OPTS-TTPO: ENHANCING FINITE-SAMPLE POLICY-GRADIENT LEARNING WITH TREE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

The policy-gradient theorem expresses the exact gradient as an expectation under the current policy, but practical methods estimate it from finitely many on-policy trajectories. Rare high-return trajectories may be missing from the policy-gradient estimate. We study whether tree search can improve their coverage under a finite budget while controlling the induced gradient bias. We introduce On-Policy Parallel Tree Search (OPTS) and Tree Trajectory Policy Optimization (TTPO) through *on-policy tree trajectories*, which sample new suffixes from the current policy at visited states. This requires no action-distribution correction, but branching changes state visitation. Our Branch Aggregation Lemma underpins TTPO: branch-weighted tree statistics recover chain expectations when branch choices and weights are fixed before outgoing transitions are sampled. OPTS selects expansion states using estimated performance differences. Under deterministic dynamics, exact values, and max-backup advantages, the induced search policy's expected return improves monotonically with the budget. Adaptive expansion violates the lemma's condition; we bound its gradient bias and show that max backup adds prefix credit to actions leading to better discovered suffixes. Against a finite chain reference, TTPG's measured bias stays near its no-branching level while NaivePG's bias grows from 0.1251 to 0.4884. Exact-value search is monotone and learned-critic summaries improve in aggregate. At matched budgets, reward- and value-guided OPTS improve correct-answer coverage and majority-vote accuracy over i.i.d. baselines. The coverage–bias diagnostic shows a larger gain from OPTS + TTPG at a smaller bias shift than Fixed-branch + NaivePG. Under matched interaction or rollout budgets, OPTS-TTPO improves MuJoCo tail returns over PPO by up to 28.6%, raises Atari-57 human-normalized IQM from 0.357 to 0.374, and improves micro-averaged `pass@32` across all four Qwen3 models.

1 INTRODUCTION

The policy-gradient theorem expresses the exact gradient as an expectation over trajectories sampled from the current policy (Sutton et al., 1999). Practical algorithms such as PPO estimate this expectation using a finite batch of on-policy trajectories (Schulman et al., 2017). Under independent sampling, each trajectory starts from the initial-state distribution and provides one sampled suffix from each visited state. Collecting more independent trajectories can improve the gradient estimate, but rare high-return trajectories may still go unobserved under a finite sampling budget. This motivates studying how to allocate the available budget to improve high-return trajectory coverage for policy-gradient learning.

This work targets a central finite-sample failure of policy-gradient learning: a finite sample of trajectories can miss rare, high-return trajectories. We use tree search to improve the chance of covering such trajectories while controlling the bias introduced by adaptive search. Tree search offers a way to use the available budget more selectively (Browne et al., 2012). Whereas every independent trajectory must start from the initial-state distribution, search can select an informative state on an already sampled prefix and draw fresh suffixes from that state. A critic-derived estimate can identify visited states with the greatest estimated room for improvement and concentrate new samples there. This mechanism changes where samples are collected, not how actions are generated: every new action is still drawn from the current policy. Therefore, the resulting suffixes remain on-policy in their conditional action distribution and require no action-distribution importance correction.

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.

Core challenges. Turning this intuition into a policy-gradient algorithm is not straightforward. First, *how should the learner aggregate a tree?* Branching duplicates prefixes and gives different states different numbers of descendants, so treating every node as an ordinary sample over-represents heavily expanded regions and generally changes the policy-gradient target, even when the advantage estimate at each transition is unbiased. Second, *where should a finite search budget be spent?* Uniform branching wastes the very advantage of search; the algorithm needs a policy-relative criterion that identifies the states at which resampling is most valuable. Third, useful allocation is necessarily adaptive: the search chooses where to expand after seeing sampled outcomes. Such posterior-dependent selection conflicts with the conditions required for unbiased tree aggregation. Thus, adaptive search creates a genuine tradeoff between improved trajectory coverage and additional estimation bias.

Our solution. We propose OPTS–TTPO, our framework that combines two methods introduced in this work: On-Policy Parallel Tree Search (OPTS) and Tree Trajectory Policy Optimization (TTPO). To make tree data usable for policy-gradient learning, we first prove the Branch Aggregation Lemma: when branching decisions and normalized local weights depend only on prefix information, a branch-weighted tree statistic has the same expectation as its ordinary on-policy chain counterpart. We then derive TTPO from this lemma, weighting each transition by its derived branch weight in the clipped PPO objective so that heavily expanded regions are not over-counted. To decide where to search, we introduce OPTS, which uses a critic-derived, policy-relative performance difference to estimate the benefit of replacing a sampled suffix with one generated by the current policy. OPTS directs the finite budget toward the highest-scoring visited states and samples new on-policy suffixes from them in parallel; under deterministic dynamics, exact current-policy values, and max-backup advantages, we prove that its induced search policy improves monotonically as the budget grows. Finally, we characterize what changes when OPTS and TTPO are combined: posterior-adaptive expansion violates the lemma's prefix-measurability condition. We decompose the resulting bias into posterior trajectory reweighting and the additional prefix credit introduced by max backup, which propagates the value of better suffixes back to their prefixes, and derive bounds for both effects. In stochastic environments, we instead use mean backup to reduce the amplification of favorable noise.

In summary, our contributions are fourfold: **(1) Principled learning from trees.** We prove the Branch Aggregation Lemma and introduce TTPO, whose branch-weighted updates recover the ordinary on-policy expectation under the lemma's conditions (Section 3.1). **(2) Performance-directed search.** We introduce OPTS, a performance-difference-guided budget-allocation method with a monotonic search-improvement guarantee under ideal conditions (Section 3.2). **(3) An explicit account of adaptivity.** We decompose and bound the bias of learning from posterior-adaptive OPTS trees with TTPO into trajectory reweighting and max-backup prefix credit (Section 3.3). **(4) Theory-backed cross-domain validation.** Controlled mechanism experiments validate the theoretical results underlying OPTS–TTPO. Under matched budgets, OPTS–TTPO improves MuJoCo tail return by 18.7% on average, raises Atari-57 human-normalized IQM from 0.357 to 0.374, and improves LLM micro-averaged `pass@32` by 0.44–2.33 percentage points across four models (Section 4).

2 Related Work

Adaptive allocation and high-return discovery. Finite-budget policy-gradient learning and high-return discovery have been studied through adaptive data collection. Robust On-Policy Sampling addresses finite-sample mismatch in policy evaluation, while PROPS extends adaptive sampling to policy-gradient control (Zhong et al., 2022; Corrado & Hanna, 2023); Adaptive Experience Selection changes replay selection to reduce gradient-estimator variance (Mohamad & Montana, 2020). Go-Explore and Self-Imitation Learning reuse promising states or past high-return trajectories (Ecoffet et al., 2021; Oh et al., 2018). For inference-time high-return discovery, repeated sampling provides a simple independent-sampling baseline (Brown et al., 2024); recent RLVR methods operate at different allocation levels: VIP allocates rollouts across prompts, PROS reuses historical prefixes, and TRACE allocates additional suffixes at intermediate prefixes (Nguyen et al., 2026; Huang & Wan, 2026; Zou et al., 2026). These methods allocate computation at different units and for different targets; we focus on high-return trajectory coverage under a fixed rollout budget and its policy-gradient interface.

Tree search as budget allocation. Tree search uses posterior information to allocate search computation. MCTS formalizes selection, expansion, simulation, and backup (Kocsis & Szepesvári, 2006;

Browne et al., 2012); AlphaGo combines learned policy and value networks with MCTS for action selection (Silver et al., 2016). AlphaGo Zero, AlphaZero, MuZero, Expert Iteration, and related methods instead distill search-improved action distributions or trajectories into a policy (Silver et al., 2017; 2018; Schrittwieser et al., 2020; Anthony et al., 2017; Hubert et al., 2021; Danihelka et al., 2022; Grill et al., 2020; Feng et al., 2023). The tree is therefore commonly used to choose a root action, construct a search or execution policy, or provide a policy-projection target.

Tree-structured policy-gradient learning. Tree-structured policy-gradient methods use branches for value estimation or credit assignment. Tree MDPs, VinePPO, and Branching Policy Optimization use whole-tree, suffix, or sibling returns (Scavuzzo et al., 2022; Kazemnejad et al., 2025; He et al., 2026); TreeRL, SPO-tree, TreeRPO, TreePO, and Tree-GRPO variants derive process or subtree signals from descendant outcomes (Hou et al., 2025; Guo et al., 2025b; Yang et al., 2025b; Li et al., 2025; Ji et al., 2026; Ding & Ye, 2026). Other methods choose branches with uncertainty, lookahead, or information gain, or optimize search-internal policies and tree backups (Dong et al., 2026; Xing et al., 2026; Zong et al., 2026; Wu et al., 2026; Zhang et al., 2026; Graf & Platzner, 2016; Zhang et al., 2019; Farquhar et al., 2018; Morimura et al., 2024; Dalal et al., 2025; Soemers et al., 2019; Tian et al., 2025). Their branching signals and learning targets differ: for example, TreeRL samples on-policy suffixes but uses group-relative process credit, while other methods optimize search-internal policies, persistent-tree policies, or subtree-derived objectives.

OPTS-TTPO targets finite-budget high-return trajectory coverage. OPTS allocates current-policy suffix rollouts to visited states using a policy-relative performance-difference estimate; TTPO learns from the resulting tree with branch-weighted aggregation, preserving the chain target under prefix-measurable expansion. We further analyze adaptive-selection bias and max-backup prefix credit.

3 OPTS-TTPO

OPTS-TTPO uses on-policy tree trajectories as the common interface between search and learning (Figure 1). Branch aggregation is the foundation for learning from prefix-measurable trees, while performance-difference-guided OPTS allocates the finite rollout budget. We characterize the prefix credit and gradient bias that arise when posterior-adaptive OPTS is combined with TTPO. Complete definitions and proofs are given in Appendices A–C.

On-policy tree trajectories. An on-policy tree trajectory $\mathcal{T} = (s_o, \mathcal{X}(\mathcal{T}))$ consists of a root state $s_o \sim \rho_0$ and a finite rooted set of transition occurrences. Starting from s_o, it recursively samples one or more transitions according to π_θ and extends their successor states to the horizon; suffixes from a common state occurrence share the prefix ending there. Each transition c from parent occurrence p is (s_p, a_c, r_c, s_c), with $a_c \sim \pi_\theta(\cdot \mid s_p)$ and $(r_c, s_c) \sim P(\cdot \mid s_p, a_c)$, and has a unique path from s_o. For transition index x, we write (s_x, a_x, r_x, s_{x^+}). Here, *on-policy* refers only to the conditional action distribution $\pi_\theta(\cdot \mid s_p)$: every action occurrence in the tree is sampled from the current policy, so attaching suffixes requires no action-distribution importance correction. Branching nevertheless changes transition multiplicities relative to a single chain. The Branch Aggregation Lemma below derives weights that recover the chain-trajectory target under prefix-measurable branching.

3.1 TTPO: How Should the Learner Aggregate a Tree?

Branch Aggregation Lemma. Policy-gradient and GAE targets are chain-trajectory sums, whereas a tree can attach several suffixes to one shared prefix. As a simple example, if three suffixes branch at depth t, naive summation counts the suffix three times relative to the prefix; averaging the three suffixes restores the chain proportions. The lemma generalizes this correction to arbitrary trees.

For a horizon-n chain $\tau = (x_0, \ldots, x_{n-1})$, define $H_\eta(\tau) := \sum_{t=0}^{n-1} \eta^t h(x_t)$ with scalar η and integrable transition contribution h. The target is $\mathbb{E}_{\tau\sim\pi_\theta}[H_\eta(\tau)]$.

For each parent occurrence p, let its child set $\mathcal{C}(p)$ and local weights $\alpha_{p,c} \geq 0$, with $\sum_{c\in\mathcal{C}(p)} \alpha_{p,c} = 1$, be determined from prefix information. Define $W(o) = 1$ and $W(c) = W(p)\alpha_{p,c}$. This prefix-measurability condition excludes dependence on the outgoing transitions or their descendants.

Let $d(x)$ be the transition depth, starting at zero. Direct aggregation sums the weighted contribution of every transition occurrence in the tree; recursive aggregation computes the discounted statistic

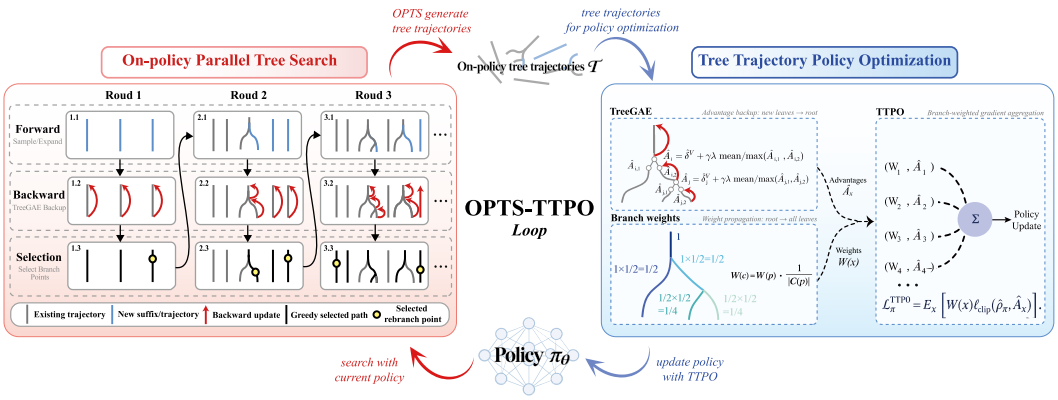


Figure 1: OPTS-TTPO. OPTS selects states maximizing the length-penalized performance-difference estimate and samples on-policy suffixes from them in parallel; TTPO applies TreeGAE and branch weights to PPO updates.

of the subtree rooted at s_p by averaging child contributions with their local weights, with zero at terminal:

$$\widehat{H}_{\eta,\mathrm{dir}}(\mathcal{T}) := \sum_{x\in\mathcal{T}} W(x)\eta^{d(x)}h(x), \qquad \widehat{R}_\eta(s_p) := \sum_{c\in\mathcal{C}(p)} \alpha_{p,c}\big[h(c) + \eta\widehat{R}_\eta(s_c)\big]. \tag{1}$$

Under this prefix-measurability condition, the Branch Aggregation Lemma (Appendix A.3) gives $\widehat{H}_{\eta,\mathrm{dir}}(\mathcal{T}) = \widehat{R}_\eta(s_o)$ and

$$\mathbb{E}_{\mathcal{T}}\left[\widehat{H}_{\eta,\mathrm{dir}}(\mathcal{T})\right] = \mathbb{E}_{\tau\sim\pi_\theta}\left[H_\eta(\tau)\right]. \tag{2}$$

Tree Trajectory Policy Gradient (TTPG). Setting $\eta = \gamma$ and $h(x) = A^{\pi_\theta}(s_x, a_x)\nabla_\theta \log \pi_\theta(a_x \mid s_x)$ in the direct aggregation form gives

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathcal{T}}\left[\sum_{x\in\mathcal{T}} W(x)\gamma^{d(x)}A^{\pi_\theta}(s_x,a_x)\nabla_\theta \log \pi_\theta(a_x \mid s_x)\right]. \tag{3}$$

The weights $W(x)$ correct the state-visitation multiplicity introduced by branching.

Tree-based Generalized Advantage Estimation (TreeGAE). Let $\delta_x^V := r_x + \gamma V(s_{x^+}) - V(s_x)$. Setting $\eta = \gamma\lambda$ and $h(x) = \delta_x^V$ in the recursive aggregation form gives

$$\widehat{A}_x^{\mathrm{TreeGAE}(V)} := \delta_x^V + \gamma\lambda \sum_{x^+\in\mathcal{C}(x)} \alpha_{x,x^+}\widehat{A}_{x^+}^{\mathrm{TreeGAE}(V)}. \tag{4}$$

TreeGAE and chain GAE have the same conditional suffix expectation; with $V = V^{\pi_\theta}$, Equation 3 preserves the policy-gradient identity.

Tree Trajectory Policy Optimization (TTPO). Let $\rho_x(\theta) := \pi_\theta(a_x \mid s_x)/\pi_{\theta_{\mathrm{old}}}(a_x \mid s_x)$. TTPO applies the branch weights to the standard clipped PPO surrogate,

$$\mathcal{L}_\pi^{\mathrm{TTPO}}(\theta) := \mathbb{E}_x\left[W(x)\min\Big(\rho_x(\theta)\widehat{A}_x, \mathrm{clip}(\rho_x(\theta), 1-\epsilon, 1+\epsilon)\,\widehat{A}_x\Big)\right]. \tag{5}$$

3.2 OPTS: Where Should a Finite Search Budget Be Spent?

Performance-Difference Estimation. Let $\tau = (s_t, a_t, r_t, \ldots, s_n) \sim (\mu, P)$ be a recorded suffix under an arbitrary policy μ. Replacing its action at s_k by a fresh action from the current policy π and then following π has expected local improvement $V^\pi(s_k) - Q^\pi(s_k, a_k) = -A^\pi(s_k, a_k)$. Their discounted sum defines the *performance-difference estimate*, whose expectation follows from the performance difference lemma (Kakade & Langford, 2002; Schulman et al., 2015a):

$$\Delta(s_t;\tau) := -\sum_{k=t}^{n-1}\gamma^{k-t}A^\pi(s_k,a_k), \qquad \mathbb{E}_{\tau\sim(\mu,P)|s_t}[\Delta(s_t;\tau)] = V^\pi(s_t) - V^\mu(s_t). \tag{6}$$

In deterministic environments, $\Delta(s_t;\tau) = V^\pi(s_t) - G_t$ for recorded suffix return G_t. In practice, replacing A^π by TreeGAE gives $\widehat{\Delta}(s_t;\tau) := -\sum_{k=t}^{n-1} \gamma^{k-t}\widehat{A}_{x_k}$, and we introduce the length penalty $\widehat{\Delta}^{(\xi)}(s_t;\tau) := \widehat{\Delta}(s_t;\tau)/(n-t)^\xi$, $\xi \in [0,1]$. When $\xi = 0$, this recovers the original performance-difference estimate; when $\xi = 1$, it averages the discounted local improvements along the suffix. Appendix B.2 gives further details.

On-Policy Parallel Tree Search. OPTS initializes on-policy chains and their TreeGAE advantages. Each round follows $c^\star(p) \in \arg\max_{c\in\mathcal{C}(p)} \widehat{A}_c$ to form an observed greedy path $\tau^\star$, then selects its highest-scoring candidate position:

$$s^\star \in \arg\max_{s_t\in\tau^\star} \widehat{\Delta}^{(\xi)}(s_t;\tau^\star). \tag{7}$$

A single batched rollout under the current policy generates suffixes from selected states and fresh trajectories from new initial states; TreeGAE is then updated from the new leaves back to the roots. A tree stops searching when it reaches $S_{\max}$ expansions or when no candidate exceeds a nonnegative baseline. Algorithm 1 gives the complete procedure.

In deterministic environments, OPTS uses max-backup TreeGAE:

$$\widehat{A}_{\max}(x) := \delta_x^V + \gamma\lambda \max_{c\in\mathcal{C}(x)} \widehat{A}_{\max}(c), \qquad \max\varnothing := 0.$$

The greedy paths of OPTS induce a search policy π_j^S, with $\pi_0^S = \pi$ (Appendix B.3). With exact $V = V^\pi$ and max-backup TreeGAE in a deterministic environment, OPTS satisfies

$$J(\pi_{j_2}^S) \geq J(\pi_{j_1}^S) \geq J(\pi), \qquad 0 \leq j_1 \leq j_2 \leq S_{\max}.$$

Appendix B.4 also establishes monotonicity of the suffix λ-return objective under nested expansion.

3.3 OPTS-TTPO: What Are the Benefits and Costs of the Adaptive Allocation?

At iteration u, OPTS samples each suffix from the current policy but selects expansion states from observed outcomes, violating the lemma's prefix-measurability condition; TTPO then learns from the branch-weighted policy gradient of tree trajectories, $\pi_u \xrightarrow{\text{OPTS}} \mathcal{T}_u \xrightarrow{\text{TTPO}} \pi_{u+1}$.

In deterministic environments, OPTS-TTPO uses max-backup TreeGAE, which yields the prefix-credit term below; in stochastic environments, it uses mean-backup TreeGAE to average child advantages while retaining branch coverage. Both variants use $\alpha_{p,c} = 1/|\mathcal{C}(p)|$ to correct branch multiplicity.

