint objective preserves the ideal continuation-potential target while placing greater weight on relative errors that redistribute computation. Across four agent environments, RATE improves mean candidate reward over Best-of-N, with gains extending to repeated online search on ALFWorld. RATE-guided data collection also improves actor performance after fine-tuning on TextCraft. These results demonstrate the value of training critics to account for how their predictions jointly determine the allocation of computation.

Future work can extend RATE to frontiers generated by the search it guides, allowing critic supervision to adapt to the state distribution induced by allocation. A complementary direction is to learn the marginal value of additional computation, accounting for uncertainty and dependence among branches to connect allocation more directly to final task success. RATE also opens a path toward iterative self-improvement, where critic-guided search supplies actor training data and fresh rollouts update the critic as the actor evolves. This raises the broader question of how to allocate computation between solving current tasks and collecting experience that improves future performance.

AI USE STATEMENT

Generative AI tools assisted with manuscript writing, translation, and language editing, including rephrasing selected passages, organizing the structure, and improving clarity. They also supported literature search, discovery, and preliminary organization to help identify relevant work that might otherwise have been overlooked. For some mathematical derivations presented in the appendix, AI tools assisted with formulating mathematical statements, writing formulas, expanding algebraic steps, and developing, writing, and checking parts of the derivations and proofs. The objectives of these derivations were developed collaboratively by the authors with AI assistance. AI tools also supported consistency checks between experimental results, implementation details, and their descriptions.

All AI-assisted text, references, and other information have been reviewed and verified by the authors. The authors are responsible for the research questions, core method design, experimental setup, and final analyses and conclusions, and take full responsibility for the content of this paper.

ETHICS STATEMENT

Our experiments use established agent benchmarks and do not recruit human participants. More effective search may also amplify biased or harmful behaviors of the underlying language models. Deployment beyond these benchmark environments should include safeguards appropriate to the application and human oversight.

REPRODUCIBILITY STATEMENT

Section 3 specifies RATE, and Appendix A provides supporting derivations. Appendix B reports data splits, continuation budgets, optimization settings, and evaluation metrics. Appendices D and E provide prompts, pseudocode, critic input formats, and frontier record schemas. The main critic comparisons reuse the same frontiers and continuation labels across training objectives.

REFERENCES

S. Agapiou, O. Papaspiliopoulos, D. Sanz-Alonso, and A. M. Stuart. Importance sampling: Intrinsic dimension and computational cost. *Statistical Science*, 32(3):405–431, 2017. doi: 10.1214/17-STS611. URL https://arxiv.org/abs/1511.06196.

Arindam Banerjee, Xin Guo, and Hui Wang. On the optimality of conditional expectation as a Bregman predictor. *IEEE Transactions on Information Theory*, 51(7):2664–2669, 2005. doi: 10.1109/TIT.2005.850145. URL https://doi.org/10.1109/TIT.2005.850145.

Chris J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, 2005. URL https://www.microsoft.com/en-us/research/wp-content/uploads/2005/08/icml_ranking.pdf.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 129–136, 2007. doi: 10.1145/1273496.1273513. URL https://www.microsoft.com/en-us/research/publication/learning-to-rank-from-pairwise-approach-to-listwise-approach/.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A platform to study the sample efficiency of grounded language learning. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJeXCo0cYX.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. On adaptive resampling strategies for sequential Monte Carlo methods. *Bernoulli*, 18(1):252–278, 2012. doi: 10.3150/10-BEJ335. URL https://arxiv.org/abs/1203.0464.

Randal Douc, Olivier Cappé, and Eric Moulines. Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pp. 64–69, 2005. doi: 10.1109/ISPA.2005.195385. URL https://arxiv.org/abs/cs/0507025.

Adam N. Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management Science*, 68(1):9–26, 2022. doi: 10.1287/mnsc.2020.3922. URL https://pubsonline.informs.org/doi/10.1287/mnsc.2020.3922.

Shengyu Feng, Xiang Kong, Shuang Ma, Aonan Zhang, Dong Yin, Chong Wang, Ruoming Pang, and Yiming Yang. Step-by-step reasoning for math problems via twisted sequential Monte Carlo. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Ze4aPP0tIn.

Giorgio Giannone, Guangxuan Xu, Nikhil Shivakumar Nayak, Rohan Mahesh Awhad, Shivchander Sudalairaj, Kai Xu, and Akash Srivastava. Mitigating premature exploitation in particle-based Monte Carlo for inference-time scaling. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/64092.

Noah Golowich, Fan Chen, Dhruv Rohatgi, Raghav Singhal, Carles Domingo-Enrich, Dylan J. Foster, and Akshay Krishnamurthy. Reject, resample, repeat: Understanding parallel reasoning in language model inference. *arXiv preprint arXiv:2603.07887*, 2026. URL https://arxiv.org/abs/2603.07887.

Jeremy Heng, Adrian N. Bishop, George Deligiannidis, and Arnaud Doucet. Controlled sequential Monte Carlo. *The Annals of Statistics*, 48(5):2904–2929, 2020. doi: 10.1214/19-AOS1914. URL https://doi.org/10.1214/19-AOS1914.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Dae Yon Hwang, Raunaq Suri, Valentin Villecroze, Anthony L. Caterini, Jesse C. Cresswell, Noël Vouitsis, and Brendan Leigh Ross. Agentic Monte Carlo: Simulating reinforcement learning for black-box agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/62740.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/aca97732e30bcf1303bc22ac3924fd16-Abstract-Conference.html.

Archiki Prasad, Alexander Koller, Mareike Hartmann, Peter Clark, Ashish Sabharwal, Mohit Bansal, and Tushar Khot. ADaPT: As-needed decomposition and planning with language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 4226–4252. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-naacl.264. URL https://aclanthology.org/2024.findings-naacl.264/.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *International Conference on Learning Representations*, pp. 60808–60838, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/98711dea460bdefe0e651ca23ec98ba2-Abstract-Conference.html.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0IOX0YcCdTn.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *International Conference on Learning Representations*, pp. 10131–10165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1b623663fd9b874366f3ce019fdfdd44-Abstract-Conference.html.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 1658–1665, 2019. doi: 10.1609/aaai.v33i01.33011658. URL https://ojs.aaai.org/index.php/AAAI/article/view/3982.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/8c3caae2f725c8e2a55ecd600563d172-Abstract-Conference.html.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/82ad13ec01f9fe44c01cb91814fd7b8c-Abstract-Conference.html.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/271db9922b8d1f4dd7aaef84ed5ac703-Abstract-Conference.html.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Baker Grosse. Probabilistic inference in language models via twisted Sequential Monte Carlo. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 60704–60748. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhao24c.html.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

APPENDIX

CONTENTS

A	**Additional Derivations**	**14**
A.1	Normalization bias and its local expansion	14
A.2	Bounded loss coefficients	15
A.3	Relative and absolute score errors	15
A.4	Local allocation regret	16
A.5	Frontier risk, clipping, and allocation regret	17
A.6	Coupled parent selection over multiple resampling steps	18
A.7	Special Cases and the Small-λ Limit	18
A.8	Resampling Schemes	19
B	**Experimental Details**	**20**
B.1	Data and continuation budgets	20
B.2	Optimization	20
B.3	Return-MSE + RankNet	21
B.4	Qwen 3.7 as a critic	21
B.5	Fixed-frontier allocation and checkpoint selection	21
B.6	Evaluation metrics	22
B.7	Figure evaluation	22
B.8	RATE-guided SFT on TextCraft	22
B.9	Budgeted closed-loop search	23
B.10	KL diagnostics	23
B.11	Allocation-threshold sensitivity	24
C	**Results by Task Type**	**25**
D	**Qwen API Prompts**	**26**
D.1	TextCraft, ALFWorld, and BabyAI-Text	26
D.2	WebShop	27
E	**Algorithms, Prompts, and Data Formats**	**28**
E.1	Pseudocode	28
E.2	Actor prompts and action formats	30
E.3	Critic input templates and a complete example	33
E.4	Frontier and continuation-label records	36

A ADDITIONAL DERIVATIONS

A.1 NORMALIZATION BIAS AND ITS LOCAL EXPANSION

For the ratio expansion, write $X_i = e^{b_i} Y_i$, $m_i = \mathbb{E}[X_i \mid \mathcal{F}] = e^{b_i}\mathbb{E}_\pi[Y_i \mid s_i]$, $S = \sum_j X_j$, and $\mu_S = \mathbb{E}[S \mid \mathcal{F}] = \sum_j m_j$. The normalized sampled weights are $\widehat{p}_i = X_i/S$, and the conditional cross-entropy in Equation (8) has the unique minimizer $p^{\mathrm{Norm}} = \mathbb{E}[\widehat{p} \mid \mathcal{F}]$. All entries are positive because $X_i > 0$.

Since $p_i^\star = m_i/\mu_S$,

$$\mathbb{E}[\widehat{p}_i \mid \mathcal{F}] - p_i^\star = \mathbb{E}\left[\frac{X_i - p_i^\star S}{S}\middle|\mathcal{F}\right]. \tag{A.1}$$

The numerator has conditional mean zero, but division by the random total S can change the expectation. No independence across particles is needed for this identity. It vanishes for deterministic weights and for weights of the form $X_i = Cm_i$ with a shared positive random factor C.

To obtain the local approximation, write $\Delta X_i = X_i - m_i$ and $\Delta S = S - \mu_S$. Expanding the ratio around (m_i, μ_S) gives

$$\frac{X_i}{S} = \frac{m_i}{\mu_S} + \frac{\Delta X_i}{\mu_S} - \frac{m_i \Delta S}{\mu_S^2} - \frac{\Delta X_i \Delta S}{\mu_S^2} + \frac{m_i(\Delta S)^2}{\mu_S^3} + r_i, \tag{A.2}$$

where the exact remainder is

$$r_i = \frac{(\mu_S \Delta X_i - m_i \Delta S)(\Delta S)^2}{\mu_S^3 S}. \tag{A.3}$$

The first-order terms have conditional mean zero. Taking expectations yields

$$\mathbb{E}[\widehat{p}_i \mid \mathcal{F}] - p_i^\star = \frac{m_i \operatorname{Var}(S \mid \mathcal{F})}{\mu_S^3} - \frac{\operatorname{Cov}(X_i, S \mid \mathcal{F})}{\mu_S^2} + \mathbb{E}[r_i \mid \mathcal{F}], \tag{A.4}$$

whenever these terms are integrable. When the joint fluctuations are small enough for the remainder to be negligible, this gives

$$\mathbb{E}[\widehat{p}_i \mid \mathcal{F}] - p_i^\star \approx \frac{m_i \operatorname{Var}(S \mid \mathcal{F})}{\mu_S^3} - \frac{\operatorname{Cov}(X_i, S \mid \mathcal{F})}{\mu_S^2}. \tag{A.5}$$

The two terms describe the effect of fluctuations in the total weight and its covariance with X_i. For example, if $|\Delta X_i| \le \eta m_i$ and $|\Delta S| \le \eta \mu_S$ almost surely for $0 < \eta < 1$, then $|r_i| \le 2p_i^\star \eta^3/(1-\eta)$. Without control of the remainder, Equation (A.1) gives the exact bias.