Max-backup search information as prefix credit. On a fixed tree, let $\widehat{g}_{\max}$ and $\widehat{g}_{\text{mean}}$ be the TTPG directions under the two backups, and define the best-minus-average gap $b(x) := \max_{c\in\mathcal{C}(x)} \widehat{A}_{\max}(c) - \sum_{c\in\mathcal{C}(x)} \alpha_{x,c}\widehat{A}_{\max}(c) \geq 0$, with $b(x) = 0$ at leaves. Writing $x \preceq u$ when x lies on the root-to-u path, Appendix C.1 proves

$$\widehat{g}_{\max} = \widehat{g}_{\text{mean}} + \nabla_\theta \mathcal{L}_{\text{search}}(\theta)|_{\theta=\theta_{\text{old}}},$$
$$\mathcal{L}_{\text{search}}(\theta) := \gamma\lambda \sum_{u\in\mathcal{T}} W(u)\gamma^{d(u)} b(u) \sum_{x\preceq u} \lambda^{d(u)-d(x)} \log \pi_\theta(a_x \mid s_x). \tag{8}$$

$\mathcal{L}_{\text{search}}$ credits actions leading to each branch point by its gap $b(u)$, with the credit decaying by λ at each step, so a discovered suffix can influence upstream learning before being reliably reproduced.

Controlling the search-induced bias. Let p be the fraction of root trees receiving an expansion and $\rho_S \leq p(1 - 2^{-S_{\max}})$ bound the expected path weight moved off the initial trajectory. Under bounded rewards, values, and policy scores, Appendix C.2 gives

$$\|\overline{g}_{\max} - \nabla J(\theta_{\text{old}})\| \leq \varepsilon_0 + \min\{2pC,\ 2\rho_S C + \varepsilon_{\text{backup}}\}, \tag{9}$$

Here C bounds tree and chain gradient norms, ε_0 is the initial chain-estimator bias (zero with exact values), and $\varepsilon_{\text{backup}}$ is bounded by the weighted gap mass in Equation 8. The coarse bound scales with p; the refined bound separates the posterior path reweighting shared by both backup rules, $2\rho_S C$, from the additional max-backup term $\varepsilon_{\text{backup}}$. Algorithm 2 gives the complete training step.

4 EXPERIMENTS

We evaluate branch aggregation, OPTS search and test-time scaling, coverage–bias trade-offs, and cross-domain policy-gradient learning (Sections 4.1–4.4). Rollout budgets are matched in LLM experiments and environment steps in control experiments; learned policies are evaluated without search. Full protocols and hyperparameters are in Appendix D.

4.1 BRANCH AGGREGATION FOR POLICY-GRADIENT ESTIMATION

Setup. Using a frozen step-400 `Qwen3-1.7B` PPO checkpoint, we use unbiased zero-baseline advantage estimates from binary Monte Carlo returns, isolating branch aggregation from critic error. We compare NaivePG with TTPG on identical trees against a finite reference pooled from 32 independently sampled chain groups, varying additional suffixes $K \in \{0, 1, 3, 7, 15\}$ and tree groups $M \in \{1, 2, 4\}$ (Appendix D.1).

Results. At $K = 0$ the two estimators coincide with relative bias 0.1251 (Figure 2). As suffixes are added, NaivePG bias grows monotonically to 0.4884 at $K = 15$, while TTPG stays between 0.1221 and 0.1260. Although reusing descendant tokens lowers NaivePG's variance, its squared bias dominates: at $M = 4$ and $K = 15$, NaivePG's relative MSE rises to 0.2398 whereas TTPG's falls from 0.0285 to 0.0239, with cosine similarities of 0.9467 and 0.9880, respectively. Both estimators use the same unbiased advantage estimates; their divergence reflects NaivePG's over-counting of branched suffixes relative to shared prefixes. Branch weighting corrects this imbalance.

Takeaway 1. Against the finite chain reference, NaivePG's measured bias grows with branching, whereas TTPG stays at its $K = 0$ level and attains lower MSE and higher gradient alignment. Unbiased advantage estimates alone do not guarantee unbiased gradient estimates.

4.2 OPTS SEARCH QUALITY AND TEST-TIME SCALING

Exact-value improvement and learned-critic trends. We test Theorem 2 on 32 depth-4 deterministic binary-tree configurations with exact V^π, max backup, and zero baseline (Appendix D.2). Reward- and value-guided OPTS use full and truncated advantages, respectively. Exact enumeration across $\lambda \in \{0, 0.3, 0.6, 0.95\}$ shows that all 1,920 adjacent-budget comparisons per mode improve both expected λ-return and true return (Figure 3, top).

The learned-critic counterpart fixes a step-400 `Qwen3-1.7B` OPTS-TTPO policy and critic, uses $\lambda = 0.999$, and varies $S_{\max} \in \{0, 1, 3, 7, 15\}$. At $S_{\max} = 15$, the micro-averaged improvements in (J_λ, J) are $(0.0706, 0.1190)$ for reward-guided OPTS and $(0.0107, 0.0254)$ for value-guided OPTS (Figure 3, bottom). All four pooled summaries increase with the search budget, although the gains are not uniformly monotone on every benchmark (Appendix F). The gain also depends on the rebranching position (Appendix G).

Coverage and test-time scaling at matched rollout budgets. We evaluate OPTS with fixed $S_{\max} = 3$ against independent sampling at matched rollout budgets $k \in \{8, 16, 32, 64, 128\}$; per-benchmark curves are in Appendix E.

With verifier rewards guiding rebranching, we report whether a searched tree contains a correct answer; the matched baseline is `pass@k`. Reward-guided OPTS rises from 0.7561 at $k = 8$ to 0.8259 at $k = 128$, compared with 0.7295 and 0.8082 for independent sampling (Figure 4(a)).

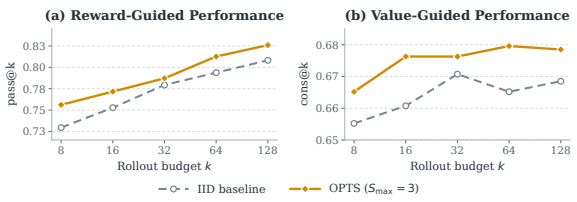


Figure 4: Matched-budget coverage and accuracy, pooled over 902 prompts. OPTS uses fixed $S_{\max} = 3$; the IID curve is the matched baseline.

Without verifier rewards, OPTS uses the learned value function and truncated GAE to guide rebranching and returns the majority answer among the value-greedy tree responses; the baseline is `cons@`k, the majority-vote accuracy of k i.i.d. samples (Wang et al., 2022). Value-guided OPTS rises from

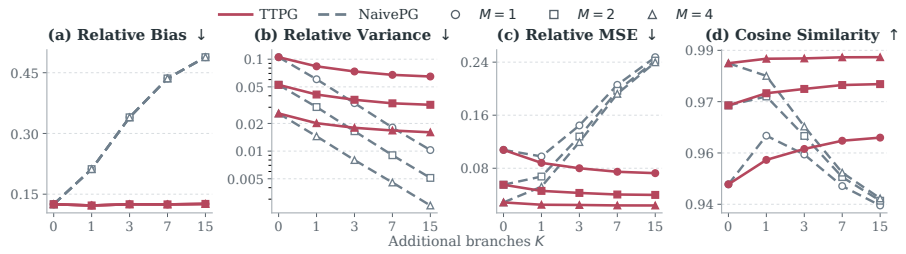


Figure 2: NaivePG versus TTPG under global token-level normalization.

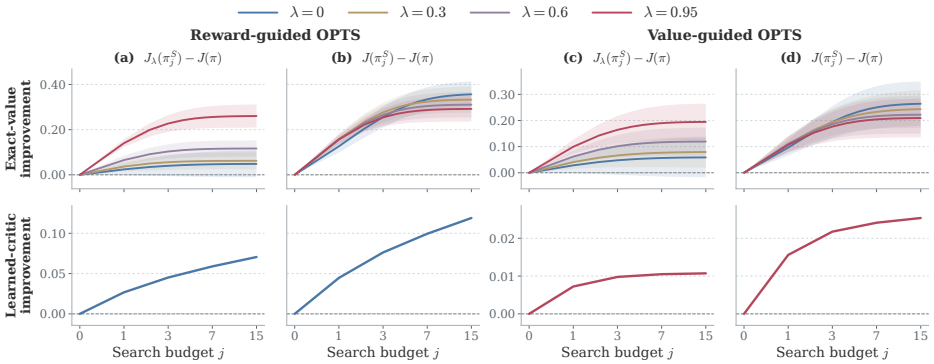


Figure 3: Search-budget improvements. Top: exact-value; bottom: learned critic.

0.6652 at $k = 8$ to 0.6785 at $k = 128$ (peaking at 0.6796 at $k = 64$), around the corresponding IID values 0.6552 and 0.6685 (Figure 4(b)).

Takeaway 2. Larger search budgets improve J_λ and J under exact values, with the same aggregate trend under learned critics; at matched budgets, reward-guided and value-guided OPTS improve coverage and majority-vote accuracy, respectively, over independent sampling.

4.3 Coverage–Bias Trade-offs and Prefix Credit

Using a frozen step-400 `Qwen3-1.7B` OPTS-TTPO actor–critic checkpoint distinct from the PPO checkpoint above, we evaluate 16,384 training prompts. Fixed branching occurs at response token 128 and skips shorter responses; the other method uses OPTS. We compare $s \in \{0, 1, 3, 7\}$ with $p = 1$. Both panels use $M = 1$ aggregation and direct returns with $V = 0$; the right panel additionally compares max and mean backup.

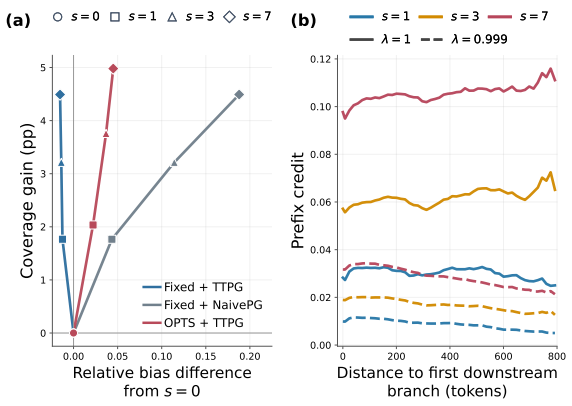


Figure 5: Mechanistic diagnostics. (a) Coverage gain versus relative bias difference from $s = 0$. (b) Token-weighted extra prefix credit (max−mean) versus distance to the first downstream branch.

In panel (a), Fixed + TTPG gains coverage while its bias shift stays near zero (slightly negative), whereas Fixed + NaivePG moves steadily to larger positive bias. OPTS + TTPG reaches the largest coverage gain with a much smaller bias shift than Fixed + NaivePG. Panel (b) shows that max backup assigns more prefix credit after more search rounds ($s = 7 > s = 3 > s = 1$). With $\lambda = 0.999$, this credit decays with distance from the first downstream branch; the $\lambda = 1$ curves retain the credit over the plotted distances. Together, the panels illustrate the coverage–bias relationship and the additional prefix credit introduced by max backup.

Takeaway 3. At matched branch counts, OPTS + TTPG gains more coverage at lower bias than fixed-branch + naivePG, while max backup adds additional prefix credit.

4.4 Cross-Domain Policy-Gradient Learning

We compare OPTS-TTPO with PPO in three domains that differ in observation modality, action space, and budget unit. MuJoCo and Atari follow the CleanRL style (Huang et al., 2022); deterministic MuJoCo and LLM use max-backup TreeGAE, while sticky-action Atari uses mean backup.

Continuous Control: MuJoCo. We select the hyperparameters on Hopper-v4 and Humanoid-v4, the two tasks with the smallest and largest action-space dimensions in our five-task MuJoCo suite (Todorov et al., 2012), respectively. We then fix $(\xi, S_{\max}) = (0.6, 1)$ for the five-task evaluation and train PPO and OPTS-TTPO for one million environment steps over ten random seeds (Figure 6).

OPTS-TTPO achieves a higher tail mean on all five tasks, with gains from 11.2% on `Hopper-v4` to 28.6% on `HalfCheetah-v4`, and a higher full-training mean on every task.

Component ablations. At the fixed $(\xi, S_{\max}) = (0.6, 1)$, the complete method achieves the largest task-averaged tail improvement over PPO (Table 1). Random + mean tests branching without guidance, Guided + mean adds adaptive selection, Guided + max adds max backup, and the unweighted row removes branch weights. Individual component gains vary by task.

Table 1: MuJoCo component ablations. Tail-return change relative to PPO (%), averaged over ten seeds at $(\xi, S_{\max}) = (0.6, 1)$. Tail return averages the final 100 logged points; Mean weights tasks equally.

Variant	Hopper	Walker2d	HalfCheetah	Ant	Humanoid	Mean
Random + mean	+10.5	+5.6	-3.2	**+15.0**	-1.3	+5.3
Guided + mean	+8.2	+16.7	+9.5	-0.6	+6.3	+8.0
Guided + max, unweighted	+7.5	+6.8	+26.2	+8.7	+8.2	+11.5
Guided + max (full)	**+11.2**	**+24.9**	**+28.6**	+13.9	**+15.0**	**+18.7**

Stochastic Discrete Control: Atari-57. We reuse the same $(\xi, S_{\max}) = (0.6, 1)$ configuration for Atari. Under the Atari-57 evaluation benchmark (Bellemare et al., 2013) with sticky actions (Machado et al., 2018), we train PPO and OPTS-TTPO for ten million environment steps over three random seeds. Because reward scales differ across games, we transform each run's return as $(R - R_{\text{random}})/(R_{\text{human}} - R_{\text{random}})$ and report the interquartile mean (IQM), the 25% trimmed mean over all 57 games and three seeds (Agarwal et al., 2021). Table 2 pairs this magnitude-sensitive summary with task-level win counts under the full-training mean return and the last-100-log mean return; complete curves are in Appendix I.

OPTS-TTPO has a higher observed IQM under both summaries, increasing from 0.247 to 0.255 over the full training run and from 0.357 to 0.374 over the last 100 logged points. It wins on 31 and 34 of the 57 games, respectively, with one tie under the latter summary. PPO remains better on 26 and 22 games, indicating that the gains depend on the task and its reward structure rather than holding uniformly across Atari-57. Appendix J compares mean and max backup at the same fixed hyperparameters.

Table 2: Human-normalized IQM and task-level win counts on Atari-57. Each win compares the two methods after averaging the corresponding raw-return summary over three seeds for that game.

	Human-normalized IQM		Task wins		
Metric	PPO	OPTS-TTPO	PPO	OPTS-TTPO	Tie
Full-training mean return	0.247	**0.255**	26	**31**	0
Last-100-log mean return	0.357	**0.374**	22	**34**	1

Mathematical Reasoning: LLM RLVR. In LLM RLVR (Guo et al., 2025a), we compare OPTS-TTPO with PPO (Schulman et al., 2017) on four Qwen3 models (Yang et al., 2025a), using VeRL (Sheng et al., 2025; Moritz et al., 2018) and 4,096 rollouts per update. OPTS-TTPO uses $S_{\max} = 3$ for all four models. DAPO (Yu et al., 2025) and REINFORCE++ (Hu et al., 2025) are external references at the same rollout budget; DAPO uses GRPO's group-relative objective (Shao et al., 2024). All methods run for 400 steps. Table 3 reports final-checkpoint `avg@32` and `pass@32` on six benchmarks, including MATH500 (Hendrycks et al., 2021) and AIME25 (Zhang & Math-AI, 2025).

OPTS-TTPO improves both macro- and micro-averaged `avg@32` and `pass@32` over PPO across all four Qwen3 models, covering both model sizes and both Base and post-trained variants. Among

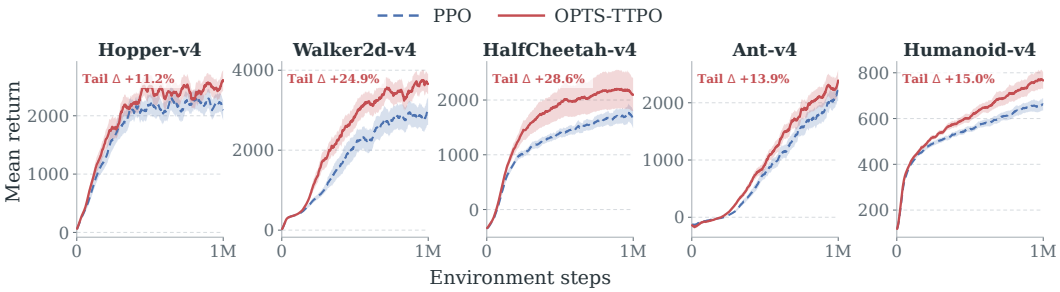


Figure 6: MuJoCo learning curves; tail Δ is the final-100-point relative return improvement over PPO.

Table 3: LLM performance at the synchronized step-400 checkpoint, from 32 independent responses per prompt. Macro Average weights the six benchmarks equally; Micro Average pools their 500, 272, 40, 30, 30, and 30 problems, respectively (902 in total). Column bests are bold.

Methods	MATH500		MinervaMath		AMC23		AIME24		AIME25		AIME26		Macro Average		Micro Average	
	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32	avg@32	pass@32
Qwen3-1.7B-Base																
PPO	0.6973	0.9080	**0.2986**	**0.5515**	0.4289	0.8500	0.0823	0.3333	0.0469	0.3333	0.0417	**0.2667**	0.2659	0.5405	0.5013	0.7384
DAPO	0.6935	0.9100	0.2878	0.5257	0.4133	**0.8750**	0.0854	0.3667	0.0490	0.3000	0.0385	**0.2667**	0.2612	0.5407	0.4953	0.7328
REINFORCE++	0.6877	0.9040	0.2911	0.5184	0.4016	0.8250	0.0823	0.3000	**0.0521**	**0.3667**	0.0354	**0.2667**	0.2584	0.5301	0.4924	0.7251
OPTS-TTPO	**0.7114**	**0.9220**	0.2920	0.5404	**0.4328**	0.8000	**0.1063**	**0.4333**	0.0479	0.3000	**0.0521**	**0.2667**	**0.2738**	**0.5437**	**0.5085**	**0.7428**
Qwen3-1.7B																
PPO	0.7994	0.9480	0.3427	0.5257	0.5938	0.9250	0.2000	**0.4667**	0.1708	**0.4000**	0.1083	0.3333	0.3692	0.5998	0.5887	0.7650
DAPO	0.8061	0.9280	0.3448	0.5221	0.5555	0.8750	0.1854	0.4000	0.1708	0.3000	0.1042	0.3000	0.3611	0.5542	0.5908	0.7439
REINFORCE++	0.7919	0.9400	0.3455	0.5368	0.5367	0.8500	0.1885	0.3667	0.1500	0.3667	0.0948	0.3333	0.3512	0.5656	0.5813	0.7561
OPTS-TTPO	**0.8271**	**0.9600**	**0.3532**	**0.5809**	**0.6281**	**0.9500**	**0.2354**	0.4333	**0.2031**	0.3667	**0.1333**	**0.3667**	**0.3967**	**0.6096**	**0.6119**	**0.7882**
Qwen3-8B-Base																
PPO	0.8305	0.9360	0.4169	0.5699	**0.6492**	0.8500	0.1948	0.5000	0.1615	0.3000	0.1302	0.3333	0.3972	0.5815	0.6311	0.7661
DAPO	0.8076	0.9460	0.3804	0.5368	0.5984	0.8250	0.1896	0.5000	0.1573	**0.3667**	0.1385	0.3000	0.3786	0.5791	0.6051	0.7616
REINFORCE++	0.8250	0.9360	0.4164	0.5478	0.6156	0.8750	0.1802	**0.6000**	0.1500	0.3333	0.1333	0.3333	0.3868	0.6042	0.6256	0.7650
OPTS-TTPO	**0.8514**	**0.9540**	**0.4241**	**0.5993**	0.6055	**0.9250**	**0.2031**	0.4667	**0.1813**	0.3333	**0.1458**	**0.3667**	**0.4019**	**0.6075**	**0.6443**	**0.7894**
Qwen3-8B																
PPO	0.8799	0.9700	0.4323	0.5588	0.7094	0.9250	**0.3531**	0.5667	0.2615	0.5333	0.2594	**0.5333**	0.4826	0.6812	0.6786	0.8016
DAPO	**0.8859**	**0.9740**	0.4493	**0.5735**	**0.7633**	0.9250	0.3208	0.6000	0.2656	0.4000	**0.2667**	**0.5333**	0.4919	0.6676	**0.6888**	0.8049
REINFORCE++	0.8749	0.9680	0.4323	0.5551	0.7164	0.9250	0.2938	0.5333	0.2406	0.4000	0.2073	0.5000	0.4609	0.6469	0.6718	0.7927
OPTS-TTPO	0.8825	**0.9740**	**0.4519**	0.5699	0.7586	**0.9500**	0.3365	**0.6667**	**0.2781**	**0.5667**	0.2594	**0.5333**	**0.4945**	**0.7101**	0.6882	**0.8126**

the four methods, it achieves the highest macro averages for both metrics on every model. The micro averages show the same ranking except for `avg@32` on Qwen3-8B, where DAPO scores 0.6888 and OPTS-TTPO scores 0.6882.

Takeaway 4. Across continuous control, Atari, and LLM reasoning, OPTS-TTPO improves all five MuJoCo task returns, Atari-57 human-normalized IQM, and average accuracy and answer coverage over PPO at matched budgets.

5 Conclusion

We study how tree search can improve finite-sample policy-gradient learning by increasing high-return trajectory coverage. We first make tree trajectories usable for learning: branch aggregation yields TTPG, TreeGAE, and TTPO, and the branch-aggregation experiment shows that TTPG stays near the unbranched reference as suffixes are added, whereas naive aggregation accumulates bias. We then use performance-difference-guided OPTS to allocate a finite search budget. Exact-value experiments show improvements in both expected λ-return and true return with every adjacent budget increase, while learned-critic experiments show the same aggregate trend. At matched test-time budgets, reward-guided OPTS improves correct-answer coverage and value-guided OPTS improves majority-vote accuracy over independent sampling. Coverage–bias diagnostics show that OPTS + TTPG gains more coverage at a smaller bias shift than fixed-branch + naivePG; max backup adds additional prefix credit, while mean backup averages sampled suffixes in stochastic environments. Across domains, OPTS-TTPO improves all five MuJoCo task returns, Atari-57 human-normalized IQM, and LLM accuracy and answer coverage over PPO at matched budgets. However, the theory bounds the bias from posterior-dependent expansion, rather than eliminating it, while the monotone-improvement guarantee is limited to deterministic dynamics with exact values. Future work includes variance reduction under prefix-measurable sampling and self-distillation from successful OPTS trajectories.

Reproducibility Statement

Appendices A–C define the tree interface and prove the branch-aggregation, search-improvement, prefix-credit, and gradient-bias results; Algorithm 2 gives the training step. Appendix D records all experiment protocols, budgets, evaluation sets, and baseline-matching rules, with optimizer details in Appendix D.5. The LLM MDP, budget alignment, overlap audit, and gradient-estimation protocol are specified in Appendices D.9, B.2, D.7, and D.1, respectively. Control experiments use CleanRL-style implementations and LLM experiments use VeRL; all benchmarks and simulators are public. Code and run scripts for all experiments are available at the anonymous repository `https://anonymous.4open.science/r/OPTS-TTPO/` and will be publicly released upon publication.

Ethics Statement

We evaluate a reinforcement-learning algorithm on public MuJoCo and Arcade Learning Environment simulators and public mathematical-reasoning datasets. No human subjects, personal data, or crowdsourced annotations are involved; rewards are simulator returns or automatic answer verification. The models are open-weight and training uses only the public data described in Appendix D. Potential risks are those common to reinforcement learning for language models, including inherited biases and the compute cost of large rollouts. We report budgets and use matched-budget comparisons.

AI Use Statement

Large language models were used as a writing aid in preparing this manuscript, including polishing grammar, improving clarity, and editing the presentation of text that the authors had written. They were not used to generate research ideas, derive or verify proofs, write experimental code, or produce or analyze experimental results. All technical content, theoretical results, implementations, and experiments are the work of the authors, who take full responsibility for the content of this paper.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34:29304–29320, 2021.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *Advances in neural information processing systems*, 30, 2017.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47: 253–279, 2013.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024. URL `https://arxiv.org/abs/2407.21787`.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Nicholas E. Corrado and Josiah P. Hanna. On-policy policy gradient reinforcement learning without on-policy sampling. *arXiv preprint arXiv:2311.08290*, 2023. URL `https://arxiv.org/abs/2311.08290`.

Gal Dalal, Assaf Hallak, Gugan Thoppe, Shie Mannor, and Gal Chechik. Policy gradient with tree expansion. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 12229–12255. PMLR, 2025. URL `https://proceedings.mlr.press/v267/dalal25a.html`.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. Policy improvement by planning with gumbel. In *International Conference on Learning Representations*, 2022.

Zheng Ding and Weirui Ye. TreeGRPO: Tree-advantage GRPO for online RL post-training of diffusion models. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3rZdp4TmUb.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2507.19849.

Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. First return, then explore. *Nature*, 590:580–586, 2021. doi: 10.1038/s41586-020-03157-9. URL https://www.nature.com/articles/s41586-020-03157-9.

Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. TreeQN and ATreeC: Differentiable tree-structured models for deep reinforcement learning. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1dh6Ax0Z.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023.

Tobias Graf and Marco Platzner. Adaptive playouts for online learning of policies during Monte Carlo tree search. *Theoretical Computer Science*, 644:53–62, 2016. doi: 10.1016/j.tcs.2016.06.029. URL https://doi.org/10.1016/j.tcs.2016.06.029.

Jean-Bastien Grill, Florent Altché, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and Remi Munos. Monte-Carlo tree search as regularized policy optimization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3769–3778. PMLR, 2020. URL https://proceedings.mlr.press/v119/grill20a.html.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen

Zhang. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645 (8081):633–638, September 2025a. ISSN 1476-4687. doi: 10.1038/s41586-025-09422-z. URL http://dx.doi.org/10.1038/s41586-025-09422-z.

Yiran Guo, Lijie Xu, Jie Liu, Dan Ye, and Shuang Qiu. Segment policy optimization: Effective segment-level credit assignment in RL for large language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025b. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a6536243037d1e32c20de85137d478da-Abstract-Conference.html.

Bowei He, Yankai Chen, Xiaokun Zhang, and Xue Liu. Branching policy optimization: Sandbox-native language agent reinforcement learning. *arXiv preprint arXiv:2607.14171*, 2026. URL https://arxiv.org/abs/2607.14171.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. Treerl: Llm reinforcement learning with on-policy tree search. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12355–12369, 2025.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization. *arXiv preprint arXiv:2501.03262*, 2025.

Baizhou Huang and Xiaojun Wan. Pros: Towards compute-efficient rlvr via rollout prefix reuse. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4badb55ba9c18bccb9d4146be328a948-Abstract-Conference.html.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and JoÃĢo GM AraÃšjo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pp. 4476–4486. PMLR, 2021.

Yuxiang Ji, Ziyu Ma, Yong Wang, Guanhua Chen, Xiangxiang Chu, and Liaoni Wu. Tree search for LLM agent reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ZpQwAFhU13.

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pp. 267–274, 2002. URL https://people.eecs.berkeley.edu/~pabbeel/cs287-fa09/readings/KakadeLangford-icml2002.pdf.

Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, and Nicolas Le Roux. VinePPO: Refining credit assignment in RL training of LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 29557–29590. PMLR, 2025. URL https://proceedings.mlr.press/v267/kazemnejad25a.html.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Yizhi Li, Qingshui Gu, Zhoufutu Wen, Ziniu Li, Tianshun Xing, Shuyue Guo, Tianyu Zheng, Xin Zhou, Xingwei Qu, Wangchunshu Zhou, et al. Treepo: Bridging the gap of policy optimization and efficacy and inference efficiency with heuristic tree-based modeling. *arXiv preprint arXiv:2508.17445*, 2025.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

Saad Mohamad and Giovanni Montana. Adaptive experience selection for policy gradient. *arXiv preprint arXiv:2002.06946*, 2020. URL https://arxiv.org/abs/2002.06946.

Tetsuro Morimura, Kazuhiro Ota, Kenshi Abe, and Peinan Zhang. Policy gradient algorithms with Monte Carlo tree learning for non-Markov decision processes. *Reinforcement Learning Journal*, 3:1351–1376, 2024. URL https://rlj.cs.umass.edu/2024/papers/Paper168.html.

Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX symposium on operating systems design and implementation (OSDI 18)*, pp. 561–577, 2018.

Hieu Trung Nguyen, Bao Nguyen, Wenao Ma, Yuzhi Zhao, Ruifeng She, and Viet Anh Nguyen. Adaptive rollout allocation for online reinforcement learning with verifiable rewards. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2602.01601.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. URL https://proceedings.mlr.press/v80/oh18b.html.

Lara Scavuzzo, Feng Chen, Didier Chetelat, Maxime Gasse, Andrea Lodi, Neil Yorke-Smith, and Karen Aardal. Learning to branch with tree MDPs. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/756d74cd58592849c904421e3b2ec7a4-Abstract-Conference.html.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *URL https://arxiv. org/abs/2402.03300*, 2(3):5, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, pp. 1279–1297, 2025.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018.

Dennis J. N. J. Soemers, Éric Piette, Matthew Stephenson, and Cameron Browne. Learning policies from self-play with policy gradients and MCTS value estimates. In *2019 IEEE Conference on Games (CoG)*, pp. 1–8. IEEE, 2019. doi: 10.1109/CIG.2019.8848037. URL https://doi.org/10.1109/CIG.2019.8848037.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Wanxin Tian, Shijie Zhang, Kevin Zhang, Xiaowei Chi, Chunkai Fan, Junyu Lu, Yulin Luo, Qiang Zhou, Yiming Zhao, Ning Liu, et al. Seea-r1: Tree-structured reinforcement fine-tuning for self-evolving embodied agents. *arXiv preprint arXiv:2506.21669*, 2025.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jinyang Wu, Shuo Yang, Yuhao Shen, Shuai Zhang, Zhengqi Wen, and Jianhua Tao. SPARK: Strategic policy-aware exploration via dynamic branching for long-horizon agentic learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23981–24004. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1100. URL https://aclanthology.org/2026.acl-long.1100/.

Shangyu Xing, Siyuan Wang, Chenyuan Yang, Xin-Yu Dai, and Xiang Ren. Lookahead tree-based rollouts for enhanced trajectory-level exploration in reinforcement learning with verifiable rewards. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/355d36f5f0922d050d9b6eaa3511d301-Abstract-Conference.html.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Zhicheng Yang, Zhijiang Guo, Yinya Huang, Xiaodan Liang, Yiwei Wang, and Jing Tang. Treerpo: Tree relative policy optimization. *arXiv preprint arXiv:2506.05183*, 2025b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Shangtong Zhang, Hao Chen, and Hengshuai Yao. ACE: An actor ensemble algorithm for continuous control with tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5789–5796, 2019. doi: 10.1609/aaai.v33i01.33015789. URL https://ojs.aaai.org/index.php/AAAI/article/view/4526.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (AIME) 2025, 2025. URL https://huggingface.co/datasets/math-ai/aime25.

Yijun Zhang, Fan Xu, Jiaxin Ding, Yule Xie, Shiqing Gao, Xin Ding, Haoxiang Zhang, Luoyi Fu, and Xinbing Wang. Information gain-based rollout policy optimization: An adaptive tree-structured rollout approach for multi-turn LLM agents. *arXiv preprint arXiv:2607.06223*, 2026. URL https://arxiv.org/abs/2607.06223.

Rujie Zhong, Duohan Zhang, Lukas Sch"afer, Stefano Albrecht, and Josiah Hanna. Robust on-policy sampling for data-efficient policy evaluation in reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/f2dbede0879b9d04ceb30f1b8b476b27-Abstract-Conference.html.

Zefang Zong, Dingwei Chen, Yang Li, Qi Yi, Bo Zhou, Chengming Li, Bo Qian, Peng Chen, and Jie Jiang. AT^{2}PO: Agentic turn-based policy optimization via tree search. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24120–24143. Association for Computational Linguistics, 2026. doi: 10.18653/v1/2026.acl-long.1106. URL https://aclanthology.org/2026.acl-long.1106/.

Heming Zou, Qi Wang, Yun Qu, Yuhang Jiang, Lizhou Cai, Yixiu Mao, Ru Peng, Xin Xu, Weijie Liu, Kai Yang, Saiyong Yang, and Xiangyang Ji. Trace: A unified rollout budget allocation framework for efficient agentic reinforcement learning. *arXiv preprint arXiv:2606.11119*, 2026. URL https://arxiv.org/abs/2606.11119.

Table of Contents

A Tree Trajectory Policy Optimization **17**

A.1 Standard Chain Policy Gradient and Generalized Advantage Estimation 17

A.2 On-Policy Tree Trajectories 17

A.3 Branch Aggregation Lemma 18

A.4 Applying the Branch Aggregation Lemma to Tree Trajectories 20

A.5 PPO-Style Tree Trajectory Policy Optimization 21

B On-Policy Parallel Tree Search **22**

B.1 Performance-Difference Estimation 22

B.2 Budget Alignment and Length Penalty Across Domains 23

B.3 On-Policy Parallel Tree Search 23

B.4 Max-Backup TreeGAE and Search-Budget Improvement 24

C OPTS-TTPO: Search-Enhanced Policy Gradients **28**

C.1 Posterior Max Backup as Prefix Credit 28

C.2 Controlling the Search-Induced Gradient Bias 29

C.3 OPTS-TTPO Training Procedure 32

D Experimental Setup **32**

D.1 Tree-Gradient Estimation: Setup and Protocol 32

D.2 OPTS Search and Test-Time Scaling 33

D.3 Coverage–Bias and Prefix-Credit Mechanism Protocol 34

D.4 Cross-Domain Policy Learning 34

D.5 Optimization Hyperparameters 35

D.6 Simulator State Restoration 36

D.7 Training–Evaluation Overlap Audit 36

D.8 LLM Prompt and Chat Template 36

D.9 LLM Reasoning as a Markov Decision Process 37

E Rollout Scaling Across Individual Benchmarks **37**

F Search-Budget Improvement of OPTS **38**

G Rebranching Position: Performance-Difference versus Random and Midpoint Selection **39**

H MuJoCo Hyper-Parameter Grid: Length-Penalty Exponent and Search Count **40**

I Full Atari-57 Learning Curves **41**

J Backup-Rule Comparison on Sticky-Action Atari **43**

K Per-Checkpoint LLM Evaluation Curves **45**

L Wall-Clock Composition of OPTS-TTPO **46**

A TREE TRAJECTORY POLICY OPTIMIZATION

This section derives Tree Trajectory Policy Optimization (TTPO) from the standard chain-trajectory policy gradient. We first define a general branch aggregation rule for chain statistics, and then instantiate its direct and recursive forms as the Tree Trajectory Policy Gradient (TTPG) and Tree-based Generalized Advantage Estimation (TreeGAE).

A.1 STANDARD CHAIN POLICY GRADIENT AND GENERALIZED ADVANTAGE ESTIMATION

Consider a finite-horizon discounted Markov decision process

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma, n)$$

and a differentiable stochastic policy $\pi_\theta(a \mid s)$. The time index is included in the state. A chain trajectory is written as $\tau = (x_0, \dots, x_{n-1})$, where $x_t = (s_t, a_t, r_t, s_{t+1})$, $s_0 \sim \rho_0$, $a_t \sim \pi_\theta(\cdot \mid s_t)$, and $(r_t, s_{t+1}) \sim P(\cdot \mid s_t, a_t)$. We set $V^\pi(s_n) = 0$ and pad an earlier terminal state by zero-reward absorbing transitions when necessary. The discounted objective and suffix return are

$$J(\theta) := \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{n-1}\gamma^t r_t\right], \qquad G_t := \sum_{k=t}^{n-1}\gamma^{k-t} r_k. \tag{A.1}$$

Let $Q^\pi(s_t, a_t) := \mathbb{E}[G_t \mid s_t, a_t]$, $V^\pi(s_t) := \mathbb{E}_{a\sim\pi(\cdot|s_t)}[Q^\pi(s_t, a)]$, and $A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t)$. The likelihood-ratio identity gives

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{n-1}\nabla_\theta \log \pi_\theta(a_t \mid s_t)\sum_{k=t}^{n-1}\gamma^k r_k\right] \\ &= \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{n-1}\gamma^t Q^{\pi_\theta}(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right].\end{aligned} \tag{A.2}$$

For any state-dependent baseline $b(s_t)$, $\mathbb{E}_{a_t\sim\pi_\theta(\cdot|s_t)}[b(s_t)\nabla_\theta \log \pi_\theta(a_t \mid s_t)] = 0$. Taking $b = V^{\pi_\theta}$ yields the standard chain policy gradient (Sutton et al., 1999):

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{n-1}\gamma^t A^{\pi_\theta}(s_t, a_t)\nabla_\theta \log \pi_\theta(a_t \mid s_t)\right]. \tag{A.3}$$

For any value function V, define the temporal-difference residual

$$\delta_t^V := r_t + \gamma V(s_{t+1}) - V(s_t). \tag{A.4}$$

Generalized Advantage Estimation (GAE) (Schulman et al., 2015b) is the discounted chain sum of TD residuals,

$$\widehat{A}_t^{\mathrm{GAE}(V)} := \sum_{k=t}^{n-1}(\gamma\lambda)^{k-t}\delta_k^V = \delta_t^V + \gamma\lambda\widehat{A}_{t+1}^{\mathrm{GAE}(V)}, \qquad \widehat{A}_n^{\mathrm{GAE}(V)} = 0. \tag{A.5}$$

When $V = V^{\pi_\theta}$, the first TD residual satisfies $\mathbb{E}[\delta_t^{V^{\pi_\theta}} \mid s_t, a_t] = A^{\pi_\theta}(s_t, a_t)$. For every $k > t$, on-policy action sampling and the Markov property give

$$\mathbb{E}\left[\delta_k^{V^{\pi_\theta}} \mid s_t, a_t\right] = \mathbb{E}\left[\mathbb{E}\left[\delta_k^{V^{\pi_\theta}} \mid s_k\right]\middle| s_t, a_t\right] = 0. \tag{A.6}$$

Consequently,

$$\mathbb{E}\left[\widehat{A}_t^{\mathrm{GAE}(V^{\pi_\theta})} \mid s_t, a_t\right] = A^{\pi_\theta}(s_t, a_t). \tag{A.7}$$

A.2 ON-POLICY TREE TRAJECTORIES

An on-policy tree trajectory $\mathcal{T} = (s_o, \mathcal{X}(\mathcal{T}))$ consists of a root state $s_o \sim \rho_0$ and a finite indexed set $\mathcal{X}(\mathcal{T})$ of transition occurrences. The index o identifies the root-state occurrence. Starting from s_o, the construction recursively samples outgoing transitions and extends their successor occurrences

to the horizon; suffixes sharing a state occurrence share the prefix ending there. A transition from a parent state occurrence s_p to a child state occurrence s_c is written as (s_p, a_c, r_c, s_c). The transition set satisfies the following conditions. First, rooted connectivity requires every transition to have a unique path from s_o: its source is either s_o or the successor-state occurrence of exactly one preceding transition. Second, every transition is generated by the current policy and the environment,

$$a_c \sim \pi_\theta(\cdot \mid s_p), \qquad (r_c, s_c) \sim P(\cdot \mid s_p, a_c). \tag{A.8}$$

When the parent–child indices are not needed, we index transition occurrences by x and denote their source state, action, reward, and successor state by s_x, a_x, r_x, s_{x^+}, respectively. The successor state s_{x^+} is also the source state of each child transition x^+. The depth of x is $d(x) \in \{0, \dots, n-1\}$, and $x \in \mathcal{T}$ means $x \in \mathcal{X}(\mathcal{T})$.

Here, *on-policy* refers to the conditional generation in Equation A.8. Because every new suffix is sampled from the current policy, tree search introduces no separate behavior policy and requires no additional action-distribution importance correction at data collection. This local property does not imply that the random tree has the same state-visitation distribution as a policy chain. Selecting a previously visited state for rebranching is a decision made by an external algorithm rather than an action generated by the policy; states preferred by the branching rule therefore receive more suffix samples in the tree. The Branch Aggregation Lemma below gives the conditions under which a branch-weighted additive statistic on such a tree recovers the expectation of its chain-trajectory counterpart.