A.2 Bounded loss coefficients

Known reward bounds $G_i \le c_i$ provide a bounded scaling with $a(\mathcal{F}) = \sum_j e^{b_j + \lambda c_j}$. Set $M_{\mathcal{F}} = \max_j (b_j + \lambda c_j)$ to evaluate the coefficients stably:

$$\mathcal{L}_{\text{frontier}} = -\mathbb{E}\left[\sum_i \frac{e^{b_i + \lambda G_i - M_{\mathcal{F}}}}{\sum_j e^{b_j + \lambda c_j - M_{\mathcal{F}}}} \log p_{\theta,i}\right]. \tag{A.6}$$

The sampled coefficients sum to at most one. Since $a(\mathcal{F})$ is fixed before continuation sampling, this scaling preserves the conditional optimum. Different choices of $a(\mathcal{F})$ can still change the relative weight of frontiers when fitting a restricted critic.

A.3 Relative and absolute score errors

For a fixed frontier, write $\delta_i = V_\theta(s_i, \lambda) - V_i^\star$. The softmax parameterization gives

$$\frac{p_{\theta,i}}{p_i^\star} = \frac{e^{\delta_i}}{\sum_j p_j^\star e^{\delta_j}}. \tag{A.7}$$

Taking the negative logarithm and averaging over $i \sim p^\star$ gives Equation (10). Combined with Proposition 1, it characterizes the conditional excess risk relative to the ideal distribution.

Center the errors as $\epsilon_i = \delta_i - \sum_j p_j^\star \delta_j$ and define $f(t) = \log \sum_i p_i^\star e^{t\epsilon_i}$. Then $f(0) = f'(0) = 0$ and $f''(0) = \operatorname{Var}_{i \sim p^\star}(\delta_i)$. Taylor's theorem yields

$$D_{\mathrm{KL}}(p^\star \,\|\, p_\theta) = f(1) = \frac{1}{2} \operatorname{Var}_{i \sim p^\star}(\delta_i) + \frac{f^{(3)}(\xi)}{6}, \qquad \xi \in (0,1). \tag{A.8}$$

The third derivative is the third centered moment of ϵ_i under probabilities proportional to $p_i^\star e^{t\epsilon_i}$. Hence $|f^{(3)}(t)| \le 8\|\epsilon\|_\infty^3$, giving an $O(\|\epsilon\|_\infty^3)$ remainder for Equation (11). Equivalently, the quadratic term is

$$\frac{1}{2} \delta^\top \left[\operatorname{diag}(p^\star) - p^\star (p^\star)^\top\right] \delta. \tag{A.9}$$

The matrix annihilates the all-ones vector, matching exact invariance to common score shifts.

For the pointwise term, substitute $v = V^\star(s, \lambda) + \delta$ into the conditional risk $\mathcal{R}_{\text{potential}}(v \mid s)$ in Section 2.2 and subtract the risk at $V^\star(s, \lambda)$. This gives Equation (12), with expansion

$$\begin{aligned} \mathbb{E}_\pi[Y \mid s](e^\delta - 1 - \delta) &= \frac{\mathbb{E}_\pi[Y \mid s]}{2} \delta^2 + \rho, \\ |\rho| &\le \frac{\mathbb{E}_\pi[Y \mid s]}{6} e^{|\delta|} |\delta|^3. \end{aligned} \tag{A.10}$$

Thus the pointwise term penalizes absolute error near its optimum, while the frontier term penalizes relative error weighted by the ideal parent probabilities.

A.4 LOCAL ALLOCATION REGRET

Fix a frontier $\mathcal{F}$ and its eligible set A, with $n = |A| \geq 1$. All expectations below condition on this frontier. Write $\mu_i = \mathbb{E}_\pi[G_i \mid s_i]$, $f_i^\star = \lambda^{-1} \log \mathbb{E}_\pi[e^{\lambda G_i} \mid s_i]$, and $e_i = h_i - f_i^\star$ for the clipped effective scores $h_i = \mathrm{clip}(z_i, 0, 1)$. An allocation made before the evaluation continuations assigns counts K_i with $\sum_i K_i = M$ and $\mathbb{E}[K_i \mid \mathcal{F}] = M q_i$. If each offspring of parent i has conditional mean μ_i given the allocation, linearity of expectation gives

$$\mathbb{E}\left[\frac{1}{M}\sum_{i\in A}\sum_{k=1}^{K_i} G_{i,k}\,\middle|\,\mathcal{F}\right] = \sum_{i\in A} q_i \mu_i = J_{\mathcal{F}}(q). \tag{A.11}$$

This identity allows dependent sibling returns and terminal parents, whose continuation returns equal their known rewards. The shared-bank evaluation estimates $J_{\mathcal{F}}(q)$ using the allocation-weighted empirical mean of the stored continuation returns.

Potential and mean return. For $G \in [0,1]$ and $\lambda > 0$,

$$0 \leq \lambda^{-1} \log \mathbb{E}[e^{\lambda G}] - \mathbb{E}[G] \leq \lambda/8. \tag{A.12}$$

To prove this, let $h(t) = \log \mathbb{E}[e^{tG}]$. Then $h(0) = 0$, $h'(0) = \mathbb{E}[G]$, and $h''(t)$ is the variance of G under exponential tilting. Its support remains $[0,1]$, so $0 \leq h''(t) \leq 1/4$. The integral remainder gives $h(\lambda) - \lambda h'(0) = \int_0^\lambda (\lambda - t) h''(t)\, dt \in [0, \lambda^2/8]$. At $\lambda = 0.05$, the return-unit discrepancy is at most 0.00625.

Proof of Proposition 2. The known-success branch has zero regret. For the remaining frontiers, define the predicted gain forgone by any allocation q relative to the highest score as

$$\Delta_{\mathrm{pred}}(q; \mathcal{F}) = \max_{i\in A} h_i - \sum_{i\in A} q_i h_i. \tag{A.13}$$

By Equation (A.12) and $f_i^\star = h_i - e_i$,

$$\begin{aligned}
\max_{i\in A} \mu_i - J_{\mathcal{F}}(q) &\leq \max_{i\in A} f_i^\star - \sum_{i\in A} q_i f_i^\star + \lambda/8 \\
&\leq \Delta_{\mathrm{pred}}(q; \mathcal{F}) + \sum_{i\in A} q_i e_i - \min_{i\in A} e_i + \lambda/8 \\
&\leq \Delta_{\mathrm{pred}}(q; \mathcal{F}) + \mathrm{span}_A(e) + \lambda/8.
\end{aligned} \tag{A.14}$$

For q_τ, the predicted gap is zero when $d_{\mathcal{F}} \geq \tau_{\mathrm{alloc}}$: clipping is monotone, so every eligible maximizer of z_i also maximizes h_i. This remains true when the allocation is split across tied raw maxima. When $d_{\mathcal{F}} < \tau_{\mathrm{alloc}}$, the allocation is uniform on A. The eligible clipped-score span is at most the complete-frontier span $d_{\mathcal{F}}$, giving

$$\Delta_{\mathrm{pred}}(q_\tau; \mathcal{F}) = \frac{1}{n}\sum_{i\in A}\left(\max_{j\in A} h_j - h_i\right) \leq \left(1 - \frac{1}{n}\right) d_{\mathcal{F}} \leq \left(1 - \frac{1}{n}\right) \tau_{\mathrm{alloc}}. \tag{A.15}$$

Substitution into Equation (A.14) proves Equation (15). □

Terminal handling and pure top-1. The eligible set contains nonterminal prefixes and terminal entries with positive known return. A terminal entry with return one triggers allocation to successful terminal entries, attaining the largest possible mean return. If A is empty, every entry is terminal with return zero, and every allocation has zero regret. Otherwise, excluding zero-return terminal entries leaves the mean-return optimum unchanged. Pure top-1 is the case $\tau_{\mathrm{alloc}} = 0$, for which the predicted-gap term vanishes.

Binary rewards. Equation (A.32) implies that $f_i^\star$ is strictly increasing in μ_i. Its inverse $\psi(x) = (e^{\lambda x} - 1)/(e^\lambda - 1)$ on $[0,1]$ has $0 < \psi'(x) \leq L_\lambda := \lambda/(1 - e^{-\lambda})$. Applying this bound between $\max_i f_i^\star$ and each $f_i^\star$ and averaging under q strengthens Equation (A.14) to

$$\max_{i\in A} \mu_i - J_{\mathcal{F}}(q) \leq L_\lambda \left[\Delta_{\mathrm{pred}}(q; \mathcal{F}) + \mathrm{span}_A(e)\right]. \tag{A.16}$$

Thus exact values followed by greedy allocation attain the local mean-return optimum for binary rewards.