A.3 Branch Aggregation Lemma

The policy gradient and GAE are expectations of discounted sums along chain trajectories, and a tree changes how often each part of a chain is observed. Section 3.1 illustrates this with three suffixes sampled at one depth of a chain: naive summation counts the suffix three times relative to the prefix, and averaging the three suffixes restores the chain proportions. The lemma generalizes this correction to arbitrary trees: it gives the weights under which tree sums recover the chain expectation, and states the conditions on how branches may be chosen for the correction to hold.

Let h be any integrable scalar- or vector-valued function of a transition and let η be a scalar. The chain statistic of interest is

$$H_\eta(\tau) := \sum_{t=0}^{n-1} \eta^t h(s_t, a_t, r_t, s_{t+1}). \tag{A.9}$$

Its on-policy target is $\mathbb{E}_{\tau \sim \pi_\theta}[H_\eta(\tau)]$.

Let $\mathcal{F}_p$ denote the sigma-field of all information available upon reaching s_p, before sampling its outgoing transitions. Write $\mathcal{C}(p)$ for its child-index set and $\alpha_{p,c}$ for a local branch weight. In the generic transition notation x, the corresponding information before sampling a_x is denoted by $\mathcal{F}_x$.

Conditions for recovering the chain expectation.

1. **Conditional on-policy sampling.** Given $\mathcal{F}_p$, each outgoing transition $c \in \mathcal{C}(p)$ is sampled according to Equation A.8.
2. **Measurable, normalized branch weights.** The finite local-weight family $(\alpha_{p,c})_{c \in \mathcal{C}(p)}$ is $\mathcal{F}_p$-measurable and satisfies

$$\alpha_{p,c} \geq 0, \qquad \sum_{c \in \mathcal{C}(p)} \alpha_{p,c} = 1. \tag{A.10}$$

The child set and its local weights may be random and prefix-dependent, but they must be $\mathcal{F}_p$-measurable, that is, determined before the corresponding outgoing transitions are sampled. Conditional on-policy sampling alone is therefore insufficient for Equation A.16.

Additional on-policy suffixes may be sampled from selected state occurrences and share their existing prefixes. Further branching may occur along the resulting suffixes, while unbranched segments retain the ordinary chain structure. The local contribution of a transition is still $h(c) := h(s_p, a_c, r_c, s_c)$, and a shared prefix appears once in the tree. Along a branch, the available information satisfies $\mathcal{F}_p \subseteq \mathcal{F}_c$.

The global branch weight is computed recursively as

$$W(o) := 1, \qquad W(c) := W(p)\alpha_{p,c}, \quad c \in \mathcal{C}(p). \tag{A.11}$$

Thus $W(c)$ is the product of the local weights along the unique path from s_o to c. We next define the branch-weighted tree statistic by direct summation over all transitions and by recursive aggregation from the leaves to the root.

Direct aggregation. The direct branch-weighted tree sum is

$$\widehat{H}_{\eta,\mathrm{dir}}(\mathcal{T}) := \sum_{c \in \mathcal{X}(\mathcal{T})} W(c)\eta^{d(c)}h(c). \tag{A.12}$$

Recursive aggregation. Let $\widehat{R}_\eta(s_p)$ denote the aggregated suffix beginning at the state occurrence s_p. Starting from the leaves, define

$$\widehat{R}_\eta(s_p) := \sum_{c \in \mathcal{C}(p)} \alpha_{p,c}\left[h(c) + \eta\widehat{R}_\eta(s_c)\right], \tag{A.13}$$

where $\widehat{R}_\eta(s_c) = 0$ when $d(c) = n-1$. At the root, define

$$\widehat{H}_{\eta,\mathrm{rec}}(\mathcal{T}) := \widehat{R}_\eta(s_o). \tag{A.14}$$

Lemma 1 (Branch aggregation). For every realized tree, the two aggregation forms satisfy

$$\widehat{H}_{\eta,\mathrm{dir}}(\mathcal{T}) = \widehat{H}_{\eta,\mathrm{rec}}(\mathcal{T}). \tag{A.15}$$

Under the recovery conditions above, both forms recover the expectation of the original chain statistic:

$$\mathbb{E}_{\mathcal{T}}\left[\widehat{H}_{\eta,\mathrm{dir}}(\mathcal{T})\right] = \mathbb{E}_{\mathcal{T}}\left[\widehat{H}_{\eta,\mathrm{rec}}(\mathcal{T})\right] = \mathbb{E}_{\tau\sim\pi_\theta}\left[H_\eta(\tau)\right]. \tag{A.16}$$

Proof. Part I: Direct–recursive equivalence.

Root layer. Equations A.11 and A.13 give

$$\begin{aligned}\widehat{R}_\eta(s_o) &= \sum_{c \in \mathcal{C}(o)} \alpha_{o,c}\left[h(c) + \eta\widehat{R}_\eta(s_c)\right] \\ &= \sum_{d(c)=0} W(c)h(c) + \eta \sum_{d(c)=0} W(c)\widehat{R}_\eta(s_c).\end{aligned} \tag{A.17}$$

One-layer expansion. For every $k \in \{0, \ldots, n-2\}$, expanding the depth-k remainder once gives

$$\begin{aligned}&\eta^{k+1} \sum_{d(p)=k} W(p)\widehat{R}_\eta(s_p) \\ &= \eta^{k+1} \sum_{d(p)=k} \sum_{c \in \mathcal{C}(p)} W(p)\alpha_{p,c}\left[h(c) + \eta\widehat{R}_\eta(s_c)\right] \\ &= \eta^{k+1} \sum_{d(c)=k+1} W(c)h(c) + \eta^{k+2} \sum_{d(c)=k+1} W(c)\widehat{R}_\eta(s_c).\end{aligned} \tag{A.18}$$

Full expansion. Applying Equation A.18 successively to Equation A.17, and using $\widehat{R}_\eta(s_c) = 0$ at $d(c) = n-1$, yields

$$\begin{aligned}\widehat{H}_{\eta,\mathrm{rec}}(\mathcal{T}) = \widehat{R}_\eta(s_o) &= \sum_{k=0}^{n-1} \sum_{d(c)=k} W(c)\eta^k h(c) \\ &= \sum_{c \in \mathcal{X}(\mathcal{T})} W(c)\eta^{d(c)}h(c) = \widehat{H}_{\eta,\mathrm{dir}}(\mathcal{T}).\end{aligned} \tag{A.19}$$

Part II: Recovery of the chain expectation.

Chain suffix target. Let τ_p be an on-policy chain suffix from s_p through the remaining horizon, and let $H_\eta(\tau_p)$ denote its remaining chain sum with the local discount starting from η^0; at the terminal state this sum is zero. Define

$$\mu_\eta(p) := \mathbb{E}_{\tau_p \sim \pi_\theta} \left[H_\eta(\tau_p) \mid \mathcal{F}_p \right]. \tag{A.20}$$

One-step chain recursion. Equation A.8 and the tower property give, for every $c \in \mathcal{C}(p)$,

$$\mathbb{E}_{\mathcal{T}} \left[h(c) + \eta \mu_\eta(c) \mid \mathcal{F}_p \right] = \mu_\eta(p). \tag{A.21}$$

Base case. At $d(c) = n - 1$,

$$\mathbb{E}_{\mathcal{T}} \left[\widehat{R}_\eta(s_c) \mid \mathcal{F}_c \right] = 0 = \mu_\eta(c). \tag{A.22}$$

Induction step. Assuming the same identity at the children of p, Equations A.13 and A.21 give

$$\begin{aligned} \mathbb{E}_{\mathcal{T}} \left[\widehat{R}_\eta(s_p) \mid \mathcal{F}_p \right] &= \sum_{c \in \mathcal{C}(p)} \alpha_{p,c} \mathbb{E}_{\mathcal{T}} \left[h(c) + \eta \mathbb{E}_{\mathcal{T}} \left[\widehat{R}_\eta(s_c) \mid \mathcal{F}_c \right] \middle| \mathcal{F}_p \right] \\ &= \sum_{c \in \mathcal{C}(p)} \alpha_{p,c} \mathbb{E}_{\mathcal{T}} \left[h(c) + \eta \mu_\eta(c) \mid \mathcal{F}_p \right] \\ &= \sum_{c \in \mathcal{C}(p)} \alpha_{p,c} \mu_\eta(p) = \mu_\eta(p). \end{aligned} \tag{A.23}$$

Root expectation. Finally, $s_o \sim \rho_0$ and the tower property give

$$\begin{aligned} \mathbb{E}_{\mathcal{T}} \left[\widehat{H}_{\eta,\mathrm{rec}}(\mathcal{T}) \right] &= \mathbb{E}_{\mathcal{T}} \left[\mathbb{E}_{\mathcal{T}} \left[\widehat{R}_\eta(s_o) \mid \mathcal{F}_o \right] \right] \\ &= \mathbb{E}_{s_o \sim \rho_0} \left[\mu_\eta(o) \right] = \mathbb{E}_{\tau \sim \pi_\theta} \left[H_\eta(\tau) \right]. \end{aligned} \tag{A.24}$$

Together with Equation A.19, this proves Equation A.16. □

A.4 Applying the Branch Aggregation Lemma to Tree Trajectories

The tree generator in this subsection satisfies the chain-expectation recovery conditions in Appendix A.3; in particular, its transitions are conditionally sampled from π_θ and its branch weights are fixed from prefix information before the corresponding outgoing transitions are sampled.

Tree Trajectory Policy Gradient (TTPG). The branching decisions and weights are held fixed under differentiation. Set $\eta = \gamma$ and

$$h(x) = A^{\pi_\theta}(s_x, a_x) \nabla_\theta \log \pi_\theta(a_x \mid s_x)$$

in the direct aggregation formula. Applying Lemma 1 and Equation A.3 gives

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathcal{T}} \left[\sum_{x \in \mathcal{T}} W(x) \gamma^{d(x)} A^{\pi_\theta}(s_x, a_x) \nabla_\theta \log \pi_\theta(a_x \mid s_x) \right]. \tag{A.25}$$

Thus TTPG is the direct-summation instance of branch aggregation.

Tree-based Generalized Advantage Estimation (TreeGAE). For a transition x, define

$$\delta_x^V := r_x + \gamma V(s_{x^+}) - V(s_x). \tag{A.26}$$

Set $\eta = \gamma\lambda$ and $h(x) = \delta_x^V$ in recursive aggregation. The resulting TreeGAE recursion is

$$\widehat{A}_x^{\mathrm{TreeGAE}(V)} := \delta_x^V + \gamma\lambda \sum_{x^+ \in \mathcal{C}(x)} \alpha_{x,x^+} \widehat{A}_{x^+}^{\mathrm{TreeGAE}(V)}, \tag{A.27}$$

with the recursive term set to zero beyond depth $n - 1$.

Applying Lemma 1 to the transition subtree beginning at x shows that TreeGAE and chain GAE have the same conditional suffix expectation under the same value function. For $V = V^{\pi_\theta}$, Equation A.7 and the Markov property give

$$\mathbb{E}\left[\widehat{A}_x^{\mathrm{TreeGAE}(V^{\pi_\theta})} \mid \mathcal{F}_x, a_x\right] = \mathbb{E}\left[\widehat{A}_t^{\mathrm{GAE}(V^{\pi_\theta})} \mid s_t, a_t\right] = A^{\pi_\theta}(s_x, a_x). \tag{A.28}$$

By Equation A.28 and the $\mathcal{F}_x$-measurability of $W(x)$,

$$\begin{aligned}&\mathbb{E}\left[W(x)\gamma^{d(x)}\widehat{A}_x^{\mathrm{TreeGAE}(V^{\pi_\theta})}\nabla_\theta \log \pi_\theta(a_x \mid s_x)\middle|\mathcal{F}_x, a_x\right]\\ &= W(x)\gamma^{d(x)}A^{\pi_\theta}(s_x, a_x)\nabla_\theta \log \pi_\theta(a_x \mid s_x).\end{aligned} \tag{A.29}$$

Summing over the tree and applying the tower property yields

$$\mathbb{E}_{\mathcal{T}}\left[\sum_{x\in\mathcal{T}} W(x)\gamma^{d(x)}\widehat{A}_x^{\mathrm{TreeGAE}(V^{\pi_\theta})}\nabla_\theta \log \pi_\theta(a_x \mid s_x)\right] = \nabla_\theta J(\theta). \tag{A.30}$$

A.5 PPO-Style Tree Trajectory Policy Optimization

In practice, as in PPO, we omit the outer state-visitation discount by setting $\gamma^{d(x)} = 1$ in the actor and critic objectives and the normalization statistics below. The discount used in return estimation and TreeGAE remains unchanged.

Let $\pi_{\theta_{\mathrm{old}}}$ be the policy used to sample the actions in the on-policy tree and define the per-node probability ratio

$$\rho_x(\theta) := \frac{\pi_\theta(a_x \mid s_x)}{\pi_{\theta_{\mathrm{old}}}(a_x \mid s_x)}. \tag{A.31}$$

For a minibatch $\mathcal{B}$ of sampled trees, let $N_{\mathcal{B}} := \sum_{\mathcal{T}\in\mathcal{B}} |\mathcal{X}(\mathcal{T})|$ be the number of transitions it contains, and define the branch-weighted empirical average by

$$\mathbb{E}_x[W(x)f(x)] := \frac{1}{N_{\mathcal{B}}}\sum_{\mathcal{T}\in\mathcal{B}}\sum_{x\in\mathcal{T}} W(x)f(x). \tag{A.32}$$

Search changes how rollout suffixes share prefixes and therefore how many transition occurrences are stored in a minibatch. Following the per-transition averaging used in PPO implementations, we divide the branch-weighted actor and critic sums by $N_{\mathcal{B}}$, so that the raw number of stored transitions does not directly scale the update. This is a practical loss normalization rather than part of the chain-expectation identity in Lemma 1. With the tree and its branch weights held fixed, the clipped TTPO actor objective extends the PPO surrogate (Schulman et al., 2017) to action-level gradient aggregation:

$$\mathcal{L}_\pi^{\mathrm{TTPO}}(\theta) := \mathbb{E}_x\left[W(x)\min\left(\rho_x(\theta)\widehat{A}_x, \mathrm{clip}(\rho_x(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_x\right)\right], \tag{A.33}$$

where $\widehat{A}_x$ is fixed when optimizing θ. At $\theta = \theta_{\mathrm{old}}$, $\rho_x(\theta_{\mathrm{old}}) = 1$, and the derivative of either clipped branch is $\widehat{A}_x\nabla_\theta \log \pi_\theta(a_x \mid s_x)$. Hence

$$\nabla_\theta \mathcal{L}_\pi^{\mathrm{TTPO}}(\theta)\big|_{\theta=\theta_{\mathrm{old}}} = \mathbb{E}_x\left[W(x)\widehat{A}_x\nabla_{\theta_{\mathrm{old}}} \log \pi_{\theta_{\mathrm{old}}}(a_x \mid s_x)\right]. \tag{A.34}$$

Thus maximizing $\mathcal{L}_\pi^{\mathrm{TTPO}}$ gives the PPO-style clipped counterpart of TTPG under the practical convention above.

For value estimation, let $\bar{\phi}$ denote the critic parameters used to construct the fixed target, define $\widehat{R}_x := \widehat{A}_x + V_{\bar{\phi}}(s_x)$, and define the clipped value prediction

$$V_\phi^{\mathrm{clip}}(s_x) := V_{\bar{\phi}}(s_x) + \mathrm{clip}\left(V_\phi(s_x) - V_{\bar{\phi}}(s_x), -\epsilon_V, \epsilon_V\right). \tag{A.35}$$

The corresponding branch-weighted value objective is

$$\mathcal{L}_V^{\mathrm{TTPO}}(\phi) := \mathbb{E}_x\left[W(x)\max\left((V_\phi(s_x) - \widehat{R}_x)^2, (V_\phi^{\mathrm{clip}}(s_x) - \widehat{R}_x)^2\right)\right]. \tag{A.36}$$

The actor objective is maximized and the value objective is minimized.

Branch-weighted advantage normalization. Whitening serves a different purpose and therefore uses the total branch mass $Z_{\mathcal{B}} := \sum_{\mathcal{T}\in\mathcal{B}} \sum_{x\in\mathcal{T}} W(x)$ to normalize the branch weights. Under the conditions of Lemma 1, the resulting weighted statistics are consistent for the corresponding moments of the chain-transition distribution; using $N_{\mathcal{B}}$ instead would leave the branch weights unnormalized and allow branch multiplicity to distort these moments. As in PPO, whitening with finite-batch statistics remains an approximation. Define the weighted advantage mean and degrees-of-freedom-corrected variance by

$$\begin{aligned} \mu_A &:= \frac{1}{Z_{\mathcal{B}}} \sum_{\mathcal{T}\in\mathcal{B}} \sum_{x\in\mathcal{T}} W(x)\widehat{A}_x, \\ \sigma_A^2 &:= \frac{\sum_{\mathcal{T}\in\mathcal{B}} \sum_{x\in\mathcal{T}} W(x)(\widehat{A}_x - \mu_A)^2}{Z_{\mathcal{B}} - Z_{\mathcal{B}}^{-1} \sum_{\mathcal{T}\in\mathcal{B}} \sum_{x\in\mathcal{T}} W(x)^2}. \end{aligned} \tag{A.37}$$

The normalized advantage is

$$\widetilde{A}_x := \frac{\widehat{A}_x - \mu_A}{\sqrt{\sigma_A^2 + \varepsilon}}. \tag{A.38}$$

When advantage normalization is enabled, $\widetilde{A}_x$ replaces $\widehat{A}_x$ in Equation A.33.

Scope and transition to adaptive search. Unlike the tree generators covered above, OPTS, introduced in the next section, selects expansion states using outcomes already observed in the tree. Its resulting child sets and local weights therefore need not be $\mathcal{F}_p$-measurable and can be correlated with the suffix statistics they aggregate. Consequently, the conditional-expectation step in Equation A.23 does not directly extend to OPTS trees, and Lemma 1 alone does not establish chain-expectation recovery for the adaptive search distribution. Appendix C separates this posterior selection effect from the additional max-backup prefix-credit term and bounds the resulting gradient bias.

B On-Policy Parallel Tree Search

This section derives a performance-difference estimate for rebranching by accumulating policy-relative local action improvements, aligns the estimate with the domain's search-budget unit, and uses it to define On-Policy Parallel Tree Search (OPTS). This is the budget-allocation layer of the method.

B.1 Performance-Difference Estimation

Let π be the policy used to resample a suffix, and let μ be an arbitrary behavior policy. Starting from s_t, μ and the environment generate $\tau = (s_t, a_t, r_t, \ldots, s_{n-1}, a_{n-1}, r_{n-1}, s_n) \sim (\mu, P)$, where s_n is terminal and $V^\pi(s_n) = 0$.

Performance-Difference Estimate. We define the performance-difference estimate at s_t on τ as

$$\Delta(s_t;\tau) := -\sum_{k=t}^{n-1} \gamma^{k-t} A^\pi(s_k, a_k). \tag{B.1}$$

Each term $-A^\pi(s_k, a_k) = V^\pi(s_k) - Q^\pi(s_k, a_k)$ is the local expected improvement from replacing a_k by a fresh action from π and following π thereafter. The estimate has two properties:

1. **Expected performance improvement.** The performance difference lemma (Kakade & Langford, 2002; Schulman et al., 2015a) gives

$$\mathbb{E}_{\tau\sim(\mu,P)|s_t}[\Delta(s_t;\tau)] = V^\pi(s_t) - V^\mu(s_t). \tag{B.2}$$

Thus $\Delta(s_t;\tau)$ estimates the expected return improvement from switching from μ to π at s_t. In particular, when $\mu = \pi$, Equation B.2 becomes

$$\mathbb{E}_{\tau\sim(\pi,P)|s_t}[\Delta(s_t;\tau)] = 0. \tag{B.3}$$

Before observing the reference suffix, retaining it and independently resampling from the same policy have equal expected value.

2. **Improvement in deterministic environments.** When the environment is deterministic, $Q^\pi(s_k, a_k) = r_k + \gamma V^\pi(s_{k+1})$, and Equation B.1 telescopes to

$$\Delta(s_t; \tau) = V^\pi(s_t) - \sum_{k=t}^{n-1} \gamma^{k-t} r_k. \tag{B.4}$$

The first term is the expected return of a new suffix sampled from π, and the second is the return of the observed reference suffix. Their difference is the expected improvement from rebranching at s_t.

Rebranching state selection. The rebranching state is the admissible state on the observed path τ with the largest performance-difference estimate:

$$s^\star \in \arg\max_{s_t \in \tau} \Delta(s_t; \tau). \tag{B.5}$$

TreeGAE estimation. In practice, TreeGAE advantages $\widehat{A}_{x_k}$ replace A^π, giving the TreeGAE-estimated performance difference:

$$\widehat{\Delta}(s_t; \tau) := -\sum_{k=t}^{n-1} \gamma^{k-t} \widehat{A}_{x_k}, \qquad s^\star \in \arg\max_{s_t \in \tau} \widehat{\Delta}(s_t; \tau). \tag{B.6}$$

B.2 Budget Alignment and Length Penalty Across Domains

The search budget is measured in rollouts for LLMs and in transitions for Atari and MuJoCo. Accordingly, we use the unpenalized estimate for LLMs and a suffix-length-penalized estimate for Atari and MuJoCo:

$$\widehat{\Delta}^{(\xi)}(s_k; \tau) = \frac{\widehat{\Delta}(s_k; \tau)}{(n-k)^\xi}, \qquad \xi \in [0, 1]. \tag{B.7}$$

At $\xi = 0$, this recovers the original performance-difference estimate, while $\xi = 1$ gives the average discounted local improvement along the suffix. Values $0 < \xi < 1$ balance total improvement against suffix length. Specific domain settings are given in Appendix D.

B.3 On-Policy Parallel Tree Search

A greedy path starts at the root and repeatedly follows $c^\star(p) \in \arg\max_{c \in \mathcal{C}(p)} \widehat{A}_c$ until reaching a leaf. The OPTS portion of Figure 1 illustrates the search process, and Algorithm 1 gives the batch-level OPTS procedure. It backs up Equation B.6 along each greedy reference path and applies Equation B.7 before selecting the rebranching position.

Selected suffixes in Algorithm 1 are sampled from π at the selected states, while fresh root trajectories that fill unused batch slots are sampled from ρ_0; both are generated in one batched rollout. Thus adaptive search requires no separate action-distribution importance correction. The selected state, however, is a function of the observed tree through the backed-up advantages and performance-difference estimates. Consequently, the induced tree distribution generally violates the prefix-measurability condition of Lemma 1. Uniform branch weights still account for branch multiplicity, but they do not by themselves remove this posterior selection effect; Appendix C.2 bounds the resulting gradient bias.

Algorithm 1 On-Policy Parallel Tree Search (OPTS)

Require: Policy π, environment P, initial distribution ρ_0, value function $\widehat{V}$, batch size B, batch runs R, per-tree search budget $S_{\max}$, budget exponent ξ, baseline

$s_{i,0} \overset{\text{i.i.d.}}{\sim} \rho_0, \quad i = 1, \dots, B$
$\{\tau_i\} \leftarrow \textsc{BatchRollout}(\{s_{i,0}\}, \pi, P)$
$\mathcal{T}_i \leftarrow \textsc{InitTree}(\tau_i), \quad j_i \leftarrow 0, \quad i = 1, \dots, B$
$\textsc{TreeGAEBackup}(\{\tau_i\} \to \text{roots, rewards, } \widehat{V})$
for $\ell = 1, \dots, R-1$ **do**
  for all active trees i with $j_i < S_{\max}$ **in parallel do**
    $\tau_i^\star \leftarrow \textsc{GreedyPath}(\mathcal{T}_i)$
    $\widehat{\Delta}(\cdot; \tau_i^\star) \leftarrow \textsc{DeltaBackup}(\tau_i^\star, \widehat{A}, \gamma)$
    $s_i^\star \in \arg\max_{s_{i,t} \in \tau_i^\star} \widehat{\Delta}^{(\xi)}(s_{i,t}; \tau_i^\star)$
  end for
  $\{s_i^\star\} \leftarrow \mathrm{Take}_B\left(\left\{ s_i^\star \mid \widehat{\Delta}^{(\xi)}(s_i^\star; \tau_i^\star) > \text{baseline} \right\}\right)$
  $s_{k,0} \overset{\text{i.i.d.}}{\sim} \rho_0, \quad |\{s_{k,0}\}| = B - |\{s_i^\star\}|$
  $(\{\tau_i\}, \{\tau_k\}) \leftarrow \textsc{BatchRollout}(\{s_i^\star\} \cup \{s_{k,0}\}, \pi, P)$
  $\mathcal{T}_i \leftarrow \textsc{Attach}(\mathcal{T}_i, s_i^\star, \tau_i), \quad j_i \leftarrow j_i + 1$
  $\mathcal{T}_k \leftarrow \textsc{InitTree}(\tau_k), \quad j_k \leftarrow 0$
  $\textsc{TreeGAEBackup}(\{\tau_i\} \cup \{\tau_k\} \to \text{roots, rewards, } \widehat{V})$
end for
return $\{\mathcal{T}_i\}$

Here k indexes new trees, and Take_B retains at most B states.

Tree-conditioned greedy path. For a fixed root s_o, couple the search trees from one OPTS run as $\mathcal{T}_j \sim q_j^{\mathrm{OPTS}}(\cdot \mid s_o; \pi, P)$, where $j \in \{0, \dots, S_{\max}\}$ is the number of completed rebranching expansions. Let $\mathcal{P}_j$ be the retained root-to-leaf paths of $\mathcal{T}_j$. OPTS only attaches new suffixes and retains all existing paths; after termination, let subsequent trees remain unchanged. Hence, for $0 \le j_1 \le j_2 \le S_{\max}$,

$$\mathcal{T}_{j_1} \subseteq \mathcal{T}_{j_2}, \qquad \mathcal{P}_{j_1} \subseteq \mathcal{P}_{j_2}. \tag{B.8}$$

At $j = 0$, $\mathcal{T}_0$ contains the initial path $\tau^{(0)} \sim p_{\pi,P}(\cdot \mid s_o)$ and $\mathcal{P}_0 := \{\tau^{(0)}\}$. Once the tree is realized, its backed-up scores determine the greedy path

$$\tau_j^\star := \textsc{GreedyPath}(\mathcal{T}_j) \in \mathcal{P}_j. \tag{B.9}$$

Greedy path policy. For a realized tree $\mathcal{T}_j$, the greedy path policy $\pi_j^\star$ executes the actions recorded on $\tau_j^\star$ while the executed history matches a prefix of that path, and follows π after the first deviation.

Induced search policy. Across OPTS runs, $\mathcal{T}_j$ is random under q_j^{OPTS}. Let τ' denote the trajectory generated by executing the greedy path policy from s_o. Marginalizing this execution over the latent search-tree randomness defines the induced, generally history-dependent search policy π_j^S:

$$p(\tau' \mid \pi_j^S, P, s_o) := \mathbb{E}_{\mathcal{T}_j \sim q_j^{\mathrm{OPTS}}} \left[p(\tau' \mid \pi_j^\star, P, s_o) \right]. \tag{B.10}$$

Thus $\pi_j^\star$ is conditioned on one realized tree, and π_j^S is its marginal execution behavior across OPTS trees.

B.4 Max-Backup TreeGAE and Search-Budget Improvement

In deterministic environments, OPTS uses max-backup TreeGAE throughout search. For $\mathcal{C}(x) \neq \varnothing$, specialize Equation A.27 with

$$\alpha_{x,x^+} = \begin{cases} 1, & x^+ = c^\star(x), \\ 0, & x^+ \neq c^\star(x), \end{cases} \qquad c^\star(x) \in \arg\max_{c \in \mathcal{C}(x)} \widehat{A}_c^{\mathrm{TreeGAE}(V)}.$$

Let $V = V^\pi$ and hold it fixed, giving

$$\widehat{A}_{\max}(x) := \delta_x^V + \gamma\lambda \max_{c\in\mathcal{C}(x)} \widehat{A}_{\max}(c), \qquad \max \varnothing := 0. \tag{B.11}$$

The greedy path follows a maximizing child at each visited state.

For a path τ, define

$$G_\lambda(\tau) := V^\pi(s_o) + \sum_{t=0}^{n-1} (\gamma\lambda)^t \delta_t^V. \tag{B.12}$$

For any policy μ, let

$$J_\lambda(\mu) := \mathbb{E}_{\tau\sim\mu}\left[G_\lambda(\tau)\right]. \tag{B.13}$$

At zero search, the only retained path is the original rollout, so $\pi_0^S = \pi$.

Theorem 1 (Monotone Improvement of the Search Objective). In the deterministic environment above with the fixed exact value function $V = V^\pi$, consider any search with nested path sets (Equation B.8) and max-backup greedy selection, and let π_j^S be its induced policy as in Equation B.10. If G_λ is integrable, then for every $\lambda \in [0, 1]$ and $0 \le j_1 \le j_2$,

$$J_\lambda(\pi_{j_2}^S) \ge J_\lambda(\pi_{j_1}^S) \ge J_\lambda(\pi) = J(\pi). \tag{B.14}$$

The first inequality is strict if $\Pr\big(G_\lambda(\tau_{j_2}^\star) > G_\lambda(\tau_{j_1}^\star)\big) > 0$.

Proof. Part I: Path optimality.

Fix a search tree $\mathcal{T}_j$ and consider its retained suffix paths. We identify the max-backup values with their maximal discounted TD sums by backward induction.

Base case. At a leaf transition x, with $t = d(x) = n - 1$,

$$\max_{\tau:\text{ suffix from } x} \sum_{k=t}^{n-1} (\gamma\lambda)^{k-t}\delta_k^V = \delta_x^V = \widehat{A}_{\max}(x).$$

Induction step. Let $t = d(x) < n - 1$ and assume the identity holds at every child $c \in \mathcal{C}(x)$. Partitioning the suffixes by their first child and using $\gamma\lambda \ge 0$ gives

$$\begin{aligned}
&\max_{\tau:\text{ suffix from } x} \sum_{k=t}^{n-1} (\gamma\lambda)^{k-t}\delta_k^V \\
&= \delta_x^V + \gamma\lambda \max_{c\in\mathcal{C}(x)} \left[\max_{\tau:\text{ suffix from } c} \sum_{k=t+1}^{n-1} (\gamma\lambda)^{k-t-1}\delta_k^V \right] \\
&= \delta_x^V + \gamma\lambda \max_{c\in\mathcal{C}(x)} \widehat{A}_{\max}(c) = \widehat{A}_{\max}(x).
\end{aligned}$$

Root conclusion. The greedy path attains each recursive maximum. Induction from the leaves to the root therefore yields

$$\begin{aligned}
G_\lambda(\tau_j^\star) &= V^\pi(s_o) + \max_{c\in\mathcal{C}(o)} \widehat{A}_{\max}(c) \\
&= V^\pi(s_o) + \max_{\tau\in\mathcal{P}_j} \sum_{t=0}^{n-1} (\gamma\lambda)^t \delta_t^V \\
&= \max_{\tau\in\mathcal{P}_j} G_\lambda(\tau).
\end{aligned} \tag{B.15}$$

Thus $\tau_j^\star \in \arg\max_{\tau\in\mathcal{P}_j} G_\lambda(\tau)$.

Part II: Budget monotonicity.

Equations B.8 and B.15 give

$$\begin{aligned} G_\lambda(\tau^\star_{j_2}) &= \max_{\tau\in\mathcal{P}_{j_2}} G_\lambda(\tau) \\ &\geq \max_{\tau\in\mathcal{P}_{j_1}} G_\lambda(\tau) = G_\lambda(\tau^\star_{j_1}) \\ &\geq G_\lambda(\tau^{(0)}). \end{aligned} \tag{B.16}$$

Taking expectations over the initial state and search tree, Equation B.10 yields

$$\begin{aligned} J_\lambda(\pi^S_{j_2}) &= \mathbb{E}[G_\lambda(\tau^\star_{j_2})] \\ &\geq \mathbb{E}[G_\lambda(\tau^\star_{j_1})] = J_\lambda(\pi^S_{j_1}) \\ &\geq \mathbb{E}[G_\lambda(\tau^{(0)})] = J_\lambda(\pi). \end{aligned}$$

With $V = V^\pi$, the Bellman equation gives

$$\begin{aligned} J_\lambda(\pi) &= \mathbb{E}_{s_o\sim\rho_0}[V^\pi(s_o)] + \sum_{t=0}^{n-1}(\gamma\lambda)^t \underbrace{\mathbb{E}_{\tau\sim\pi}\big[\mathbb{E}[\delta^V_t \mid s_t]\big]}_{=0} \\ &= \mathbb{E}_{s_o\sim\rho_0}[V^\pi(s_o)] = J(\pi). \end{aligned}$$

For strictness,

$$\Pr\big(G_\lambda(\tau^\star_{j_2}) > G_\lambda(\tau^\star_{j_1})\big) > 0 \quad\Longrightarrow\quad J_\lambda(\pi^S_{j_2}) - J_\lambda(\pi^S_{j_1}) = \mathbb{E}[G_\lambda(\tau^\star_{j_2}) - G_\lambda(\tau^\star_{j_1})] > 0.$$

□

Theorem 2 (Monotone Improvement under OPTS). In a deterministic environment with the fixed exact value function $V = V^\pi$, assume integrable returns, $\mathrm{baseline} \geq 0$, independent complete on-policy suffix rollouts, and suffix-closed admissibility (s_t admissible $\Rightarrow$ s_{t+1} admissible for $t < n-1$). Retain the incumbent on max-backup ties. Then, for $0 < \gamma \leq 1$, $\lambda \in [0,1]$, and $0 \leq j_1 \leq j_2 \leq S_{\max}$,

$$J(\pi^S_{j_2}) \geq J(\pi^S_{j_1}) \geq J(\pi). \tag{B.17}$$

Proof. Part I: Incumbent return bound.

Fix $\mathcal{T}_j$, and let x be the transition leaving the selected state s_t on $\tau^\star_j$, with $t = d(x)$. Write $\widehat{\Delta}(s_k)$ for the estimates on this path in Equation B.6, with $\widehat{\Delta}(s_n) = 0$. Along the greedy path, Equation B.11 gives

$$G_t - V(s_t) = \sum_{k=t}^{n-1}\gamma^{k-t}\delta^V_k = -\widehat{\Delta}(s_t) + \gamma\lambda\widehat{\Delta}(s_{t+1}). \tag{B.18}$$

The selection rule gives

$$\frac{\widehat{\Delta}(s_t)}{(n-t)^\xi} > \mathrm{baseline} \geq 0 \quad\Longrightarrow\quad \widehat{\Delta}(s_t) > 0.$$

For $t < n-1$, suffix closure and maximality imply

$$\frac{\widehat{\Delta}(s_{t+1})}{(n-t-1)^\xi} \leq \frac{\widehat{\Delta}(s_t)}{(n-t)^\xi} \quad\Longrightarrow\quad \widehat{\Delta}(s_{t+1}) \leq \left(\frac{n-t-1}{n-t}\right)^\xi \widehat{\Delta}(s_t). \tag{B.19}$$

Combining Equations B.18–B.19 yields

$$G_t - V(s_t) \leq -\left[1 - \gamma\lambda\left(\frac{n-t-1}{n-t}\right)^\xi\right]\widehat{\Delta}(s_t) \leq 0. \tag{B.20}$$

At the final transition,

$$t = n-1 \quad\Longrightarrow\quad G_t - V(s_t) = -\widehat{\Delta}(s_t) < 0.$$

Part II: Fresh-suffix return bound.

For the fresh on-policy suffix τ', use primes for its sampled quantities and let $\mathcal{F}'_k$ be the pre-action information. Let $\widehat{A}'_k$ be its chain GAE from Equation A.5, under the same V. On-policy sampling and the Bellman equation give, for every $c \in \mathbb{R}$,

$$\mathbb{E}[\widehat{A}'_k \mid \mathcal{F}'_k] = 0 \quad \Longrightarrow \quad \mathbb{E}[\widehat{A}'_k \mathbf{1}\{\widehat{A}'_k > c\} \mid \mathcal{F}'_k] \geq 0. \tag{B.21}$$

For $0 < \lambda \leq 1$, we prove by backward induction that, for every threshold c,

$$\mathbb{E}\Big[\big(G'_k - V(s'_k) - \widehat{A}'_k\big)\mathbf{1}\{\widehat{A}'_k > c\} \mid \mathcal{F}'_k\Big] \geq 0. \tag{B.22}$$

Base case. At the final transition,

$$G'_{n-1} - V(s'_{n-1}) = \delta'_{n-1} = \widehat{A}'_{n-1}.$$

Induction step. Assume Equation B.22 holds at $k+1$ for every threshold. Subtracting the GAE recursion from the return recursion gives

$$\begin{aligned} G'_k - V(s'_k) - \widehat{A}'_k &= \gamma\big(G'_{k+1} - V(s'_{k+1}) - \lambda\widehat{A}'_{k+1}\big) \\ &= \gamma\big(G'_{k+1} - V(s'_{k+1}) - \widehat{A}'_{k+1}\big) + \gamma(1-\lambda)\widehat{A}'_{k+1}. \end{aligned}$$

Given $\mathcal{F}'_{k+1}$, the first TD residual is known and

$$\widehat{A}'_k > c \quad \Longleftrightarrow \quad \widehat{A}'_{k+1} > \frac{c - \delta'_k}{\gamma\lambda}.$$

The induction hypothesis and Equation B.21 therefore yield

$$\begin{aligned} &\mathbb{E}\Big[\big(G'_k - V(s'_k) - \widehat{A}'_k\big)\mathbf{1}\{\widehat{A}'_k > c\} \mid \mathcal{F}'_{k+1}\Big] \\ &= \gamma \underbrace{\mathbb{E}\Big[\big(G'_{k+1} - V(s'_{k+1}) - \widehat{A}'_{k+1}\big)\mathbf{1}\{\widehat{A}'_k > c\} \mid \mathcal{F}'_{k+1}\Big]}_{\geq 0} \\ &\quad + \gamma(1-\lambda) \underbrace{\mathbb{E}\Big[\widehat{A}'_{k+1}\mathbf{1}\{\widehat{A}'_k > c\} \mid \mathcal{F}'_{k+1}\Big]}_{\geq 0} \geq 0. \end{aligned}$$

Taking $\mathbb{E}[\cdot \mid \mathcal{F}'_k]$ completes the induction.

Initial-state conclusion. Equations B.21–B.22 give

$$\mathbb{E}[(G'_t - V(s_t))\mathbf{1}\{\widehat{A}'_t > c\} \mid s_t] \geq \mathbb{E}[\widehat{A}'_t \mathbf{1}\{\widehat{A}'_t > c\} \mid s_t] \geq 0. \tag{B.23}$$

At $\lambda = 0$, $\widehat{A}'_t = \delta'_t$ and the later TD residuals have zero conditional contribution, so

$$\mathbb{E}[(G'_t - V(s_t))\mathbf{1}\{\widehat{A}'_t > c\} \mid s_t] = \mathbb{E}[\delta'_t \mathbf{1}\{\delta'_t > c\} \mid s_t] \geq 0.$$

Part III: Budget monotonicity.

The new suffix is selected when $\widehat{A}'_t > \widehat{A}_{\max}(x)$, with the greedy prefix unchanged. Conditional on $\mathcal{T}_j$, independence of the fresh rollout and Equations B.20 and B.23 give

$$\begin{aligned} &\gamma^t \mathbb{E}\Big[(G'_t - G_t)\mathbf{1}\{\widehat{A}'_t > \widehat{A}_{\max}(x)\} \mid \mathcal{T}_j\Big] \\ &= \gamma^t \mathbb{E}\Big[(G'_t - V(s_t))\mathbf{1}\{\widehat{A}'_t > \widehat{A}_{\max}(x)\} \mid \mathcal{T}_j\Big] \\ &\quad + \gamma^t\big(V(s_t) - G_t\big)\Pr\Big(\widehat{A}'_t > \widehat{A}_{\max}(x) \mid \mathcal{T}_j\Big) \\ &\geq \gamma^t\big(V(s_t) - G_t\big)\Pr\Big(\widehat{A}'_t > \widehat{A}_{\max}(x) \mid \mathcal{T}_j\Big) \geq 0. \end{aligned} \tag{B.24}$$

Terminated runs contribute zero increment. The tower property yields

$$\begin{aligned} &J(\pi^S_{j+1}) - J(\pi^S_j) \\ &= \mathbb{E}_{s_o\sim\rho_0,\mathcal{T}_j}\Big[\gamma^t \mathbb{E}\Big[(G'_t - G_t)\mathbf{1}\{\widehat{A}'_t > \widehat{A}_{\max}(x)\} \mid \mathcal{T}_j\Big]\Big] \geq 0. \end{aligned}$$

Iterating from $\pi^S_0 = \pi$ proves Equation B.17. For strictness at an expansion j,

$$\Pr\Big(G_t < V(s_t),\ \widehat{A}'_t > \widehat{A}_{\max}(x)\Big) > 0 \quad \Longrightarrow \quad J(\pi^S_{j+1}) > J(\pi^S_j).$$

□

At $\lambda = 0$, OPTS selects the greedy-Q^π path; at $\lambda = 1$, it selects the largest-return path. Intermediate values trade off preserving the best complete return already found against selecting high-Q^π states from which subsequent rebranching can improve it.