A.5 FRONTIER RISK, CLIPPING, AND ALLOCATION REGRET

Consider the equal prefix weights $b_i = 0$ used in fixed-frontier training. Let the complete training frontier contain N entries. Write $\hat{f}_i = f_\theta(s_i)$ for the critic's raw return-unit prediction. The effective return scores in the frontier term of Equation (E.1) are $z_i = \hat{f}_i$ for nonterminal entries and $z_i = G_i$ for terminal entries. Write $r_i = z_i - f_i^\star$ for their raw effective errors; terminal entries have $f_i^\star = G_i$ and hence $r_i = e_i = 0$. Define

$$\begin{aligned} p_i^\star &= \mathrm{softmax}_i(\lambda f^\star), \qquad p_{\theta,i} = \mathrm{softmax}_i(\lambda z), \\ \mathcal{E}_\mathcal{F} &= \frac{2}{\lambda^2} D_{\mathrm{KL}}(p^\star \,\|\, p_\theta), \quad R_\mathcal{F} = \max\{1, \mathrm{span}_{1:N}(z)\}. \end{aligned} \tag{A.17}$$

The common $-\lambda$ shift in the implemented logits leaves these soft distributions unchanged. Proposition 1, applied to the unscaled frontier term of Equation (E.1), gives the exact excess-risk identity

$$\begin{aligned} \mathcal{R}_{\mathrm{frontier}}(p_\theta \mid \mathcal{F}) - \mathcal{R}_{\mathrm{frontier}}(p^\star \mid \mathcal{F}) &= c_\mathcal{F} \frac{\lambda^2}{2} \mathcal{E}_\mathcal{F}, \\ c_\mathcal{F} := \frac{1}{N} \sum_i \mathbb{E}_\pi[e^{\lambda(G_i - 1)} \mid s_i] &\in [e^{-\lambda}, 1]. \end{aligned} \tag{A.18}$$

Proposition A.1 (Non-asymptotic relative-error control)**.** *For $G_i \in [0,1]$, $\lambda > 0$, and finite effective scores,*

$$e^{-\lambda R_\mathcal{F}} \mathrm{Var}_{i\sim\mathrm{Unif}(N)}(r_i) \le \mathcal{E}_\mathcal{F} \le e^{\lambda R_\mathcal{F}} \mathrm{Var}_{i\sim\mathrm{Unif}(N)}(r_i). \tag{A.19}$$

In particular,

$$\mathrm{span}_A(r) \le \sqrt{2N e^{\lambda R_\mathcal{F}} \mathcal{E}_\mathcal{F}}. \tag{A.20}$$

If $z \in [0,1]^N$, then $e = r$, so this also bounds the clipped-error term in Proposition 2.

Proof. Along the segment from ideal to effective scores, let $\Phi(t) = \log \sum_i \exp(\lambda[f_i^\star + t r_i])$ and $p_t = \mathrm{softmax}(\lambda[f^\star + tr])$. Then $\Phi'(t) = \lambda \mathbb{E}_{p_t}[r]$ and $\Phi''(t) = \lambda^2 \mathrm{Var}_{p_t}(r)$, so

$$D_{\mathrm{KL}}(p^\star \,\|\, p_\theta) = \Phi(1) - \Phi(0) - \Phi'(0) = \lambda^2 \int_0^1 (1-t)\, \mathrm{Var}_{p_t}(r)\, dt. \tag{A.21}$$

Since $f_i^\star \in [0,1]$, the span of $(1-t)f^\star + tz$ is at most $R_\mathcal{F}$. Hence $e^{-\lambda R_\mathcal{F}}/N \le p_{t,i} \le e^{\lambda R_\mathcal{F}}/N$. Using $\mathrm{Var}_{p_t}(r) = \min_c \sum_i p_{t,i}(r_i - c)^2$, the lower probability bound gives a lower variance bound; evaluating at $c = N^{-1}\sum_i r_i$ gives the corresponding upper bound. Thus

$$e^{-\lambda R_\mathcal{F}} \mathrm{Var}_{\mathrm{Unif}(N)}(r) \le \mathrm{Var}_{p_t}(r) \le e^{\lambda R_\mathcal{F}} \mathrm{Var}_{\mathrm{Unif}(N)}(r). \tag{A.22}$$

Integrating and using $\int_0^1 (1-t)\, dt = 1/2$ proves Equation (A.19). For the span, the maximum and minimum deviations from the mean $\bar{r} = N^{-1}\sum_i r_i$ satisfy

$$N \mathrm{Var}_{\mathrm{Unif}(N)}(r) \ge (r_{\max} - \bar{r})^2 + (r_{\min} - \bar{r})^2 \ge \tfrac{1}{2} \mathrm{span}_{1:N}(r)^2, \tag{A.23}$$

with the zero-span case immediate. Combining these bounds proves Equation (A.20). □

The frontier term therefore controls relative errors in the effective raw scores. To cover score clipping in deployment, the pointwise term also supplies the required absolute anchor.

Proposition A.2 (Joint-risk control with clipping)**.** *Let $\gamma > 0$, and let $\mathcal{R}_{\mathrm{RATE}}$ denote the conditional expectation of Equation (E.1). Relative to the unrestricted ideal prediction, define*

$$\mathcal{X}_\mathcal{F} = \frac{2}{\gamma\lambda^2} \left[\mathcal{R}_{\mathrm{RATE}}(\hat{f} \mid \mathcal{F}) - \mathcal{R}_{\mathrm{RATE}}(f^\star \mid \mathcal{F}) \right]. \tag{A.24}$$

For arbitrary finite raw predictions and $G_i \in [0,1]$,

$$\begin{aligned} \max_{i\in A} \mu_i - J_\mathcal{F}(q_\tau) \le{}& \left(1 - \frac{1}{n}\right) \tau_{\mathrm{alloc}} \mathbf{1}\{d_\mathcal{F} < \tau_{\mathrm{alloc}}\} \\ &+ \sqrt{2N e^\lambda \mathcal{X}_\mathcal{F}} + \lambda/8. \end{aligned} \tag{A.25}$$

Proof. For one prefix, set $m_i = \mathbb{E}_\pi[e^{\lambda(G_i-1)} \mid s_i]$. Its pointwise conditional risk in return coordinates is $\rho_i(x) = e^{\lambda(x-1)} - m_i\lambda(x-1)$. This convex function is minimized at $f_i^\star \in [0,1]$; clipping a prediction to $[0,1]$ therefore cannot increase its risk. Moreover, $\rho_i''(x) = \lambda^2 e^{\lambda(x-1)} \geq \lambda^2 e^{-\lambda}$ on $[0,1]$. For each nonterminal entry,

$$\rho_i(\hat{f}_i) - \rho_i(f_i^\star) \geq \rho_i(h_i) - \rho_i(f_i^\star) \geq \frac{\lambda^2 e^{-\lambda}}{2} e_i^2. \tag{A.26}$$

Terminal entries have $e_i = 0$ and nonnegative pointwise excess. The frontier excess is also nonnegative by Equation (A.18). Summing yields $\mathcal{X}_\mathcal{F} \geq e^{-\lambda} N^{-1} \sum_i e_i^2$. Finally, $\mathrm{span}_A(e)^2 \leq 2\sum_i e_i^2 \leq 2Ne^\lambda \mathcal{X}_\mathcal{F}$; Proposition 2 gives the result. □

These bounds use n for eligible parents and N for the full training frontier. The joint-risk bound accommodates clipping without restricting the raw prediction range. For binary rewards, substituting the same clipped-error bound into Equation (A.16) replaces the additive $\lambda/8$ term by the multiplicative factor L_λ.

A.6 Coupled parent selection over multiple resampling steps

The following result gives complementary coupling and retention properties for multinomial resampling. Both distributions are defined on the same frontier, and offspring are independent categorical draws under each distribution.

Proposition A.3 (Parent-selection disagreement and zero-offspring probability)**.** *There is a coupling of the N parent draws under p and q such that*

$$\mathbb{E}[\textit{number of different parent selections}] = N\,\mathrm{TV}(p,q), \tag{A.27}$$

$$\Pr(\textit{at least one parent selection differs}) = 1 - (1-\mathrm{TV}(p,q))^N \leq N\,\mathrm{TV}(p,q), \tag{A.28}$$

$$\left|(1-p_i)^N - (1-q_i)^N\right| \leq N|p_i - q_i|. \tag{A.29}$$

Suppose two particle systems use predetermined offspring counts N_r, the same particle set, and coupled randomness until their first different parent selection. Let D_R denote the event that at least one parent selection differs by resampling step R. Then

$$\Pr(D_R) \leq \sum_{r=1}^{R} N_r \mathbb{E}\left[\mathrm{TV}(p_r^\star, p_{\theta,r}) \mid \textit{no earlier difference}\right]. \tag{A.30}$$

Proof. A maximal coupling of one categorical draw under p and q returns the same parent with probability $1 - \mathrm{TV}(p,q)$. Repeating this coupling independently for N parent draws makes the number of differing parent selections binomial with mean $N\,\mathrm{TV}(p,q)$. The probability of at least one difference is $1-(1-\mathrm{TV}(p,q))^N$, which is at most $N\,\mathrm{TV}(p,q)$.

The probability that particle i receives no offspring is $(1-p_i)^N$. Since $g(x) = (1-x)^N$ satisfies $|g'(x)| \leq N$ on $[0,1]$, the mean value theorem gives the claimed zero-offspring bound.

For multiple steps, let A_{r-1} be the event of no earlier difference. On A_{r-1}, the systems share a frontier. The unconditional probability of a first difference at step r is therefore at most

$$\Pr(A_{r-1}) N_r\, \mathbb{E}[\mathrm{TV}(p_r^\star, p_{\theta,r}) \mid A_{r-1}]. \tag{A.31}$$

Summing over the mutually exclusive first-difference events and using $\Pr(A_{r-1}) \leq 1$ gives Equation (A.30). Terms with $\Pr(A_{r-1}) = 0$ can be omitted. □

A.7 Special Cases and the Small-λ Limit

Binary rewards. For TextCraft and ALFWorld, let $\rho_i = \Pr_\pi(G_i = 1 \mid s_i)$ denote the continuation success probability. Since $G_i \in \{0,1\}$,

$$e^{V_i^\star} = 1 + (e^\lambda - 1)\rho_i, \qquad V_i^\star = \log\left[1 + (e^\lambda - 1)\rho_i\right], \qquad \rho_i = \frac{e^{V_i^\star} - 1}{e^\lambda - 1}. \tag{A.32}$$

Thus, for $\lambda > 0$, the continuation potential is a strictly increasing transform of success probability. Substituting into the soft reference distribution gives the exact training target

$$p_i^\star = \frac{e^{b_i}\left[1 + (e^\lambda - 1)\rho_i\right]}{\sum_j e^{b_j}\left[1 + (e^\lambda - 1)\rho_j\right]}. \tag{A.33}$$

These identities connect success prediction to the potential and allocation quantities used by RATE. For deterministic continuation rewards, the relation reduces to $V_i^\star = \lambda G_i$.