C OPTS-TTPO: Search-Enhanced Policy Gradients

The Branch Aggregation Lemma is the foundation for learning from prefix-measurable trees, and the performance-difference estimate allocates rollout budget. OPTS deliberately departs from the lemma's sampling condition by using observed suffixes to decide where to expand and, under max backup, which discovered suffix to propagate. Each new suffix remains conditionally sampled from the current policy, but the posterior decisions change the induced tree distribution. This section explains the structure contributed by that posterior information and bounds the resulting gradient bias.

We use uniform local branch weights $\alpha_{p,c} = 1/|\mathcal{C}(p)|$, with W given by Equation A.11, to account for branch multiplicity. For a fixed state s_t and action sequence $a_{t:n-1}$, deterministic dynamics uniquely determine the suffix return G_t; replaying the same sequence therefore reproduces its return. In this setting, we use max-backup TreeGAE, the backup rule under which Appendix B establishes the search-improvement guarantees. Under stochastic dynamics, G_t need not be uniquely determined even after conditioning on the same action sequence. Maximizing a finite set of backed-up suffix scores can then select both suffix quality and environmental noise, causing high-variance return distributions to be favored over suffixes with higher conditional expected returns. We therefore use mean-backup TreeGAE to average sampled suffixes and reduce this max-selection effect while retaining the coverage gained from branching. Adaptive expansion affects both variants; max backup contributes the additional prefix-credit term analyzed below.

C.1 Posterior Max Backup as Prefix Credit

Fix a tree $\mathcal{T}$ sampled from $\pi_{\theta_{\text{old}}}$, a critic V, and a common λ. Using this critic, compute $\widehat{A}_{\max}$ by Equation B.11 and $\widehat{A}_{\text{mean}}$ by Equation A.27 with the uniform weights above. Let $\widehat{g}_{\max}$ and $\widehat{g}_{\text{mean}}$ denote their TTPG directions, evaluated at θ_{old}:

$$\widehat{g}_{\max} := \sum_{x\in\mathcal{T}} W(x)\gamma^{d(x)}\widehat{A}_{\max}(x)\nabla_\theta \log \pi_\theta(a_x \mid s_x)\Big|_{\theta=\theta_{\text{old}}}, \tag{C.1}$$

with $\widehat{g}_{\text{mean}}$ defined analogously. The tree, critic, weights, and backed-up marks are held fixed during this actor update.

At a transition x with children, define the best-minus-average suffix gap

$$b(x) := \max_{c\in\mathcal{C}(x)} \widehat{A}_{\max}(c) - \sum_{c\in\mathcal{C}(x)} \alpha_{x,c}\widehat{A}_{\max}(c) \geq 0, \tag{C.2}$$

and set $b(x) = 0$ at a leaf. Write $x \preceq u$ when x lies on the root-to-u path, including u itself.

Theorem 3 (Posterior Prefix Reinforcement). For every such tree, define the frozen-tree surrogate

$$\mathcal{L}_{\text{search}}(\theta) := \gamma\lambda \sum_{u\in\mathcal{T}} W(u)\gamma^{d(u)}b(u) \sum_{x\preceq u} \lambda^{d(u)-d(x)} \log \pi_\theta(a_x \mid s_x). \tag{C.3}$$

Then

$$\widehat{g}_{\max} = \widehat{g}_{\text{mean}} + \nabla_\theta \mathcal{L}_{\text{search}}(\theta)|_{\theta=\theta_{\text{old}}}. \tag{C.4}$$

Consequently, at θ_{old}, their directional derivatives of this surrogate satisfy

$$\nabla_\theta \mathcal{L}_{\text{search}}^\top \widehat{g}_{\max} - \nabla_\theta \mathcal{L}_{\text{search}}^\top \widehat{g}_{\text{mean}} = \|\nabla_\theta \mathcal{L}_{\text{search}}\|^2 \geq 0. \tag{C.5}$$

Proof. **Part I: Propagation of the suffix gap.** Subtracting the two backup recursions gives

$$\begin{aligned} &\widehat{A}_{\max}(x) - \widehat{A}_{\text{mean}}(x) \\ &= \gamma\lambda \left[b(x) + \sum_{c \in \mathcal{C}(x)} \alpha_{x,c} \big(\widehat{A}_{\max}(c) - \widehat{A}_{\text{mean}}(c) \big) \right]. \end{aligned} \tag{C.6}$$

Expanding to the leaves and using the product definition of W yields

$$W(x)\big(\widehat{A}_{\max}(x) - \widehat{A}_{\text{mean}}(x)\big) = \gamma\lambda \sum_{u:\, x \preceq u} W(u)(\gamma\lambda)^{d(u)-d(x)} b(u). \tag{C.7}$$

Part II: Credit assigned to the prefix. Substitute Equation C.7 into the difference of the TTPG sums and exchange the ancestor–descendant sums:

$$\begin{aligned} &\widehat{g}_{\max} - \widehat{g}_{\text{mean}} \\ &= \gamma\lambda \sum_{u \in \mathcal{T}} W(u)\gamma^{d(u)} b(u) \sum_{x \preceq u} \lambda^{d(u)-d(x)} \nabla_\theta \log \pi_\theta(a_x \mid s_x) \Big|_{\theta=\theta_{\text{old}}}. \end{aligned} \tag{C.8}$$

This is the gradient of Equation C.3. Taking its inner product with that gradient proves Equation C.5. □

What posterior max backup contributes. In a deterministic environment with $V = V^\pi$, Equation B.15 identifies each max-backup mark with the best retained suffix λ-return relative to its source-state value. Thus $b(x)$ measures the best-minus-average λ-return gap among the retained child suffixes. Max backup passes this relative gain to the actions leading to the branch point. At $\lambda = 1$, the inner sum in Equation C.3 is the log-likelihood of the action prefix through u; for $0 < \lambda < 1$, the additional credit decays with the distance to that branch point. At $\lambda = 0$, both backups reduce to the local TD residual.

Equation C.5 quantifies the additional first-order change in this weighted prefix objective under a max-backup update. It explains how a discovered suffix can influence upstream learning before the current policy reliably reproduces that suffix. Subsequent on-policy rollouts and downstream updates determine how this credit is translated into realized return.

The identity is pathwise, so it also holds after averaging the two directions over the same OPTS tree distribution. Adaptive expansion determines which suffixes are available; posterior max backup determines how their relative values are propagated. We account for both effects below.

C.2 Controlling the Search-Induced Gradient Bias

Draw an initial trajectory $\tau^{(0)} \sim \pi_{\theta_{\text{old}}}$ and let $j \in \{0, \ldots, S_{\max}\}$ be its random number of OPTS expansions. Allocation and stopping may depend on all observed search outcomes. Evaluate $\widehat{g}_{\max}$ and $\widehat{g}_{\text{mean}}$ on the resulting tree $\mathcal{T}_j$, and write $\overline{g}_{\max} := \mathbb{E}[\widehat{g}_{\max}]$. The mean-backup direction contains the adaptive tree-selection effect; the difference $\widehat{g}_{\max} - \widehat{g}_{\text{mean}}$ isolates the additional prefix-credit contribution of posterior max backup. Let $\widehat{g}_{\text{ch}}(\tau)$ be the chain-GAE policy-gradient sum with the same critic and discount, and define $\varepsilon_0 := \|\mathbb{E}[\widehat{g}_{\text{ch}}(\tau^{(0)})] - \nabla J(\theta_{\text{old}})\|$. This initial-chain bias vanishes with exact advantages. For chain GAE, exact $V^{\pi_{\theta_{\text{old}}}}$ likewise gives

$$\mathbb{E}[\widehat{g}_{\text{ch}}(\tau^{(0)})] = \nabla J(\theta_{\text{old}}), \qquad \varepsilon_0 = 0,$$

by the chain policy-gradient and GAE identities.

Suppose $|r| \le R_{\max}$, $|V| \le V_{\max}$, and $\| \nabla_\theta \log \pi_\theta(a \mid s)|_{\theta=\theta_{\text{old}}} \| \le L_\pi$. The TD residual satisfies

$$|r + \gamma V(s') - V(s)| \le |r| + \gamma|V(s')| + |V(s)| \le R_{\max} + (1+\gamma)V_{\max}.$$

For either tree backup,

$$\left|\sum_c \alpha_{x,c}\widehat{A}(c)\right| \le \sum_c \alpha_{x,c}|\widehat{A}(c)| \le \max_c |\widehat{A}(c)|, \qquad \left|\max_c \widehat{A}(c)\right| \le \max_c |\widehat{A}(c)|.$$

Recursing from the leaves over at most n steps gives the uniform absolute bound for chain GAE and both tree backups:

$$A_{\text{bd}} := \big[R_{\max} + (1+\gamma)V_{\max}\big] \sum_{t=0}^{n-1} (\gamma\lambda)^t. \tag{C.9}$$

With $C_n := \sum_{t=0}^{n-1} \gamma^t$, the normalized branch weights have total mass one at each depth, so every corresponding tree or chain direction has norm at most $C_n A_{\text{bd}} L_\pi$.

Let

$$p := \Pr(j > 0), \qquad \rho_S := \mathbb{E}[1 - 2^{-j}] \le p(1 - 2^{-S_{\max}}). \tag{C.10}$$

Here p is the population fraction of root trees receiving at least one search expansion, and ρ_S bounds the expected total path weight assigned to paths other than the initial trajectory. The extra population-gradient contribution of max backup is measured by

$$\varepsilon_{\text{backup}} := \|\mathbb{E}\left[\widehat{g}_{\max} - \widehat{g}_{\text{mean}}\right]\|. \tag{C.11}$$

Proposition 2 (Search-Induced Gradient Bias Bound). Under the preceding conditions,

$$\begin{aligned} &\|\overline{g}_{\max} - \nabla J(\theta_{\text{old}})\| \\ &\le \varepsilon_0 + \min\left\{2pC_nA_{\text{bd}}L_\pi,\ 2\rho_S C_n A_{\text{bd}} L_\pi + \varepsilon_{\text{backup}}\right\}, \end{aligned} \tag{C.12}$$

where the backup contribution satisfies

$$\varepsilon_{\text{backup}} \le \gamma\lambda L_\pi \mathbb{E}\left[\sum_{u \in \mathcal{T}_j} W(u)\gamma^{d(u)} b(u) \sum_{r=0}^{d(u)} \lambda^r\right]. \tag{C.13}$$

Proof. Part I: Adaptive sampling and branch weighting. For each retained path $\tau = (x_0, \ldots, x_{n-1}) \in \mathcal{P}_j$, let $w_\tau := W(x_{n-1})$. Expanding the mean recursion gives the pathwise identity

$$\widehat{g}_{\text{mean}} = \sum_{\tau \in \mathcal{P}_j} w_\tau \widehat{g}_{\text{ch}}(\tau), \qquad \sum_\tau w_\tau = 1. \tag{C.14}$$

This uses the direct–recursive aggregation identity from Lemma 1. Adding one child to a branch point with m children on the initial path multiplies its existing local weight by $m/(m+1) \ge 1/2$. Expansions elsewhere leave that path's weight unchanged. Hence $w_{\tau^{(0)}} \ge 2^{-j}$ and

$$\begin{aligned} &\left\|\widehat{g}_{\text{mean}} - \widehat{g}_{\text{ch}}(\tau^{(0)})\right\| \\ &= \left\|\sum_\tau w_\tau \left[\widehat{g}_{\text{ch}}(\tau) - \widehat{g}_{\text{ch}}(\tau^{(0)})\right]\right\| \\ &= \left\|\sum_{\tau \ne \tau^{(0)}} w_\tau \left[\widehat{g}_{\text{ch}}(\tau) - \widehat{g}_{\text{ch}}(\tau^{(0)})\right]\right\| \\ &\le \sum_{\tau \ne \tau^{(0)}} w_\tau \left\|\widehat{g}_{\text{ch}}(\tau) - \widehat{g}_{\text{ch}}(\tau^{(0)})\right\| \\ &\le 2C_n A_{\text{bd}} L_\pi (1 - w_{\tau^{(0)}}). \end{aligned} \tag{C.15}$$

The initial trajectory retains its on-policy marginal law. By the triangle inequality and $\|\mathbb{E}[\cdot]\| \le \mathbb{E}[\|\cdot\|]$,

$$\begin{aligned}
&\|\mathbb{E}[\widehat{g}_{\mathrm{mean}}] - \nabla J(\theta_{\mathrm{old}})\| \\
&= \left\|\mathbb{E}[\widehat{g}_{\mathrm{mean}} - \widehat{g}_{\mathrm{ch}}(\tau^{(0)})] + \mathbb{E}[\widehat{g}_{\mathrm{ch}}(\tau^{(0)})] - \nabla J(\theta_{\mathrm{old}})\right\| \\
&\le \varepsilon_0 + \left\|\mathbb{E}[\widehat{g}_{\mathrm{mean}} - \widehat{g}_{\mathrm{ch}}(\tau^{(0)})]\right\| \\
&\le \varepsilon_0 + \mathbb{E}\left[\left\|\widehat{g}_{\mathrm{mean}} - \widehat{g}_{\mathrm{ch}}(\tau^{(0)})\right\|\right] \\
&\le \varepsilon_0 + 2C_n A_{\mathrm{bd}} L_\pi \, \mathbb{E}[1 - w_{\tau^{(0)}}] \\
&\le \varepsilon_0 + 2C_n A_{\mathrm{bd}} L_\pi \, \mathbb{E}[1 - 2^{-j}] \\
&= \varepsilon_0 + 2\rho_S C_n A_{\mathrm{bd}} L_\pi .
\end{aligned} \tag{C.16}$$

Part II: Posterior max backup. Equation C.4 separates the two contributions exactly:

$$\begin{aligned}
&\overline{g}_{\mathrm{max}} - \nabla J(\theta_{\mathrm{old}}) \\
&= \mathbb{E}[\widehat{g}_{\mathrm{mean}}] - \nabla J(\theta_{\mathrm{old}}) \\
&\quad + \mathbb{E}[\widehat{g}_{\mathrm{max}} - \widehat{g}_{\mathrm{mean}}].
\end{aligned} \tag{C.17}$$

By the triangle inequality and Part I,

$$\begin{aligned}
&\|\overline{g}_{\mathrm{max}} - \nabla J(\theta_{\mathrm{old}})\| \\
&\le \|\mathbb{E}[\widehat{g}_{\mathrm{mean}}] - \nabla J(\theta_{\mathrm{old}})\| + \|\mathbb{E}[\widehat{g}_{\mathrm{max}} - \widehat{g}_{\mathrm{mean}}]\| \\
&\le \varepsilon_0 + 2\rho_S C_n A_{\mathrm{bd}} L_\pi + \varepsilon_{\mathrm{backup}},
\end{aligned}$$

which gives the second alternative in Equation C.12. Using Equation C.8 and the nonnegative coefficients,

$$\begin{aligned}
\varepsilon_{\mathrm{backup}} &= \|\mathbb{E}[\widehat{g}_{\mathrm{max}} - \widehat{g}_{\mathrm{mean}}]\| \\
&\le \mathbb{E}[\|\widehat{g}_{\mathrm{max}} - \widehat{g}_{\mathrm{mean}}\|] \\
&= \gamma\lambda \, \mathbb{E}\left[\left\|\sum_{u\in\mathcal{T}_j} W(u)\gamma^{d(u)} b(u) \sum_{x\preceq u} \lambda^{d(u)-d(x)} \, \nabla_\theta \log \pi_\theta(a_x \mid s_x)|_{\theta=\theta_{\mathrm{old}}}\right\|\right] \\
&\le \gamma\lambda \, \mathbb{E}\left[\sum_{u\in\mathcal{T}_j} W(u)\gamma^{d(u)} b(u) \sum_{x\preceq u} \lambda^{d(u)-d(x)} \left\| \nabla_\theta \log \pi_\theta(a_x \mid s_x)|_{\theta=\theta_{\mathrm{old}}}\right\|\right] \\
&\le \gamma\lambda L_\pi \, \mathbb{E}\left[\sum_{u\in\mathcal{T}_j} W(u)\gamma^{d(u)} b(u) \sum_{x\preceq u} \lambda^{d(u)-d(x)}\right] \\
&= \gamma\lambda L_\pi \, \mathbb{E}\left[\sum_{u\in\mathcal{T}_j} W(u)\gamma^{d(u)} b(u) \sum_{r=0}^{d(u)} \lambda^{r}\right],
\end{aligned}$$

which proves Equation C.13.

Part III: A uniform cap. On $j = 0$, the max-backup direction equals the initial chain direction. Consequently,

$$\begin{aligned}
&\|\overline{g}_{\mathrm{max}} - \nabla J(\theta_{\mathrm{old}})\| \\
&\le \varepsilon_0 + \mathbb{E}\left[\mathbf{1}\{j > 0\} \left\|\widehat{g}_{\mathrm{max}} - \widehat{g}_{\mathrm{ch}}(\tau^{(0)})\right\|\right] \\
&\le \varepsilon_0 + 2pC_n A_{\mathrm{bd}} L_\pi,
\end{aligned} \tag{C.18}$$

which proves the first alternative. □

Dependence on the search budget. Both search contributions vanish on unexpanded trees. Since $|\widehat{A}_{\max}(c)| \le A_{\mathrm{bd}}$,

$$0 \le b(u) \le A_{\mathrm{bd}} - (-A_{\mathrm{bd}}) = 2A_{\mathrm{bd}}.$$

A node with one child has $b(u) = 0$, and each expansion adds one suffix. Thus

$$\big|\{u \in \mathcal{T}_j : b(u) > 0\}\big| \le j, \qquad \sum_{u:\, b(u)>0} W(u)\gamma^{d(u)} \le j.$$

The unit branch mass at each depth also gives

$$\sum_{u:\, b(u)>0} W(u)\gamma^{d(u)} \le \sum_{t=0}^{n-1} \gamma^t \sum_{u:\, d(u)=t} W(u) = C_n.$$

Combining these bounds with Equation C.13,

$$\begin{aligned}
\varepsilon_{\text{backup}} &\le 2\gamma\lambda A_{\text{bd}} L_\pi \, \mathbb{E}\left[\sum_{u:\, b(u)>0} W(u)\gamma^{d(u)}\right] \sum_{r=0}^{n-1} \lambda^r \\
&\le 2\gamma\lambda A_{\text{bd}} L_\pi \, \mathbb{E}[\min\{j, C_n\}] \sum_{r=0}^{n-1} \lambda^r \\
&\le 2\gamma\lambda A_{\text{bd}} L_\pi \, \mathbb{E}[\mathbf{1}\{j > 0\}] \min\{S_{\max}, C_n\} \sum_{r=0}^{n-1} \lambda^r \\
&= 2p\gamma\lambda A_{\text{bd}} L_\pi \min\{S_{\max}, C_n\} \sum_{r=0}^{n-1} \lambda^r. \qquad \text{(C.19)}
\end{aligned}$$

Thus the searched-tree fraction scales both contributions, the expansion cap bounds the number of posterior decisions, and the suffix gaps quantify the additional credit actually introduced by max backup. Normalized branch weights keep the total gradient scale bounded as the tree grows. At $S_{\max} = 0$, the bound reduces to ε_0, and exact values recover the unbiased chain policy gradient.