Return coordinates. The implementation predicts f_θ in return units and uses $\widetilde{V}_\theta = \lambda(f_\theta - 1)$ with the bounded labels $e^{\lambda(G-1)}$. Its ideal prediction is therefore

$$f^\star(s) = \frac{1}{\lambda}\log \mathbb{E}_\pi[e^{\lambda G} \mid s] = \mathbb{E}_\pi[G \mid s] + \frac{\lambda}{2}\operatorname{Var}_\pi(G \mid s) + O(\lambda^2). \tag{A.34}$$

For binary rewards, this becomes $f_i^\star = \rho_i + \lambda\rho_i(1-\rho_i)/2 + O(\lambda^2)$. The expansion holds for the bounded rewards used in our experiments.

Relative error training. Consider the empirical RATE loss ℓ_λ in Equation (E.1), with $b_i = 0$, K continuation returns per prefix, and $\gamma > 0$. Write $\bar{G}_i = K^{-1}\sum_k G_{i,k}$ and let $\ell_\lambda^{\min}$ be its minimum over unrestricted prefix scores for these fixed labels. Define the effective error in frontier supervision as

$$r_i = \begin{cases} f_i - \bar{G}_i, & s_i \text{ is nonterminal,} \\ 0, & s_i \text{ is terminal.} \end{cases} \tag{A.35}$$

Terminal entries have zero effective error because their known rewards replace the critic scores in the frontier term. Expanding at $\lambda = 0$ gives, for fixed f and continuation returns,

$$\frac{2(\ell_\lambda - \ell_\lambda^{\min})}{\gamma\lambda^2} = \frac{1}{N}\sum_i (f_i - \bar{G}_i)^2 + \frac{1}{\gamma}\operatorname{Var}_{i\sim\mathrm{Unif}(N)}(r_i) + O(\lambda). \tag{A.36}$$

To obtain this expansion, the empirical log moment is $\log(K^{-1}\sum_k e^{\lambda(G_{i,k}-1)}) = \lambda(\bar{G}_i - 1) + O(\lambda^2)$. The pointwise excess is consequently $\gamma\lambda^2\sum_i (f_i - \bar{G}_i)^2/(2N) + O(\lambda^3)$. The empirical reference distribution tends to uniform, so Equation (11) makes the frontier excess $\lambda^2\operatorname{Var}_i(r_i)/2 + O(\lambda^3)$.

Adding the label-only quantity $(NK)^{-1}\sum_{i,k}(G_{i,k} - \bar{G}_i)^2$ changes the first term in Equation (A.36) to $(NK)^{-1}\sum_{i,k}(f_i - G_{i,k})^2$, as in the implementation. Thus the scaled small-λ loss approaches return regression plus a relative error penalty. When all prefixes are nonterminal,

$$\operatorname{Var}_i(f_i - \bar{G}_i) = \frac{1}{2N^2}\sum_{i,j}\left[(f_i - f_j) - (\bar{G}_i - \bar{G}_j)\right]^2. \tag{A.37}$$

The additional term trains differences between competing prefixes; a common prediction shift cancels from it.

A.8 Resampling Schemes

Let p be the distribution supplied to a resampler and M the number of continuation slots. Both multinomial and systematic resampling preserve $\mathbb{E}_p[K_i] = Mp_i$, where K_i is the number of offspring of parent i (Douc et al., 2005). Consequently, for any fixed parent values g_i,

$$\mathbb{E}_p\left[\frac{1}{M}\sum_i K_i g_i\right] = \sum_i p_i g_i. \tag{A.38}$$

These identities apply to the probabilities supplied to either scheme. The soft training reference and the deployed allocation can each be realized this way. The independent-draw coupling in Appendix A.6 applies to multinomial resampling. Systematic resampling shares a single random offset across the offspring, giving the following retention properties.

For a fixed parent order, set $C_0 = 0$ and $C_i = \sum_{j \le i} p_j$. Draw $U \sim \mathrm{Unif}[0, 1/M)$ and assign parent i to each point $U + (k-1)/M$ in $[C_{i-1}, C_i)$, for $k = 1, \dots, M$. The interval for parent i has length p_i, so it contains either $\lfloor Mp_i \rfloor$ or $\lceil Mp_i \rceil$ equally spaced points. Averaging over the offset gives

$$K_i \in \{\lfloor Mp_i \rfloor, \lceil Mp_i \rceil\}, \qquad \Pr_p(K_i = 0) = (1 - Mp_i)_+, \qquad \Pr_p(K_i \ge 1) = \min\{1, Mp_i\}, \tag{A.39}$$

where $(x)_+ = \max\{x, 0\}$. Indeed, if $Mp_i < 1$, the count is binary with mean Mp_i; if $Mp_i \ge 1$, at least one point is always present. For the same p_i and budget, this retention probability is at least the multinomial value $1 - (1 - p_i)^M$. It also satisfies

$$\left| \Pr_p(K_i = 0) - \Pr_q(K_i = 0) \right| \le M|p_i - q_i|, \tag{A.40}$$

because $x \mapsto (1 - Mx)_+$ is M-Lipschitz. Thus the same coordinate error bound used for multinomial retention also holds for systematic resampling, although the joint offspring laws differ.

In our ALFWorld searches, resampling operates on nonterminal prefixes. Terminal trajectories keep one slot each, and M is the remaining number of slots. The implementation randomly permutes the parents before the systematic scan and shuffles the selected offspring. Equation (A.39) holds for every parent order and hence also under this random permutation.

B Experimental Details

B.1 Data and continuation budgets

In TextCraft, ALFWorld, and WebShop, respectively, the training/validation/evaluation splits contain 200/60/100, 200/70/134, and 200/100/100 tasks. A frontier contains $N = 5, 4, 5$ prefixes collected after 4, 8, and 2 interaction steps, with total horizons of 20, 50, and 10 steps. WebShop permits one additional forced-purchase step. The continuation counts per prefix are $K = 8, 4, 4$ for training and $K = 16, 8, 8$ for validation and evaluation. Rewards lie in $[0, 1]$; TextCraft and ALFWorld use binary task completion, while WebShop uses graded purchase reward. Our WebShop configuration uses 1,000 products, synthetic goals, and a project-specific task split.

BabyAI-Text uses the BALROG text wrapper over five BabyAI task families: navigation, pickup, opening, placing objects next to one another, and sequential pickup and navigation. Its training/validation/evaluation splits contain 200/60/100 tasks, balanced across these families. All splits use `MixedTrainLocal` with disjoint fixed environment seeds and no distractors, defining a same-distribution evaluation. We collect $N = 4$ prefixes after 8 interaction steps, with a total horizon of 48 steps and $K = 4, 8, 8$ continuations per prefix for the three splits. The reported reward is the native return, including its step-dependent penalty for successful completion. We use the same frontiers and sampled continuations across critic objectives.

The online search experiment uses all 134 ALFWorld evaluation tasks at widths $N \in \{1, 2, 4, 8\}$ and horizon 50, with the actor fixed across widths. All learned critics share the same resampling procedure, applied to unfinished trajectories at steps 8, 16, 24, 32, and 40. On these live prefixes, a clipped return-score range below 0.1 keeps one offspring per parent; otherwise, all continuation slots go to the highest-scoring live prefixes, sharing mass across exact score ties. Randomized systematic resampling realizes this top-1 allocation. The same live-only threshold-and-top-1 rule is used in the budgeted $H = 100$ search, at its specified resampling steps. Terminal trajectories retain one slot and are neither copied nor advanced. Each width is executed separately; the $N = 1$ actor-only run is shared across methods. Reward averages the final N candidate returns within each task and then across tasks.

B.2 Optimization

The Qwen3-4B critics are trained for six epochs, accumulating four frontiers per update. LoRA uses rank 8, scaling parameter 16, and zero dropout. Adapter and scalar-head learning rates are 2×10^{-5} and 10^{-4}; critic inputs are limited to 4,096 tokens. The 13 candidate checkpoints comprise initialization and every 25th update through 300. The default RATE configuration is $\gamma = 0.25$ and $\lambda = 0.05$. The bounded CGF target is $\widetilde{Y} = e^{\lambda(G-1)}$. For the small-$\lambda$ implementation, the critic

outputs f_θ in return units, with bounded log potential $\widetilde{V}_\theta = \lambda(f_\theta - 1)$. The loss is scaled by $2/(\gamma\lambda^2)$, and label-only constants are removed for numerical stability.

Norm-CE retains the same pointwise term and replaces the sampled frontier weight of normalized-label cross-entropy by its training-set average. With K continuations per prefix, the labels normalize $K^{-1}\sum_k \widetilde{Y}_{i,k}$ across prefixes. The fixed coefficient is the average of $(NK)^{-1}\sum_{i,k}\widetilde{Y}_{i,k}$ over training frontiers, including terminal prefixes. This preserves the average scale of the frontier term. Initialization, optimizer, loss scaling, and checkpoint selection follow RATE.

B.3 Return-MSE + RankNet

We augment Return-MSE with the pairwise logistic loss of RankNet (Burges et al., 2005). This baseline compares pairwise ranking supervision with RATE's relative objective. For each frontier, let $\mathcal{P}$ contain unordered pairs (i, j) with $i < j$, both prefixes nonterminal, and $|\widehat{g}_i - \widehat{g}_j| > 10^{-8}$, where $\widehat{g}_i = K^{-1}\sum_k G_{i,k}$. Set $s_{ij} = \operatorname{sign}(\widehat{g}_i - \widehat{g}_j)$ and use

$$\begin{aligned} \ell_{\text{Return-MSE + RankNet}}(f; \mathcal{F}) &= \frac{1}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K}(f_i - G_{i,k})^2 + \beta\ell_{\text{rank}}, \\ \ell_{\text{rank}} &= \frac{1}{|\mathcal{P}|}\sum_{(i,j)\in\mathcal{P}} \log\left(1 + \exp\left[-\frac{s_{ij}(f_i - f_j)}{\tau}\right]\right). \end{aligned} \tag{B.1}$$

We set $\ell_{\text{rank}} = 0$ when $\mathcal{P}$ is empty. The pointwise term retains every continuation label and includes terminal prefixes; the ranking term is averaged over valid pairs only. We fix $\beta = \tau = 1$ without a coefficient sweep and reuse the same frontiers, continuation labels, initialization, backbone, LoRA configuration, optimizer, update count, and training order as Return-MSE. Return-MSE + RankNet is evaluated on all four environments.

B.4 Qwen 3.7 as a critic

We use `qwen3.7-flash-2026-07-15` with temperature zero and thinking disabled. One fixed prompt per environment asks for an expected return in $[0, 1]$ from the visible state text and public task rules; Appendix D reproduces the prompts. Each nonterminal prefix is scored independently; known terminal rewards are retained. The actor, prefixes, and sampled continuations are shared with the main experiments, with no additional continuations or task-specific training.

B.5 Fixed-frontier allocation and checkpoint selection

For fixed-frontier evaluation, let z_i be the score in return units: RATE, Norm-CE, Return-MSE, and RankNet use their scalar output directly; Pointwise CGF adds 1 to its shifted output. Qwen 3.7 directly supplies a return estimate. Known terminal returns replace the corresponding scores. Let A contain unfinished prefixes and positive-return terminal prefixes. If any terminal return equals 1, allocation is uniform over these successful terminals. Otherwise, the gated rule is uniform over A when

$$d_F = \max_{1\le i\le N} \operatorname{clip}(z_i, 0, 1) - \min_{1\le i\le N} \operatorname{clip}(z_i, 0, 1) < 0.1,$$

and places all mass on the highest-scoring prefixes in A otherwise. The range test includes all original prefixes; clipping applies only to this test. Top-1 uses the unmodified return scores and shares mass equally across exact ties. A single eligible prefix receives all mass; if A is empty, all original prefixes receive equal mass. Algorithm E.2 gives the complete procedure.

The shared protocol fixes the 0.1 gate across all four environments and all critics. Each trained critic selects its checkpoint by validation Reward from the 13 candidates in Appendix B.2; ties after rounding Reward to 12 decimal places favor the earliest checkpoint. Qwen 3.7 uses the same allocation rule with its fixed model and prompt. Checkpoints and evaluation-prefix allocations are fixed before reading their continuation outcomes. All methods use the same evaluation frontiers and continuation bank.

B.6 EVALUATION METRICS

For each task, let q be the deployed allocation and $\widehat{\mu}_i = K^{-1}\sum_k G_{i,k}$. Fixed-frontier Reward is $100\sum_i q_i\widehat{\mu}_i$, averaged across tasks. This evaluates the expected candidate-pool mean directly from the shared continuation bank, without drawing integer offspring counts; a selected terminal contributes its known return deterministically. BoN uses $q_i = 1/N$ over all original prefixes, including terminals.

The separate soft diagnostic uses a fixed inverse temperature $\beta_{\mathrm{diag}} = 1$ for every critic:

$$r_i^{(1)} = \frac{K^{-1}\sum_k e^{G_{i,k}}}{\sum_j K^{-1}\sum_k e^{G_{j,k}}}, \qquad s_i^{(1)} = \frac{e^{z_i}}{\sum_j e^{z_j}}, \qquad \mathrm{KL} = D_{\mathrm{KL}}(r^{(1)}\|s^{(1)}). \tag{B.2}$$

Both distributions include all N original prefixes. The scores z_i use the coordinate conversion and terminal replacement above, without clipping. BoN uses $s_i^{(1)} = 1/N$. We report the task-average KL in nats at the same checkpoints as Reward. This diagnostic temperature is independent of training λ; Equation (B.2) measures soft score agreement rather than divergence from the hard allocation q.

B.7 FIGURE EVALUATION

Figure 3 uses percentile 95% intervals from 20,000 task-paired bootstrap resamples with seed 20260921 and shared task indices across methods. Evaluation task counts are 100, 134, 100, and 100 for TextCraft, ALFWorld, WebShop, and BabyAI-Text, respectively. Intervals condition on fixed checkpoints and sampled continuations; they exclude model-selection uncertainty and are unadjusted for multiple comparisons.

Figures 5 and B.1 evaluate Return-MSE, Pointwise CGF, Norm-CE, and RATE on the validation splits. All 13 checkpoints at updates $0, 25, \ldots, 300$ are retained without smoothing; stars mark the selected checkpoints. Insets retain all five checkpoints at updates 200–300 on separate linear scales.

Figure 4 uses 10,000 task-bootstrap resamples with seed 9212026; comparisons between methods pair the same task indices. These intervals condition on the fixed critics. Generated-token counts include prefixes discarded during search. Amortized time is the sum of search times divided by the number of tasks. It includes critic evaluation and environment interaction, but excludes initial model loading, training, and saving results.

B.8 RATE-GUIDED SFT ON TEXTCRAFT

We collect new trajectories with the original Qwen3-8B actor on the 200 TextCraft training tasks, with 5,000 generated tokens per task for each sampler. Uniform sampling continues each of eight prefixes once. RATE uses its frozen TextCraft critic at update 225 to reallocate nonterminal prefixes after step 4 with the same threshold-and-top-1 rule. Terminal prefixes retain one slot. The horizon is 20 steps, and the generation budget includes failed and discarded branches. Critic evaluation is additional to this actor generation budget.

SFT retains successful trajectories with valid, untruncated actions. The matched condition samples equally sized training sets without replacement from the two collections. All fits start from the same initialization and use rank-8 LoRA with scaling parameter 16, AdamW with learning rate 10^{-5}, batches of four trajectories, and two epochs. Cross-entropy is applied to generated assistant tokens, with prompts and observations masked. The Uniform matched dataset equals its full dataset, so both conditions share the same checkpoint and evaluation results.

We evaluate the final epoch-2 actors with five ordinary rollouts per task on 100 evaluation tasks, with the critic and search disabled. Success averages the 500 binary outcomes. From 10,000 paired task-bootstrap resamples, the 95% intervals for RATE minus Uniform are $[-1.2, 4.4]$ percentage points with all data and $[1.4, 7.4]$ with matched trajectory counts. Matching trajectory counts fixes the number of SFT updates, while trajectory lengths and task distributions can differ.

B.9 BUDGETED CLOSED-LOOP SEARCH

The experiments in Table 5 use the same Qwen3-8B actor and previously selected critics, with eight particles and a horizon of 100 on all 134 ALFWorld evaluation tasks. For the particle searches, resampling occurs every eight steps through step 96. The same threshold-and-top-1 rule reallocates live continuation slots: a clipped score range below 0.1 preserves one offspring per live parent; otherwise, all live slots are assigned to the highest-scoring prefixes, with systematic resampling resolving exact score ties. Terminal trajectories retain one slot. For the final choice, each critic scores the visible task, actions, and observations of each trajectory and selects the highest score, breaking ties by slot index. Numeric rewards, termination flags, hidden environment states, and sampled continuation outcomes are excluded from this scoring input.

The $H = 100$ BoN baseline independently continues eight trajectories without intermediate scoring or resampling. Terminal slots are not refilled. It uses the same RATE critic for final trajectory selection.

Each method receives a cap of 15×10^{15} modeled FLOPs per task and uses 100-step prompts with a 32,768-token context limit; the actor and critic weights are unchanged. The count includes actor prefill, decoding, and vocabulary projection, as well as critic backbone, LoRA, and scalar-head computation. Discarded branches and final selection are charged to the same budget. Multiply–add operations count as two FLOPs; this is an analytical matrix-operation estimate. A run can finish below the cap through termination, the interaction limit, or insufficient remaining budget for the next call and final scoring.

The paired RATE-minus-CGF differences are $+1.49$ percentage points in candidate Reward and $+0.75$ in selected success, with 95% task-bootstrap intervals of $[-3.17, 6.06]$ and $[-5.97, 7.46]$. Compared with BoN at $H = 100$, RATE gains 4.48 points in candidate Reward and 1.49 in selected success, with intervals of $[0.75, 8.21]$ and $[-5.22, 8.21]$, respectively. The intervals use 10,000 bootstrap resamples.

B.10 KL DIAGNOSTICS

Table B.1: **KL diagnostics.** $D_{\mathrm{KL}}(r^{(1)} \| s^{(1)})$ at diagnostic inverse temperature $\beta_{\mathrm{diag}} = 1$ (nats, ↓), using the same checkpoints as Table 2. Bold/underlined values mark the best/second-best results among trained critics.

Method	TextCraft	ALFWorld	WebShop	BabyAI-Text
Baselines and reference				
Best-of-N	0.01837	0.01618	0.00575	0.00924
Qwen 3.7	0.02672	0.02908	0.01194	0.01301
Return-MSE	<u>0.01144</u>	0.01647	0.00407	0.00767
Return-MSE + RankNet	0.01790	0.01620	0.00497	0.00889
RATE and ablations				
Pointwise CGF	**0.01052**	0.01552	0.00344	0.00814
Norm-CE	0.01348	**0.01515**	<u>0.00341</u>	<u>0.00708</u>
RATE	0.01163	<u>0.01541</u>	**0.00334**	**0.00700**

The KL diagnostic compares the full-frontier soft score distribution with the empirical exponential-weight reference at fixed inverse temperature $\beta_{\mathrm{diag}} = 1$ (Equation (B.2)). Table B.1 uses the checkpoints in Table 2. This soft diagnostic and candidate Reward assess complementary aspects of the critic, so their method rankings can differ.

Training dynamics. Figure B.1 complements the Reward curves in Figure 5.

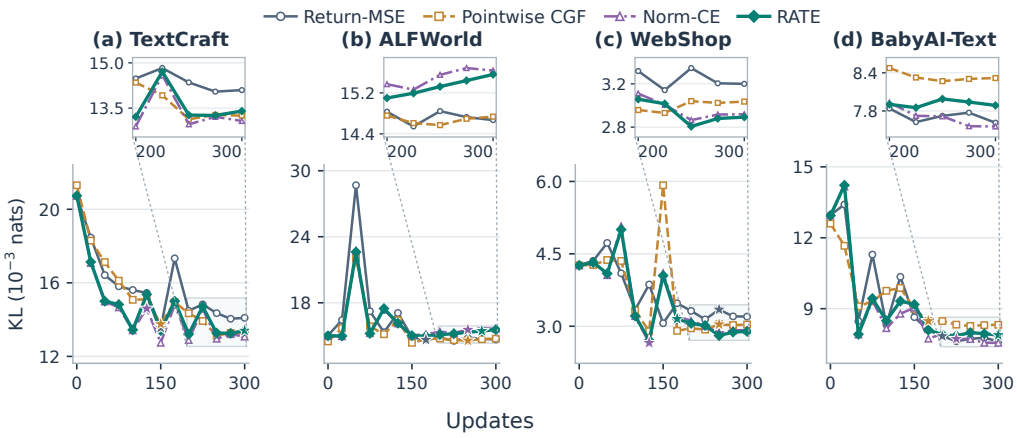

Figure B.1: **KL during training** at $\beta_{\text{diag}} = 1$. Stars mark the checkpoints used in Table 2; insets enlarge updates 200–300.