C.3 OPTS-TTPO Training Procedure

Algorithm 2 OPTS-TTPO training step

Require: π_θ, P, ρ_0, $\widehat{V}$, B, R, $S_{\max}$, ξ, baseline

1: $\mathcal{B} \leftarrow \text{OPTS}(\pi_\theta, P, \rho_0, \widehat{V}, B, R, S_{\max}, \xi, \text{baseline})$ by Algorithm 1
2: Set $\alpha_{p,c} \leftarrow 1/|\mathcal{C}(p)|$ and compute $W(x)$ by Equation A.11
3: Construct TTPO losses using the TreeGAE advantages in $\mathcal{B}$ and branch weights $W(x)$ by Section 3.1
4: Update the critic with $\mathcal{L}_V^{\text{TTPO}}$ and the actor with $\mathcal{L}_\pi^{\text{TTPO}}$

D Experimental Setup

We describe the protocols in the same order as the main text. Shared optimization details, data-overlap auditing, prompt formatting, and the language-model MDP formulation follow these experiment-specific settings.

D.1 Tree-Gradient Estimation: Setup and Protocol

For the gradient-aggregation experiment in Section 4.1, we freeze a step-400 `Qwen3-1.7B` PPO actor and use 16,384 training prompts. The estimator side contains eight midpoint-branch tree groups with $K \in \{0, 1, 3, 7, 15\}$ additional suffixes; the reference side contains 32 independently sampled chain groups. Both sides use global token-level aggregation. The reference gradient, tree construction, and metrics are described below.

Reference gradient. We freeze the step-400 `Qwen3-1.7B` PPO checkpoint and evaluate all gradients on the 16,384-prompt training set. Returns are binary verifier outcomes, with $\gamma = \lambda = 1$

and no learned baseline. For a collection $\mathcal{C}$ of chain responses, define

$$N(\mathcal{C}) := \sum_{\tau \in \mathcal{C}} \sum_{x \in \tau} R(\tau) \nabla_\theta \log \pi_\theta(a_x \mid s_x), \qquad D(\mathcal{C}) := \sum_{\tau \in \mathcal{C}} |\tau|, \tag{D.1}$$

where x ranges over valid response-token transitions. We independently sample 32 chain groups, each containing one response per prompt, and form the reference over their union $\mathcal{C}^\star$ as $g^\star = N(\mathcal{C}^\star)/D(\mathcal{C}^\star)$. This reference is finite; reported bias is deviation from $g^\star$, not population bias.

Tree gradients. We independently sample eight tree groups, each containing one tree per prompt. Each tree branches at the midpoint of its backbone response, retains the original suffix, and samples 15 additional on-policy suffixes. The settings $K \in \{0, 1, 3, 7, 15\}$ use nested subsets of the first K additional suffixes. Within a tree, let R_j be the binary return of suffix $j \in \{0, \dots, K\}$. The shared-prefix target is the branch mean $\bar{R}_K := (K+1)^{-1} \sum_{j=0}^{K} R_j$, while every token on suffix j uses R_j.

For $A \in \{\mathrm{NaivePG}, \mathrm{TTPG}\}$, let $w^A(x) = 1$ for NaivePG and $w^A(x) = W(x)$ for TTPG. Under the single uniform branch point used here, $W(x) = 1$ on the shared prefix and $W(x) = 1/(K+1)$ on every suffix. Writing $\widetilde{R}_K(x)$ for the assigned prefix or suffix target of x and $\mathcal{X}_{i,K}$ for all valid response-token occurrences in group i, we accumulate

$$N^A_{i,K} := \sum_{x \in \mathcal{X}_{i,K}} w^A(x) \widetilde{R}_K(x) \nabla_\theta \log \pi_\theta(a_x \mid s_x), \qquad D^A_{i,K} := \sum_{x \in \mathcal{X}_{i,K}} w^A(x). \tag{D.2}$$

Every $M \in \{1, 2, 4\}$ consecutive groups form one non-overlapping block $\mathcal{B}_r$ and one globally normalized estimate

$$\hat{g}^A_{r,K,M} = \frac{\sum_{i \in \mathcal{B}_r} N^A_{i,K}}{\sum_{i \in \mathcal{B}_r} D^A_{i,K}}, \qquad |\mathcal{B}_r| = M, \qquad R_M = \frac{8}{M}. \tag{D.3}$$

Thus each estimate merges group numerators and token-weight masses before division.

Metrics. For each A, K, and M, write $g_r = \hat{g}^A_{r,K,M}$ and $\bar{g} = R_M^{-1} \sum_{r=1}^{R_M} g_r$. We report

$$\mathrm{Bias} = \frac{\|\bar{g} - g^\star\|_2}{\|g^\star\|_2}, \qquad \mathrm{Var} = \frac{\sum_{r=1}^{R_M} \|g_r - \bar{g}\|_2^2}{(R_M - 1)\|g^\star\|_2^2}, \tag{D.4}$$

$$\mathrm{MSE} = \frac{\sum_{r=1}^{R_M} \|g_r - g^\star\|_2^2}{R_M \|g^\star\|_2^2}, \qquad \mathrm{Cos} = \frac{1}{R_M} \sum_{r=1}^{R_M} \frac{\langle g_r, g^\star \rangle}{\|g_r\|_2 \|g^\star\|_2}. \tag{D.5}$$

D.2 OPTS Search and Test-Time Scaling

Exact-value experiment. We construct 32 fixed environment–policy configurations (seeds 0–31). Each environment is a depth-4 deterministic binary tree with 15 nonterminal states and 16 terminal states. Intermediate rewards are zero; rewards on terminal transitions are a random permutation of $\{0/15, 1/15, \dots, 15/15\}$. At each nonterminal state, the right-action probability is drawn independently and uniformly from $\{1/4, 1/3, 1/2, 2/3, 3/4\}$, with the complementary probability assigned to the left action. Reward permutations and policy probabilities use separate random streams. The policy remains fixed, and its exact V^π is computed recursively with $\gamma = 1$.

Both guidance modes use max-backup TreeGAE, $\xi = 0$, a zero search baseline, and $\lambda \in \{0, 0.3, 0.6, 0.95\}$. Reward-guided OPTS uses full advantages including the terminal reward. Value-guided OPTS truncates advantage computation before the terminal transition and bootstraps with exact V^π at the last nonterminal state, equivalently setting the final TD residual to zero. Both modes sample complete on-policy suffixes; only the advantage computation is truncated. Each search round selects the largest performance-difference estimate on the current greedy path and samples one new suffix. Search stops if no estimate exceeds zero; max-backup ties retain the incumbent path, and ties between candidate positions select the earliest position.

We enumerate all possible initial trajectories and subsequent suffix samples, weighting them by their policy probabilities to obtain exact expectations for every budget $j \in \{0, 1, \dots, 15\}$. Both modes

are evaluated using the full-reward J_λ and true return J along their guidance-greedy paths. For each mode and metric, the monotonicity check comprises $32 \times 4 \times 15 = 1{,}920$ comparisons between adjacent budgets. Figure 3 (top) plots improvements relative to $j = 0$; curves and shaded bands show the mean and $\pm$ one sample standard deviation across the 32 configurations.

Learned-critic experiment. For the search-budget experiment in Section 4.2, we freeze a step-400 `Qwen3-1.7B` OPTS-TTPO actor and critic, use 902 held-out prompts with 32 trees per prompt, and vary $S_{\max} \in \{0, 1, 3, 7, 15\}$ with $\lambda = 0.999$. Reward-guided and value-guided search reuse the same root-response texts. We report the full verifier-reward J_λ and terminal verifier return J along the guidance-greedy paths, relative to $S_{\max} = 0$.

Matched-budget test-time scaling. For matched rollout-budget test-time scaling, we evaluate fixed $S_{\max} = 3$ at budgets $k \in \{8, 16, 32, 64, 128\}$, corresponding to $k \times 902$ rollouts. A fresh root rollout and a suffix rollout from a branch point each count as one rollout. Every search round contains 902 rollouts, with unused rebranching slots filled by fresh initial trees. Reward-guided OPTS is compared with i.i.d. `pass@k`; value-guided OPTS uses the learned value function and truncated GAE and returns the majority answer among value-greedy tree responses, with i.i.d. self-consistency (`cons@k`) as its baseline. OPTS and i.i.d. baselines use temperature 1.0, top-$p = 0.95$, and no top-k truncation.

D.3 Coverage–Bias and Prefix-Credit Mechanism Protocol

This protocol produces the two panels in Section 4.3. We freeze the step-400 `Qwen3-1.7B` OPTS-TTPO actor–critic checkpoint; it is distinct from the PPO checkpoint used for the branch-aggregation experiment. We run the experiment on 16,384 training prompts. The fixed-branch method branches at response token 128 and skips responses shorter than 128 tokens; the other method runs OPTS.

Matched coverage and bias comparison. For the OPTS side, we retain eight trees per prompt and record the search snapshots $s \in \{0, 1, 3, 7\}$, with $p = 1$. OPTS selects the rebranching trees using its performance-difference rule. The fixed controls branch at response token 128. At every prompt and search round, each fixed control receives exactly the number of branches selected by OPTS for that prompt and round; its fixed-position trees are sampled uniformly without replacement and do not use their own rewards for allocation. The plotted methods are Fixed + TTPG, Fixed + NaivePG, and OPTS + TTPG (the OPTS max-backup snapshot). We use $M = 1$ theory aggregation and recompute the gradient coefficients from direct returns with $V = 0$, so panel (a) isolates the coverage–bias relation rather than learned-critic error. The horizontal coordinate is the relative bias difference from $s = 0$; the vertical coordinate is the prompt-coverage gain in percentage points.

Prefix-credit comparison. Using the same OPTS trees and snapshots, we reconstruct max- and mean-backup return advantages with $V = 0$ and compare their difference at shared prefixes. Panel (b) reports token-weighted mean extra credit, grouped by the distance in tokens to the first downstream branch, for $s \in \{1, 3, 7\}$. Solid curves use $\lambda = 1$ and dashed curves use $\lambda = 0.999$. Thus the panel measures the credit introduced by max backup on the searched trees; it does not estimate a separate critic effect.

D.4 Cross-Domain Policy Learning

MuJoCo. The suite contains `Hopper-v4`, `Walker2d-v4`, `HalfCheetah-v4`, `Ant-v4`, and `Humanoid-v4`. PPO and OPTS-TTPO use one million environment steps and ten random seeds per task. OPTS-TTPO uses max-backup TreeGAE and action-level performance-difference selection with $(\xi, S_{\max}) = (0.6, 1)$.

Atari-57. We use `NoFrameskip` variants with sticky actions ($p_{\text{repeat}} = 0.25$ per emulated frame), agent frame skip 4, frame max-pooling, 84×84 grayscale observations, four-frame stacking, reward clipping, and up to 30 no-ops on reset. CleanRL-style PPO uses a CNN actor–critic. Each method is trained for 10M environment steps with eight parallel environments, 128-step rollouts, and three random seeds per game. OPTS-TTPO reuses $(\xi, S_{\max}) = (0.6, 1)$, uses mean-backup TreeGAE, and selects states with the action-level performance-difference estimate. We summarize each run by the

mean return over all logged points and over the final 100 logged points; IQM is computed over the 57 games and three seeds after human normalization, and task wins compare the three-seed mean within each game.

LLM RLVR. We train four Qwen3 models with VeRL for 400 steps on `math12k` and the competition subset of `NuminaMath-1.5-RL-Verifiable` (16K prompts), comparing OPTS-TTPO with PPO, DAPO, and REINFORCE++ at 4,096 completed rollouts per update. OPTS-TTPO uses eight rounds of 512 trajectories. At each selection step, $p = 0.3$ caps the fraction of established trees that may enter search. All four models use the same maximum search count $S_{\max} = 3$ for OPTS-TTPO training. Training uses actor/critic learning rates $10^{-6}/10^{-5}$, max-backup TreeGAE with $\lambda = 0.999$, and the zero baseline. The answer verifier assigns reward 1 to a verified correct final answer and 0 otherwise; intermediate response-token rewards are zero, and the terminal response token receives this verifier reward, which TreeGAE propagates to preceding tokens. Evaluation uses `MATH500` (the `math12k` test split), `MinervaMath`, `AMC23`, `AIME24`, `AIME25`, and `AIME26`, containing 500, 272, 40, 30, 30, and 30 problems. The main table reports both equal-weight Macro Average and question-pooled Micro Average; final comparisons use step 400 and 32 independent responses per prompt, sampled at temperature 1.0 and top-$p = 0.95$, with top-$k = -1$ (no truncation) and a maximum response length of 2,048 tokens.

D.5 Optimization Hyperparameters

Within each control domain, PPO and OPTS-TTPO use the same optimizer and update settings; they differ only in tree construction, advantage backup, and branch weighting. Table A1 lists the complete shared settings.

Table A1: Optimization hyperparameters shared by PPO and OPTS-TTPO in the control experiments.

Setting	MuJoCo	Atari-57
Optimizer	Adam, $\epsilon = 10^{-5}$	Adam, $\epsilon = 10^{-5}$
Learning rate	3×10^{-4}, linearly annealed to 0	2.5×10^{-4}, linearly annealed to 0
Rollout batch	$1 \times 2048 = 2048$ transitions	$8 \times 128 = 1024$ transitions
Minibatches / update epochs	32 (64 transitions each) / 10	4 (256 transitions each) / 4
Discount / GAE parameter	$\gamma = 0.99$, $\lambda = 0.95$	$\gamma = 0.99$, $\lambda = 0.95$
Policy clip / clipped value loss	0.2 / yes	0.1 / yes
Entropy / value-loss coefficient	0 / 0.5	0.01 / 0.5
Advantage normalization / gradient norm	yes / 0.5	yes / 0.5
Independent training seeds	10 per task and method	3 per game and method

All LLM methods use AdamW with $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0.1, actor learning rate 10^{-6}, ten warmup steps followed by a constant learning rate, gradient clipping at 1.0, one policy-optimization epoch per update, zero entropy coefficient, and no KL penalty in either the reward or loss. Training rollouts use temperature 1.0, top-$p = 1.0$, no top-k truncation, and a maximum response length of 2,048 tokens. Table A2 gives the method-specific settings.

Table A2: Method-specific hyperparameters for LLM RLVR.

Method	Method-specific settings
PPO	4,096 prompts and rollouts per update; GAE with $\gamma = 1$ and $\lambda = 0.999$; learned critic with learning rate 10^{-5} and value clip 0.5; symmetric policy clip 0.2; token-level loss aggregation.
DAPO	512 prompts with 8 responses each; group-relative advantages and no critic; asymmetric policy clip $(0.2, 0.28)$ with dual-clip constant 10; token-level loss aggregation; overlong-response buffer of 1,024 tokens with penalty factor 1.0.
REINFORCE++	512 prompts with 8 responses each; per-prompt mean reward baseline followed by global advantage whitening, with no critic; symmetric policy clip 0.2; token-level loss aggregation.
OPTS-TTPO	4,096 rollouts collected over the search rounds specified in Appendix D; TreeGAE with a learned critic at learning rate 10^{-5}; symmetric policy clip 0.2; branch-weighted token-level loss aggregation.

D.6 Simulator State Restoration

OPTS-TTPO uses simulator snapshot-and-restore access to sample new suffixes from visited states, while PPO collects ordinary forward rollouts. Interaction budgets count newly generated transitions, including branch transitions; restoring a snapshot does not itself advance the environment or count as an interaction.

MuJoCo. We snapshot joint positions, velocities, simulation time, auxiliary simulator fields, and derived quantities used by observations. Snapshots also preserve the environment random-number-generator state, elapsed episode steps, episode statistics, observation/reward normalization statistics, and the discounted-return accumulator. Restoration reinstates the physical state, calls `mj_forward`, and restores the saved derived quantities and wrapper state. Each environment is restored independently to the selected state and its cached observation before new actions are sampled from the current policy.

Atari. We use ALE's `cloneSystemState` and `restoreSystemState` to save and restore emulator state, including the sticky-action random-number-generator state. Snapshots also preserve frame-stack and max-pooling buffers, life-loss bookkeeping, elapsed-step and episode-statistics counters, and the random state used for reset no-ops. Thus a restored branch resumes the saved environment random stream, while actions are newly sampled from the current policy. Environments maintain separate snapshots, with batched policy inference. After a terminal or truncated trajectory, a selected branch restores its snapshot and observation; otherwise, the environment follows the standard wrapped reset, including episodic-life handling. Each update collects 128 new transitions per environment, bootstrapping nonterminal rollout boundaries with the critic. Tree records are rebuilt after each policy update; unfinished environment trajectories continue, but their carried-over initial segments are excluded from branching.

D.7 Training–Evaluation Overlap Audit

Before constructing the 16,384-prompt LLM training split, we audit the union of the `math12k` and `NuminaMath-1.5-RL-Verifiable` training data against all 902 problems in `MATH500`, `MinervaMath`, `AMC23`, and `AIME24--26`. For each example, we extract the user problem, remove all whitespace, and convert it to lowercase. We remove every training example whose normalized problem exactly matches any normalized evaluation problem, as well as duplicate normalized problems within the merged training pool. We then apply the 1,024-token prompt-length filter, shuffle the remaining pool, and retain 16,384 prompts. The resulting training split has zero exact normalized matches with the six evaluation sets. This audit detects exact problem reuse up to whitespace and letter case; it does not treat paraphrases as matches.

D.8 LLM Prompt and Chat Template

All LLM training and evaluation examples use the same system prompt:

Instruction Prompt for Mathematical Reasoning

You are a math problem solver. For each problem, think through it step by step within <think> </think> tags, then provide your final answer using \boxed{}.

Requirements:
- Show your complete reasoning process inside <think> tags.
- Provide your final answer inside \boxed{}.

Example:
User: If 3x + 7 = 22, what is x?
Assistant: <think>
3x + 7 = 22
3x = 22 - 7 = 15
x = 15 / 3 = 5
</think>
The answer is \boxed{5}.

For every dataset, the original problem text is inserted without an additional task-specific instruction using the following message structure:

```
[
  {"role": "system", "content": SYSTEM_PROMPT},
  {"role": "user",   "content": PROBLEM}
]
```

We do not supply a custom Jinja template. VeRL applies the chat template distributed with the corresponding Qwen3 checkpoint with `add_generation_prompt=True`; thus the two messages are serialized with the checkpoint's system, user, and assistant role delimiters, followed by the assistant generation prefix. Qwen3 thinking mode is retained: evaluation sets `enable_thinking=True` explicitly, while training uses the checkpoint template's default thinking behavior. The same message construction and tokenizer template are used for all four training methods and all six evaluation sets.

D.9 LLM Reasoning as a Markov Decision Process

The initial state $s_o \sim \rho_0$ is a problem prompt. At generation step t, the state contains the prompt and the generated response prefix, and the action is the next response token:

$$s_t = (s_o, a_0, \ldots, a_{t-1}), \qquad a_t \sim \pi_\theta(\cdot \mid s_t).$$

The transition appends a_t to the prefix deterministically. An episode ends at an end-of-sequence token or the response-length limit. Intermediate rewards are zero; the final response token receives the answer-verification score. The critic estimates the remaining return from each prefix, and TreeGAE propagates the terminal reward through the sampled suffixes. Its regression target is $\widehat{R}_x = \widehat{A}_x + V_{\bar{\phi}}(s_x)$, as in Section 3.1.

Rebranching at s_t retains the prompt and response prefix and samples a new suffix from π_θ. Thus each transition occurrence corresponds to a response-token occurrence, and multiple suffixes share their sampled prefix. OPTS selects rebranching positions using the backed-up scores; TTPO aggregates the resulting token contributions using the branch weights $W(x)$.

E Rollout Scaling Across Individual Benchmarks

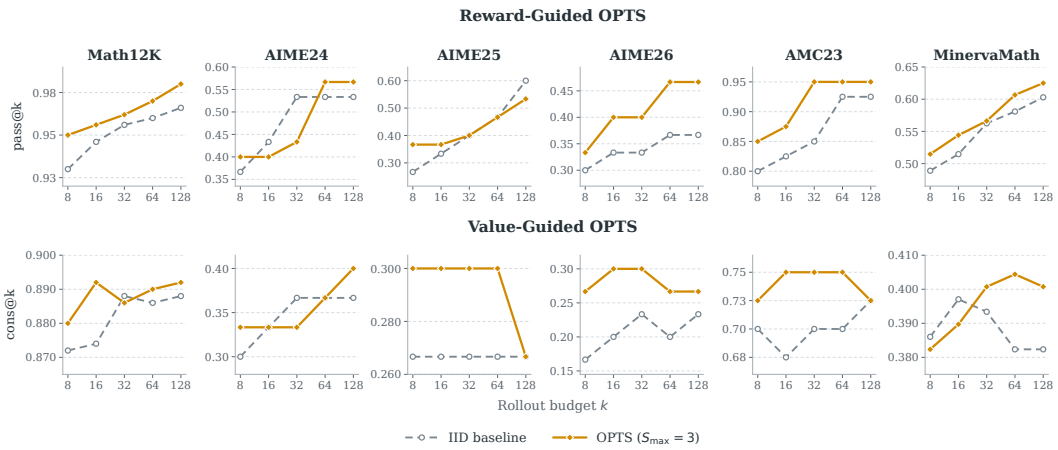


Figure A1: Rollout scaling of OPTS across individual reasoning benchmarks. Top: reward-guided OPTS versus i.i.d. `pass@k`. Bottom: value-guided OPTS versus i.i.d. self-consistency. Each column reports one benchmark under matched rollout budgets $k \in \{8, 16, 32, 64, 128\}$ at fixed $S_{\max} = 3$.