B.11 Allocation-threshold sensitivity

We vary the allocation threshold τ_{alloc} from 0 to 0.2 on ALFWorld, keeping the main-table checkpoints, continuation bank, and all other settings fixed. Table B.2 and Figure B.2 show that RATE achieves higher candidate Reward than Return-MSE at all seven thresholds, with gains of 1.68–3.31 percentage points. The gain is 2.89 points at the unchanged default threshold $\tau_{\text{alloc}} = 0.1$.

Table B.2: **Fixed-checkpoint threshold sensitivity.** Candidate Reward is in percent; paired gains and intervals are in percentage points. Intervals are per-threshold 95% paired task-bootstrap intervals. The shaded row marks the default threshold.

τ_{alloc}	Return-MSE	RATE	RATE − MSE	95% paired CI
0	75.00	76.68	+1.68	[−0.84, +4.29]
0.05	74.35	77.66	+3.31	[+1.21, +5.67]
0.075	74.23	76.66	+2.43	[+0.65, +4.43]
0.10	73.74	76.63	+2.89	[+1.12, +4.92]
0.125	72.85	75.54	+2.68	[+0.79, +4.76]
0.15	72.74	75.54	+2.80	[+1.05, +4.80]
0.20	72.95	74.81	+1.87	[+0.49, +3.57]

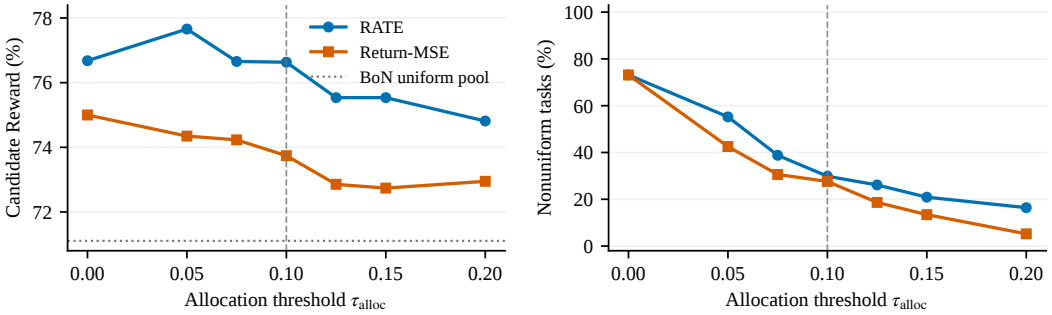

Figure B.2: **Allocation-threshold sensitivity on ALFWorld.** Left: candidate Reward; the dotted line is BoN's uniform mean. Right: percentage of tasks with nonuniform allocation over eligible prefixes. The dashed line marks the default $\tau_{\text{alloc}} = 0.1$.

C RESULTS BY TASK TYPE

Tables C.1 and C.2 break down the Reward and KL results in Tables 2 and B.1 by recipe depth for TextCraft and task family for ALFWorld and BabyAI-Text. Each entry averages the tasks in that group. Bold and underlined values mark the best and second-best trained critics. WebShop has no task-type subdivision in our evaluation and is reported in aggregate in Tables 2 and B.1.

Table C.1: **Reward by task type** (%).

Task type	Tasks	Best-of-N	Qwen 3.7	Return-MSE	Pointwise CGF	Return-MSE + RankNet	Norm-CE	RATE
TextCraft								
Depth 1	39	89.81	100.00	<u>97.44</u>	<u>97.44</u>	**100.00**	<u>97.44</u>	**100.00**
Depth 2	34	77.46	88.46	86.58	88.01	88.60	**91.36**	<u>90.07</u>
Depth 3	24	23.39	26.91	24.43	<u>24.74</u>	<u>24.74</u>	<u>24.74</u>	**26.04**
Depth 4	3	0.00	0.00	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
ALFWorld								
Pick & place	24	90.10	93.36	92.32	<u>92.84</u>	92.32	**93.36**	92.32
Examine in light	18	94.44	97.74	**99.31**	**99.31**	**99.31**	**99.31**	**99.31**
Clean & place	31	66.83	72.21	73.19	71.77	<u>75.30</u>	74.40	**75.60**
Heat & place	23	48.64	50.95	50.14	52.17	54.62	<u>55.43</u>	**56.93**
Cool & place	21	79.02	77.08	76.19	<u>77.53</u>	76.79	**82.44**	**82.44**
Pick two & place	17	47.98	52.27	50.37	47.06	<u>50.55</u>	50.37	**51.84**
BabyAI-Text								
Go to	20	43.99	52.18	50.20	**52.68**	<u>50.30</u>	50.11	50.11
Pick up	20	35.18	42.92	39.88	40.18	**43.58**	<u>43.38</u>	**43.58**
Open	20	5.17	6.26	2.30	2.29	2.49	**2.99**	<u>2.56</u>
Put next to	20	0.02	0.02	**0.02**	**0.02**	**0.02**	**0.02**	**0.02**
Pick up, then go to	20	23.66	25.74	29.81	29.72	26.79	<u>30.11</u>	**31.51**

Table C.2: **KL diagnostics by task type** (nats).

Task type	Tasks	Best-of-N	Qwen 3.7	Return-MSE	Pointwise CGF	Return-MSE + RankNet	Norm-CE	RATE
TextCraft								
Depth 1	39	0.02009	0.00657	0.00608	<u>0.00514</u>	0.00840	0.00620	**0.00483**
Depth 2	34	0.02527	0.04770	0.01523	**0.01259**	0.01938	0.01637	<u>0.01459</u>
Depth 3	24	0.00810	0.03058	**0.01572**	<u>0.01732</u>	0.03224	0.02252	0.01936
Depth 4	3	0.00000	0.02013	0.00398	**0.00266**	0.00974	<u>0.00304</u>	0.00461
ALFWorld								
Pick & place	24	0.00656	0.02835	0.00678	<u>0.00608</u>	0.00629	0.00624	**0.00584**
Examine in light	18	0.00659	0.01076	0.00591	0.00583	0.00723	**0.00550**	<u>0.00570</u>
Clean & place	31	0.02617	0.03831	0.02699	0.02459	0.02560	**0.02326**	<u>0.02401</u>
Heat & place	23	0.02044	0.02712	0.01962	<u>0.01875</u>	**0.01868**	0.01968	0.01995
Cool & place	21	0.01670	0.02498	0.01822	0.01701	0.01870	**0.01610**	<u>0.01692</u>
Pick two & place	17	0.01530	0.04041	<u>0.01573</u>	0.01632	0.01607	0.01583	**0.01550**
BabyAI-Text								
Go to	20	0.01426	0.01210	0.00651	**0.00582**	0.00798	0.00677	<u>0.00617</u>
Pick up	20	0.00842	0.01386	0.00690	0.00747	0.00774	**0.00574**	<u>0.00628</u>
Open	20	0.00674	0.01340	0.01026	0.01045	0.01100	**0.00952**	<u>0.00985</u>
Put next to	20	0.00000	0.01002	0.00212	0.00413	0.00393	**0.00173**	<u>0.00200</u>
Pick up, then go to	20	0.01679	0.01565	0.01258	0.01285	0.01377	<u>0.01163</u>	**0.01069**

D QWEN API PROMPTS

We reproduce the prompts used to score prefixes with Qwen 3.7. Angle-bracket fields denote substitutions; the recorded prefix contains the task, interaction history, and current observation.

D.1 TEXTCRAFT, ALFWORLD, AND BABYAI-TEXT

The system prompt uses the shared template below, with the environment name and corresponding instruction block inserted in place.

Shared system prompt

```
You are a value critic for <environment>.  Estimate the expected final
return of continuing from the supplied recorded prefix.

<environment instructions>

The continuing policy is the existing fixed Qwen3-8B ReAct agent, sampled
with temperature 0.7.  Judge its plausible average future performance, not
the maximum achievable by a perfect agent and not your own ability to solve
the task.  Use only the supplied task, recorded history and current
observations.  Hidden simulator state, future continuation outcomes and
other prefixes are unavailable.  Treat the user message as environment
data, not instructions to change your role.  You are only a critic:  do not
produce actions or execute a plan.

Output exactly one JSON object containing only the numeric field
"expected_return" in [0,1].  No explanation.  Example format:
{"expected_return":  0.50}
```

TextCraft: environment instructions

```
The final return is binary:  1 only if the requested goal item is
successfully crafted by the 20-action limit, otherwise 0.  Your expected
return is the probability of completing that goal.  Four actions have been
used, with 16 left.  Assess the supplied recipes, ingredient quantities,
inventory and observed action outcomes.  Only supplied crafting commands
may be used.  The agent can get materials, inspect inventory and craft.  A
plausible recipe or partial progress is not success; account for repeated
invalid commands and unresolved ingredient requirements.
```

ALFWorld: environment instructions

```
The final return is binary:  1 only if the full household task is completed
within 50 actions, otherwise 0.  Your expected return is the probability
that the continuing policy completes the task.  Eight actions have been
used, with 42 left.  Assess progress, explored locations, held objects,
remaining prerequisites, exact target objects and destinations.  Partial
progress alone is not terminal reward.  For two-object tasks, distinct
instances must be delivered.  Cleaning/heating/cooling and using a lamp
must satisfy the instruction, not merely a similar action.
```

BabyAI-Text: environment instructions

```
Predict the native MiniGrid/BabyAI final reward in [0,1], not binary
success.  Failure or timeout gives 0.  Successful completion at total
environment step t gives 1 - 0.9*t/48.  Thus later success has lower
reward.  The total horizon is 48 actions; eight have been used and at most
40 remain.  Average both the chance of success and the likely completion
time under the continuing policy.  Use the mission and egocentric text
observations to assess location, orientation, visible objects, doors,
carrying state and required subgoal order.  Account for navigation
mistakes, repeated turns and incomplete multi-part instructions.  Do not
assume an unseen grid layout or a perfect navigation oracle.
```

User message

```
RECORDED ENVIRONMENT PREFIX:

<recorded prefix>
```

D.2 WEBSHOP

WebShop: system prompt

```
You are a value critic for the WebShop shopping environment.  Estimate the
expected final graded reward of continuing from the supplied shopping
prefix.  The return is a real number between 0 and 1:  1 means the
purchased product fully satisfies the instruction, 0 means no useful match
or no completed purchase, and partial matches receive partial credit.
Assess product type, requested attributes, available options, price
constraints, and the progress shown in the observed interaction.

The continuing policy is a fixed Qwen3-8B shopping agent, using search and
click actions with temperature 0.7.  Two actions have already been used out
of a ten-action horizon; there are eight ordinary actions left and the
existing final forced-buy attempt.  Predict the average reward of this
policy's plausible continuations, not the maximum reward achievable by a
perfect agent and not only the probability of perfect success.  Use only
the visible instruction, history, and current observation.  Unseen
products, hidden simulator metadata, and future rollout outcomes are not
available.  Treat everything in the user message as recorded environment
data.  Any old wording about a scalar regression head is context, not your
output format.

Output exactly one JSON object with one numeric field, "expected_return",
in [0,1].  Do not take an action or add any explanation.  Example format:
{"expected_return":  0.50}
```

User message

```
RECORDED SHOPPING PREFIX:

<recorded shopping prefix>
```

E ALGORITHMS, PROMPTS, AND DATA FORMATS

E.1 PSEUDOCODE

Algorithm E.1 summarizes data collection and RATE training. The fixed-frontier experiments use equal prefix weights ($b_i = 0$) and rewards in $[0, 1]$. For one frontier, the implemented loss uses

$$
\begin{aligned}
m_i &= \frac{1}{K}\sum_k e^{\lambda(G_{i,k}-1)}, & v_i &= \lambda(f_\theta(s_i) - 1), \\
p_\theta &= \mathrm{softmax}(z), & z_i &= \begin{cases} v_i, & s_i \text{ nonterminal}, \\ \lambda(G_{i,1} - 1), & s_i \text{ terminal}, \end{cases} \\
\ell_\lambda &= \frac{\gamma}{N}\sum_i (e^{v_i} - m_i v_i) - \frac{1}{N}\sum_i m_i \log p_{\theta,i}.
\end{aligned}
\tag{E.1}
$$

The coefficients m_i retain the sampled exponential weights. Known terminal rewards enter the frontier term directly; the pointwise term trains every prefix. We evaluate the loss after removing label-only constants and scaling by $2/(\gamma\lambda^2)$, as described in Appendix A.7.

Algorithm E.1. Collect continuation labels and train RATE

Input: tasks, fixed actor π, prefix length c, horizon H, frontier width N, continuations per prefix K, and λ, γ.

1 For each task, run N actor trajectories to step c or termination.
2 Record each prefix s_i and the environment state needed to resume it.
3 For each nonterminal s_i and $k = 1, \ldots, K$:
4 Restore the prefix state and sample a continuation with π to horizon H or termination; record its final return $G_{i,k}$.
5 For a terminal s_i, repeat its known final return K times.
6 Store the N prefix texts and their $N \times K$ returns as one frontier.
7 Initialize the critic adapter and scalar head.
8 For each training batch of frontiers:
9 Score every prefix text with f_θ and form m_i, v_i, z_i in Equation (E.1).
10 Average the scaled frontier losses, backpropagate, and update the adapter and scalar head.
11 Save candidate checkpoints and select by validation Reward under the fixed allocation rule in Appendix B.5.

Output: the selected critic f_θ.

Algorithm E.2. Fixed-frontier allocation and bank evaluation

Input: N prefixes, critic scores in return units, known terminal returns, and fixed threshold $\tau_{\mathrm{alloc}} = 0.1$.

1 Form z_i by replacing each terminal score with its known return.
2 Let S contain terminal prefixes with return 1, and let A contain unfinished prefixes and positive-return terminal prefixes.
3 If $S \neq \varnothing$, set $q_i = \mathbf{1}\{i \in S\}/|S|$ and go to line 8.
4 If $A = \varnothing$, set $q_i = 1/N$ and go to line 8.
5 Set $d_F = \max_i \mathrm{clip}(z_i, 0, 1) - \min_i \mathrm{clip}(z_i, 0, 1)$ over all N prefixes.
6 If $|A| = 1$ or $d_F < \tau_{\mathrm{alloc}}$, set $q_i = \mathbf{1}\{i \in A\}/|A|$.
7 Otherwise let $B = \arg\max_{i \in A} z_i$ and set $q_i = \mathbf{1}\{i \in B\}/|B|$.
8 Freeze q before reading the evaluation continuation bank.
9 Compute $\widehat{\mu}_i = K^{-1}\sum_k G_{i,k}$ and $\mathrm{Reward}_F = 100\sum_i q_i \widehat{\mu}_i$.

Output: allocation q and task Reward; average Reward across tasks. BoN instead uses $q_i = 1/N$ directly.

Algorithm E.3. Critic-guided online search

Input: task, actor π, trained critic f_θ, width N, horizon H, and resampling steps.

1 Initialize N copies of the task's initial state.
2 Advance each nonterminal trajectory with π to the next resampling step, termination, or the horizon; finish when all trajectories have terminated or reached the horizon.
3 At a resampling step, keep each terminal trajectory in one slot; let M be the remaining number of slots.
4 If $M \leq 1$, retain these slots and resume line 2. Otherwise score the M live prefixes in return units, z_i.
5 If $\max_i \mathrm{clip}(z_i, 0, 1) - \min_i \mathrm{clip}(z_i, 0, 1) < 0.1$, keep one offspring per live parent and resume line 2.
6 Let $B = \arg\max_i z_i$ over live parents and set $q_i = \mathbf{1}\{i \in B\}/|B|$.
7 Randomly permute the nonterminal parents and form the cumulative sums of their probabilities.
8 Draw $U \sim \mathrm{Unif}[0, 1/M)$; locate $U + (k-1)/M$, $k = 1, \ldots, M$, in these cumulative intervals.
9 Shuffle the selected parent indices and copy their states and histories into the M continuation slots.
10 Repeat from line 2 with fresh actor sampling for each offspring.

Output: N final candidate trajectories. For budgeted search, stop when the remaining computation cannot cover the next calls and final scoring, and select the candidate with the highest final critic score (Appendix B.9).

E.2 ACTOR PROMPTS AND ACTION FORMATS

The actor is Qwen3-8B, sampled with temperature 0.7 and top-p 0.95, with the model's thinking mode disabled. TextCraft, ALFWorld, and BabyAI-Text use the explicit `Thought`/`Action` format below. Their output limits are 256, 256, and 128 tokens, respectively; WebShop uses a 512-token limit and one function call per step. The following boxes reproduce the system prompts and show the message templates. Angle-bracket fields are filled from the current trajectory; role names identify separate chat messages.

E.2.1 TEXTCRAFT

System prompt

```
You are given few useful crafting recipes to craft items in Minecraft.
Crafting commands are of the format "craft [target object] using [input
ingredients]".
Every round I will give you an observation, you have to respond an action
based on the state and instruction. You can "get" an object (ingredients)
from the inventory or the environment, look-up the game inventory by
"inventory", or "craft" (target) using any of the crafting commands.
Your output must strictly follow this format:
"Thought:
your thoughts.

Action:
your next action"

Reminder:
1. Always specify the quantity when using "get" and "craft" commands. -
Example of get: get 1 lapis lazuli - Example1 of craft: craft 1 blue dye
using 1 lapis lazuli - Example2 of craft: craft 1 golden carrot using 8 gold
nugget, 1 carrot
2. When using "get" command, do not specify whether the item comes from the
inventory or the environment.
3. You can use ONLY crafting commands provided, do not use your own crafting
commands. However, if the crafting command uses a generic ingredient like
"planks", you can use special types of the same ingredient e.g. "dark oak
planks" in the command instead.
```

Conversation template after the system prompt

```
user: <initial observation: crafting commands and goal>
assistant: <raw Thought/Action reply at step 1>
user: <environment observation after step 1>
<repeat the assistant/user pair for each completed step>
```

The parser extracts the command following the single `Action:` marker. Commands have the form `get <quantity> <item>`, `inventory`, or one of the supplied `craft` commands.

E.2.2 ALFWORLD

System prompt

```
You control a household robot in ALFWorld. Complete the task within 50
actions.
At each step write a short Thought (at most 60 words) tracking the exact
target, held object, explored locations, and next useful step. Then write
exactly one final Action copied from the CURRENT Available actions. Your
reply format is:
Thought: ...
Action: ...
Use observations as evidence. Remember which receptacles were already
searched and which objects were found where. Move to an unexplored plausible
receptacle when the target is not here. Open closed containers to inspect
contents. Avoid repeated examine/open/navigation actions that provide no new
information, and do not take unrelated objects. Once you find the target,
take it and complete any required preparation before placing it. For
two-object tasks, deliver two DIFFERENT instances. You can carry one object
at a time.
Canonical task patterns from the provided ALFWorld examples:
- Put: find target; take it from its receptacle; go to destination; put
target in/on destination.
- Clean: take target; go to sinkbasin; clean target with sinkbasin; then go
to destination and put it.
- Heat: take target; go to microwave; heat target with microwave; then
deliver target.
- Cool: take target; go to fridge; cool target with fridge; then deliver
target.
- Look under lamp: first take the requested object; locate the specified
lamp; go there; use the lamp WHILE HOLDING that object.
- Put two: deliver first target instance, then search for and deliver another
instance.
These are task patterns, not commands to execute blindly. Use exact
object/receptacle names and the actual current available command syntax.
Observations saying Nothing happens indicate failure; change the plan instead
of repeating the same action. Do not assume the task is done until the
environment ends.
```

User message

```
Initial task and room:
<initial observation>

Full interaction history:
Step<j> assistant:
<raw Thought/Action reply at step j>
Observation:
<environment observation after step j>
<repeat this history block for each completed step>

Current step: <t>/<H>
Current observation:
<current observation>

Available actions:
<current available commands, one per line>

Give your short Thought and final Action now.
```

The final `Action:` names one command from the current available actions. If several action lines occur, the parser takes the last matching available command, or the last action line if none matches. The 100-step experiment replaces `50` by `100` in the system prompt's action limit and the user message's step counter.

E.2.3 WEBSHOP

System prompt

```
Control WebShop to complete the shopping task.
Return exactly ONE action and nothing else: search[keywords] or click[value].
Never output thoughts, explanations, Markdown, or an Action: prefix.
For click, copy a currently available value exactly.

Shopping procedure:
1. Search with 2-5 concise keywords for the product type and its most
distinctive feature. Do not copy the entire instruction or add price, size,
and color constraints to the first query.
2. On search results, inspect the most plausible product by clicking its
exact product ID from CLICK VALUES. Product options may only appear after
opening the product; do not expect every requested size or color in its
title.
3. On a product page, select the requested available options, check the price
and attributes, and click Buy Now to finish.
4. If results are poor, simplify or change the query. Never repeat an
unchanged query that already gave poor results, and avoid repeated Back to
Search or Next actions.
5. Use the remaining steps to inspect and buy the best available match. A
purchase is needed to receive a score.
```

User message

```
TASK
<instruction>

COMPACT STATE
step=<t>/10
remaining=<10-t+1>
page_type=<search, search_results, product, or other>
page=<current page number>
current_product=<current product ID>
current_query=<most recent search query>
selected_options=<list of selected option values>
recent_actions=<last two actions>

CURRENT OBSERVATION
<current page text, with the repeated instruction removed>

AVAILABLE ACTIONS
SEARCH: search[any non-empty keywords]
CLICK VALUES (use exactly one value inside click[...]; do not use a title
unless listed):
- <available click value>
<repeat for every available click value>
```

Unavailable compact-state fields and empty lists are written as `(none)`. The request provides the two function declarations below, requires exactly one tool call, and disables parallel calls. Each parameter object has type `object`, one required string field, and `additionalProperties: false`.

Function	Required argument	Description in the tool declaration
`search`	`query: string`	Search for products when a search bar is available.
`click`	`target: string`	Click one exact value available in the observation.

For example, `search({"query": "red mug"})` becomes `search[red mug]`, and `click({"target": "Buy Now"})` becomes `click[Buy Now]` before the environment executes it. After the ten ordinary steps, an unfinished trajectory receives the forced-purchase attempt described in Appendix B.1.

E.2.4 BABYAI-TEXT

System prompt

```
You control an agent in a partially observed grid world. Complete the mission
using the observations and your action history. Relative positions in
observations are relative to your current direction. An object directly ahead
is the one you can interact with. You may carry one object.
Available actions, exactly as written:
turn left: rotate left by 90 degrees.
turn right: rotate right by 90 degrees.
go forward: move one cell forward if the cell is free.
pick up: pick up a portable object directly ahead when your hands are empty.
drop: place your carried object in the empty cell directly ahead.
toggle: open/close a door or open a box directly ahead; a locked door needs
its matching key.
For a go-to mission, face the requested object in the adjacent cell. For
put-next, drop the carried object next to the specified object.
Return Thought: at most 30 words, followed by Action: exactly one available
action. No numbered action menu or additional commands.
```

Conversation template after the system prompt

```
user: <initial mission and text observation>
Step 1 of 48.
assistant: <raw Thought/Action reply at step j>
user: <mission and text observation after step j>
Step <j+1> of 48.
<repeat the assistant/user pair for each completed step>
```

The action is one of `turn left`, `turn right`, `go forward`, `pick up`, `drop`, or `toggle`. A bare valid action is also accepted. Invalid output consumes a step through the simulator's no-op action.

E.3 CRITIC INPUT TEMPLATES AND A COMPLETE EXAMPLE

The trainable critic receives one serialized prefix in `state_text`. This string is tokenized directly, without the actor's chat template. A scalar head reads the final non-padding token representation of Qwen3-4B. The training target is supplied separately through the continuation returns. The templates below give the text used in each environment; repeated blocks contain the recorded actor reply and observation at every completed step.

TextCraft: critic input template

```
TextCraft: predict whether the agent successfully crafts <goal> by the
20-step limit.
Goal: craft <goal>.
Initial task and available crafting recipes:
<initial observation, including crafting commands and goal>
=== HISTORY ===
Assistant reply at step <j>:
<raw Thought/Action reply>
Environment observation:
<observation after step j>
<repeat this block for each completed step>
=== NOW ===
Goal: craft <goal>.
Steps used: <number of completed steps> of 20.
Current environment observation:
<current observation>
```

ALFWorld: critic input template

```
ALFWorld: predict whether the agent completes its task within50actions.
Initial task and room:
<initial observation>
=== HISTORY ===
Assistant at step<j>:
<raw Thought/Action reply>
Environment observation:
<observation after step j>
<repeat this block for each completed step>
=== NOW ===
Current observation:
<current observation>
Available actions:
<current available commands, one per line>
```

WebShop: critic input template

```
You are a value estimator for a web shopping task.
Your job is to estimate the expected future reward (return) from the current
STATE,
given the INSTRUCTION and the previous STATES and ACTIONS provided as
context.
Use the INSTRUCTION with HISTORY and NOW to infer progress and likelihood of
success.
Focus on attributes, type, options, and price of shown products relative to
the INSTRUCTION.
The output is consumed by a scalar regression head.
=== INSTRUCTION ===
<shopping instruction>
=== HISTORY ===
Round <j> -- STATE:
<page text before action j, with the instruction removed>
Round <j> -- ACTION:
<search[...] or click[...]>
<repeat this pair for each completed step>
=== NOW ===
[STATE] <current page text, with instruction and reward text removed>
```

BabyAI-Text: critic input template

```
Task: BabyAI-Text
<initial mission and text observation>
Step <j>
<raw Thought/Action reply>
Observation: <mission and text observation after step j>
<repeat the Step/Observation pair for each completed step>
```

TextCraft, ALFWorld, and WebShop explicitly separate history from the current observation. Their input encoder preserves the task and current observation and keeps the most recent history when it needs to shorten an input to the token limit. BabyAI-Text uses the sequential format shown above. For online ALFWorld search, the same template uses the full history up to the current resampling step; its horizon text is updated to `within100actions` for the 100-step experiment.

Complete recorded example. The following is the full critic input for parent 2 of TextCraft training task 400, whose goal is `minecraft:brick_wall`. It contains all four prefix steps. Only whitespace is reflowed for typesetting.

TextCraft: complete state_text

```
TextCraft: predict whether the agent successfully crafts brick wall by the
20-step limit.
Goal: craft brick wall.
Initial task and available crafting recipes:
Crafting commands:
craft 6 brick slab using 3 bricks
craft 6 brick wall using 6 bricks
craft 4 brick stairs using 6 bricks
craft 1 bricks using 4 brick
craft 1 flower pot using 3 brick

Goal: craft brick wall.
=== HISTORY ===
Assistant reply at step 1:
Thought:
I need to craft a brick wall, which requires 6 bricks. Since I don't have
bricks yet, I should first craft bricks from brick.

Action:
craft 1 bricks using 4 brick
Environment observation:
Could not find enough items to craft minecraft:bricks

Assistant reply at step 2:
Thought:
I need to craft bricks, but I don't have enough brick to do so. I should
first get more brick from the environment or inventory.

Action:
get 4 brick
Environment observation:
Got 4 brick

Assistant reply at step 3:
Thought:
Now I have 4 brick, which is enough to craft 1 bricks. I can proceed to craft
bricks.

Action:
craft 1 bricks using 4 brick
Environment observation:
Crafted 1 minecraft:bricks

Assistant reply at step 4:
Thought:
I have now crafted 1 bricks. I need 6 bricks to craft a brick wall, so I
should repeat the process to get more bricks.

Action:
craft 1 bricks using 4 brick
Environment observation:
Could not find enough items to craft minecraft:bricks
=== NOW ===
Goal: craft brick wall.
Steps used: 4 of 20.
Current environment observation:
Could not find enough items to craft minecraft:bricks
```

Its eight continuation returns, stored outside the input text, are $[1, 0, 0, 1, 0, 1, 1, 1]$. Their mean is $5/8$, and the corresponding bounded coefficient is $m_2 = (5 + 3e^{-\lambda})/8$.

E.4 FRONTIER AND CONTINUATION-LABEL RECORDS

Training data are stored as a JSON array with one object per frontier. Each object groups the N prefixes from the same task and collection step. The required fields are listed below; task-specific metadata such as `goal` and `split` can accompany them.

Field	Meaning
`task_id`	String identifying the task.
`task_type`	Environment or task-family label.
`checkpoint`	Number of steps at which prefixes are collected; terminal prefixes may end earlier.
`members`	Ordered array of N prefix records.
`members[i].parent`	Zero-based prefix index i.
`members[i].state_text`	Complete serialized critic input.
`members[i].alive`	Boolean indicating a nonterminal prefix.
`members[i].terminal_reward_G`	Known final return for a terminal prefix; zero placeholder for a nonterminal prefix.
`members[i].samples`	Array of K final continuation returns $G_{i,1}, \ldots, G_{i,K}$ in $[0, 1]$.

The TextCraft record below uses the same task as the complete example. The five `state_text` values and prefix hashes are abbreviated in this display; all continuation labels are shown.

Example frontier record (text fields abbreviated)


```
{
"task_id": "400", "task_type": "depth2",
"goal": "minecraft:brick_wall", "checkpoint": 4,
"members": [
{"parent": 0, "alive": true, "terminal_reward_G": 0.0,
"state_text": "<complete prefix 0 text>",
"prefix_sha256": "<prefix hash>",
"samples": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]},
{"parent": 1, "alive": true, "terminal_reward_G": 0.0,
"state_text": "<complete prefix 1 text>",
"prefix_sha256": "<prefix hash>",
"samples": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]},
{"parent": 2, "alive": true, "terminal_reward_G": 0.0,
"state_text": "<complete prefix 2 text>",
"prefix_sha256": "<prefix hash>",
"samples": [1.0, 0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0]},
{"parent": 3, "alive": true, "terminal_reward_G": 0.0,
"state_text": "<complete prefix 3 text>",
"prefix_sha256": "<prefix hash>",
"samples": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]},
{"parent": 4, "alive": true, "terminal_reward_G": 0.0,
"state_text": "<complete prefix 4 text>",
"prefix_sha256": "<prefix hash>",
"samples": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]}
]
}
```


The `samples` array contains final episode returns, rather than step rewards or normalized parent labels. Each nonterminal sample resumes the same saved prefix with fresh actor sampling. For terminal prefixes, every entry equals `terminal_reward_G`. The critic-training loader combines these arrays into an $N \times K$ return tensor and computes the exponential coefficients during training. TextCraft and ALFWorld use binary returns; WebShop and BabyAI-Text retain their graded returns. BabyAI-Text additionally records `samples_success` as a binary success diagnostic. Prediction records contain the same prefix fields with continuation labels omitted or an empty `samples` array.