F Search-Budget Improvement of OPTS

Theorem 2 assumes a fixed exact value function; this experiment tests the trend with a learned critic. We fix the step-400 `Qwen3-1.7B` OPTS-TTPO policy and critic, set $\lambda = 0.999$, and vary $S_{\max} \in \{0, 1, 3, 7, 15\}$ with 32 trees per prompt on 902 held-out prompts. Reward- and value-guided search reuse the same root-response texts. We report the greedy-path verifier λ-return J_λ and selected-terminal return J, each relative to $S_{\max} = 0$. Figure A2 gives all four quantities by benchmark. Reward-guided curves increase throughout; value-guided J_λ peaks before $S_{\max} = 15$ on `AIME25` and `MinervaMath`, so the exact-value guarantee need not hold with an approximate critic. Micro averages nevertheless increase over all five budgets, by $(0.0706, 0.1190)$ for reward guidance and $(0.0107, 0.0254)$ for value guidance in (J_λ, J). The reward-guided gain also depends on rebranching position (Appendix G).

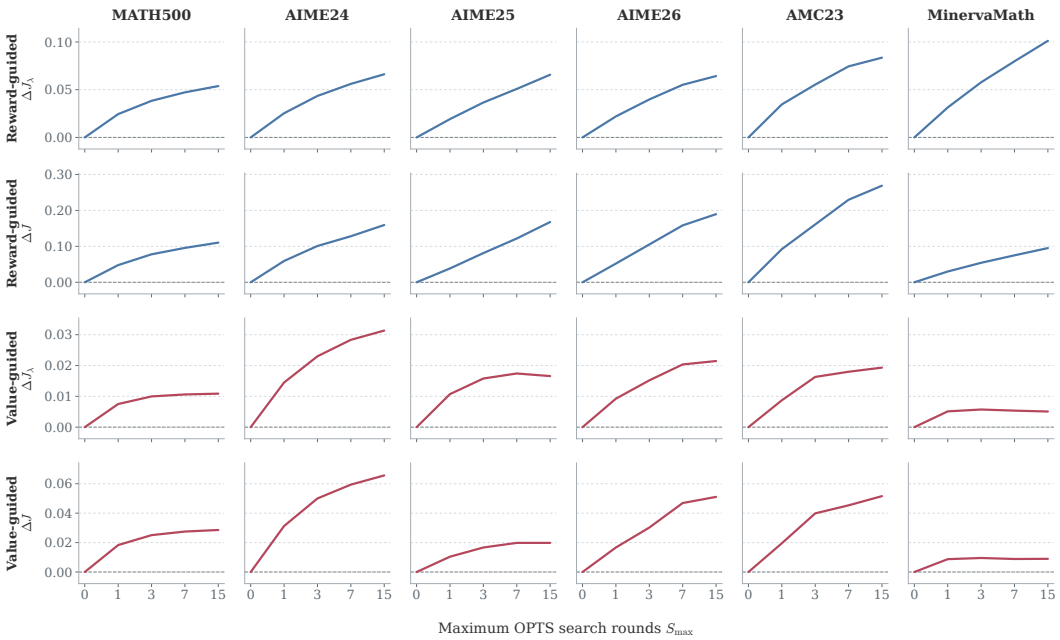


Figure A2: Learned-critic search-budget results by benchmark. Rows report reward-guided ΔJ_λ, reward-guided ΔJ, value-guided ΔJ_λ, and value-guided ΔJ, where each difference is relative to $S_{\max} = 0$. Columns show the six reasoning benchmarks.

G Rebranching Position: Performance-Difference versus Random and Midpoint Selection

OPTS always rebranches by the performance-difference estimate, so we test whether the position it picks matters. Using the step-400 `Qwen3-1.7B` OPTS-TTPO policy and critic, 902 held-out prompts, and 32 trees per prompt, we run OPTS under three rebranching rules: performance-difference selection, a uniformly random position on the same greedy path, and the fixed midpoint of that path. The two control rules keep the per-round tree selection of OPTS, so the number of rebranchings per round is matched; only the node position changes. Figure A3 reports the greedy-terminal `Avg@32` as the search budget grows. Performance-difference selection is best on five of the six benchmarks at every budget above zero, and its margin widens with $S_{\max}$; on AIME25 random selection is marginally higher.

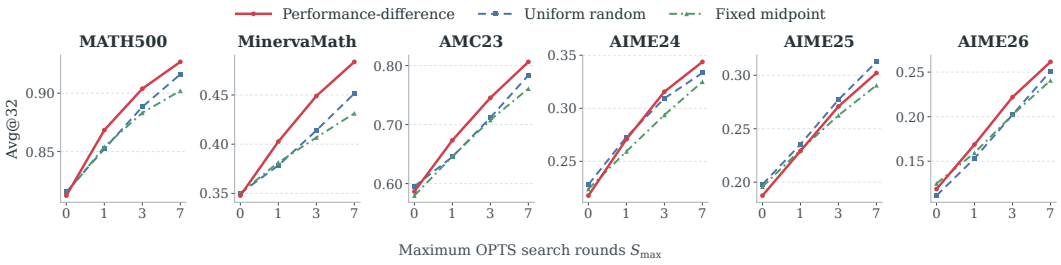


Figure A3: Rebranching position matters. `Avg@32` versus the maximum number of OPTS search rounds for performance-difference, uniformly random, and fixed-midpoint rebranching on the same trees and budgets.

H MuJoCo Hyper-Parameter Grid: Length-Penalty Exponent and Search Count

We sweep $\xi \in \{0, 0.1, \ldots, 1.0\}$ and $s \in \{1, \ldots, 8\}$ on Hopper-v4 and Humanoid-v4, the smallest- and largest-action-space tasks in the five-task suite. The remaining OPTS-TTPO settings are unchanged: 1M steps, 10 seeds per cell, 88 cells per task, and max-backup TreeGAE. Figure A4 reports full-training and tail returns plus a cross-task summary that min–max normalizes each grid before averaging. The main-text setting $(\xi, s) = (0.6, 1)$ (black outline) is best under both metrics and is fixed for the remaining tasks; the heatmaps show the complete grid.

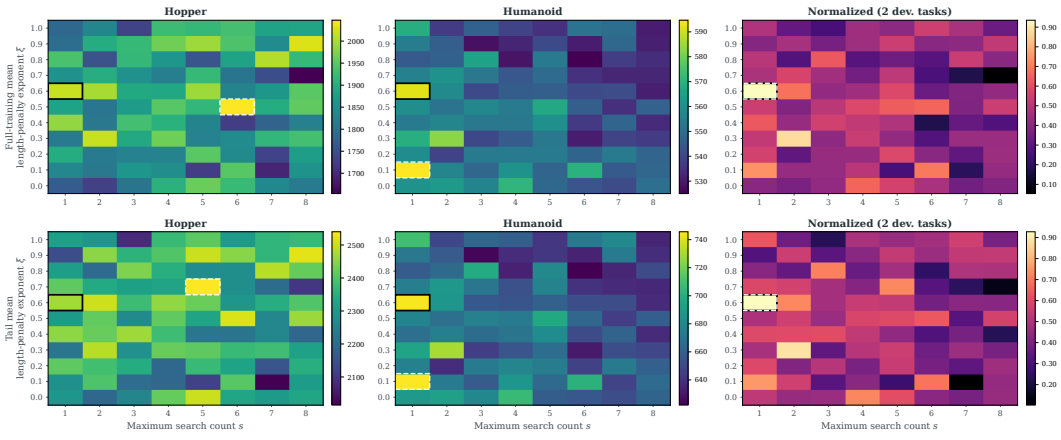


Figure A4: $\xi \times s$ grid on the MuJoCo development tasks. Top: full-training mean return; bottom: tail return. Black outline: the selected configuration $(0.6, 1)$; white dashed outline: the best cell of each panel. The right-most column averages the min–max-normalized grids of Hopper-v4 and Humanoid-v4.

I Full Atari-57 Learning Curves

Each panel corresponds to one Atari game, the horizontal axis is the number of environment interaction steps, and the vertical axis is the raw mean return for that game. Both methods are trained under the sticky-action protocol, and OPTS-TTPO uses mean-backup TreeGAE, the stochastic-environment setting, with $\xi = 0.6$ and $S_{\max} = 1$ as in the main text. Different random seeds of the same algorithm are shown as thin curves with the same color, with smoothing used only for visualization.

Because reward scales differ substantially across Atari games, Figure A5 is intended for within-task comparison and for inspecting where gains or regressions occur. The aggregate cross-task conclusion is therefore based on the human-normalized IQM and task-level win counts reported in the main text rather than on visually aggregating these raw-return curves.



Figure A5: Full Atari-57 learning curves for PPO and OPTS-TTPO. Each panel shows one game; curves are smoothed raw mean returns for different random seeds.

J BACKUP-RULE COMPARISON ON STICKY-ACTION ATARI

We compare mean- and max-backup OPTS-TTPO at the same $(\xi, S_{\max}) = (0.6, 1)$ under the main-text sticky-action protocol: 57 games, three seeds, and ten million interaction steps per run. For each algorithm, we first average the three seed curves pointwise, then average all logged points or the final 100 points. Win/loss/tie counts compare these game-level means using an absolute tie tolerance of 10^{-12}; the per-game tables show the same values rounded to two decimals.

Table A3: Backup-rule comparisons at fixed hyperparameters.

Metric	Comparison	Wins	Losses	Ties
Full training	Max vs. Mean	19	22	16
Full training	Max vs. PPO	27	30	0
Full training	Mean vs. PPO	**31**	26	0
Tail	Max vs. Mean	18	22	17
Tail	Max vs. PPO	**34**	22	1
Tail	Mean vs. PPO	**34**	22	1

Mean backup has a slight edge in game-level wins against max backup, while both beat PPO on 34 games under the tail metric. These results do not establish universal superiority of either backup rule or isolate environmental stochasticity as a causal factor.

Table A4: Per-game seed-mean returns (1), shown to two decimals. Full: all logged points; Tail: final 100 points. Win counts use unrounded values.

	Full			Tail		
Game	PPO	Mean	Max	PPO	Mean	Max
ALE_Surround-v5	-2.83	-2.94	-2.94	-1.14	-0.91	-0.91
AlienNoFrameskip-v4	989.13	947.58	945.15	1186.07	1236.76	1372.65
AmidarNoFrameskip-v4	173.34	192.09	191.86	246.73	301.20	307.77
AssaultNoFrameskip-v4	922.34	967.38	968.90	1073.07	1278.37	1208.08
AsterixNoFrameskip-v4	1553.26	1671.41	1690.02	2051.14	2223.03	2194.67
AsteroidsNoFrameskip-v4	1031.89	1045.94	1042.91	1077.44	1081.97	1105.55
AtlantisNoFrameskip-v4	30513.96	30637.19	28223.74	40791.67	32212.33	27254.17
BankHeistNoFrameskip-v4	310.09	288.34	221.75	430.11	437.08	406.26
BattleZoneNoFrameskip-v4	8426.72	9203.13	9694.18	10452.50	10219.44	11091.39
BeamRiderNoFrameskip-v4	487.94	451.52	451.52	704.01	663.40	663.40
BerzerkNoFrameskip-v4	688.99	612.03	618.03	836.57	714.46	735.34
BowlingNoFrameskip-v4	14.40	12.27	12.39	18.25	16.85	17.46
BoxingNoFrameskip-v4	33.90	35.09	35.09	51.22	54.24	54.24
BreakoutNoFrameskip-v4	55.06	58.26	54.77	120.19	125.61	115.80
CentipedeNoFrameskip-v4	2818.38	2761.23	2806.74	3237.38	3071.08	3212.12
ChopperCommandNoFrameskip-v4	2229.34	2249.77	2135.29	3819.40	3684.47	3281.21
CrazyClimberNoFrameskip-v4	21412.02	20278.52	20278.52	23151.00	23777.56	23777.56
DefenderNoFrameskip-v4	8376.84	8961.33	8783.60	10542.53	12488.08	12195.61
DemonAttackNoFrameskip-v4	1276.39	1302.78	1314.93	2044.33	1925.20	2188.04
DoubleDunkNoFrameskip-v4	-0.81	-0.70	-0.70	-0.05	-0.06	-0.06
EnduroNoFrameskip-v4	26.72	22.69	22.69	40.72	33.14	33.14
FishingDerbyNoFrameskip-v4	-16.08	-12.51	-12.51	5.97	14.45	14.45
FreewayNoFrameskip-v4	3.52	3.24	3.24	5.96	5.88	5.88
FrostbiteNoFrameskip-v4	231.70	231.77	237.73	250.58	249.40	255.50
GopherNoFrameskip-v4	1122.37	1075.35	904.68	1912.06	1894.34	1332.87
GravitarNoFrameskip-v4	397.21	376.87	336.78	642.40	617.09	518.12
HeroNoFrameskip-v4	8702.10	8840.72	9149.41	11663.23	12233.24	12438.86
IceHockeyNoFrameskip-v4	-1.54	-1.60	-1.58	-1.38	-1.28	-1.19
JamesbondNoFrameskip-v4	223.91	229.30	234.10	248.14	267.07	263.86

Table A5: Per-game seed-mean returns (2), shown to two decimals. Full: all logged points; Tail: final 100 points. Win counts use unrounded values.

	Full			Tail		
Game	PPO	Mean	Max	PPO	Mean	Max
KangarooNoFrameskip-v4	1229.65	1612.89	1587.39	1802.94	2458.78	2444.39
KrullNoFrameskip-v4	3762.75	3802.40	3776.63	4173.87	4482.65	4402.41
KungFuMasterNoFrameskip-v4	6102.91	5726.26	5726.26	5609.22	6604.17	6604.17
MontezumaRevengeNoFrameskip-v4	0.21	0.06	0.48	0.00	0.17	0.00
MsPacmanNoFrameskip-v4	1185.86	1131.68	1184.35	1471.88	1464.83	1551.24
NameThisGameNoFrameskip-v4	1160.37	1182.86	1182.86	1337.90	1352.00	1352.00
PhoenixNoFrameskip-v4	2744.98	2826.81	2768.06	3813.27	3939.31	3836.27
PitfallNoFrameskip-v4	-3.08	-3.89	-4.91	0.00	0.00	-0.48
PongNoFrameskip-v4	1.54	0.58	0.58	4.61	3.50	3.50
PrivateEyeNoFrameskip-v4	2.80	6.12	6.11	4.67	12.20	12.20
QbertNoFrameskip-v4	4894.80	5279.52	4903.30	6714.89	7282.50	7130.39
RiverraidNoFrameskip-v4	4081.11	3991.97	4060.80	4840.25	5520.78	4899.51
RoadRunnerNoFrameskip-v4	17287.81	8471.32	20939.21	28086.25	9731.23	33341.33
RobotankNoFrameskip-v4	2.23	2.02	2.02	3.56	2.88	2.88
SeaquestNoFrameskip-v4	498.59	526.78	364.72	582.07	709.10	399.03
SkiingNoFrameskip-v4	-6673.30	-4877.35	-4877.35	-6523.64	-4600.00	-4600.00
SolarisNoFrameskip-v4	304.76	308.64	308.64	293.27	295.07	295.07
SpaceInvadersNoFrameskip-v4	442.97	430.26	424.46	606.18	568.85	496.05
StarGunnerNoFrameskip-v4	7459.37	7351.37	7403.09	11215.28	11361.83	11826.67
TennisNoFrameskip-v4	-1.17	-1.73	-1.73	-0.14	-0.15	-0.15
TimePilotNoFrameskip-v4	2206.19	1999.81	2059.30	2904.39	2557.28	2662.74
TutankhamNoFrameskip-v4	46.56	48.76	44.14	37.73	40.55	48.30
UpNDownNoFrameskip-v4	6777.72	7977.87	6655.13	9677.93	11131.87	8754.43
VentureNoFrameskip-v4	1.71	3.62	3.62	0.00	16.50	16.50
VideoPinballNoFrameskip-v4	9998.85	9542.93	9625.38	17789.33	15498.07	15813.72
WizardOfWorNoFrameskip-v4	1602.55	1657.13	1644.41	2541.53	2613.94	2636.61
YarsRevengeNoFrameskip-v4	7644.99	10684.72	8326.65	8370.44	15992.55	11214.26
ZaxxonNoFrameskip-v4	1087.67	1727.66	1744.45	2367.19	3139.61	3122.94

K PER-CHECKPOINT LLM EVALUATION CURVES

The four `Qwen3-1.7B` methods are evaluated every 20 steps from 20 to 400 using the final-checkpoint protocol: 32 responses per prompt on the 902-prompt set, scored by the same verifier. OPTS-TTPO uses the main-text configuration with max-backup TreeGAE and $S_{\max} = 3$. It matches critic-free baselines on `avg@32` for most of training and pulls ahead in the second half, especially on AIME24–26; its `pass@32` advantage appears early and persists except on AIME24.

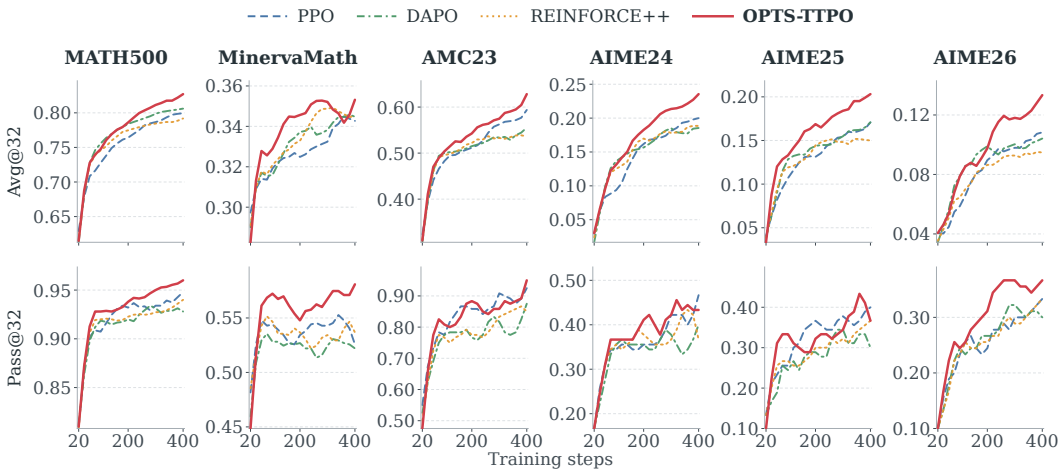


Figure A6: Qwen3-1.7B per-checkpoint evaluation across six reasoning benchmarks. Each column shows one benchmark, with `avg@32` above `pass@32`; all 20 checkpoints of every method are scored with the same verifier. Curves show a three-point moving average that retains the step-20 and step-400 endpoints.

L WALL-CLOCK COMPOSITION OF OPTS-TTPO

Table A6 reports the internal runtime composition recorded in the `Qwen3-1.7B` OPTS-TTPO training logs for Section 4.4. Rollout generation accounts for 43.7% of each step, critic forwards and updates for 34.0%, and actor updates for 16.5%. The explicitly timed path refresh, branch selection, and TreeGAE backup account for 0.7%; beyond these operations, OPTS also makes the rollout rounds serial and performs critic forwards after each round. All timings are per training step, summing repeated operations over all rollout rounds.

Table A6: Wall-clock composition of an OPTS-TTPO step (`Qwen3-1.7B`, $S_{\max} = 3$, eight H200 GPUs).

Stage of one training step	Seconds	Share
Rollout generation (vLLM, all rounds)	348.1	43.7%
Critic update	214.4	26.9%
Actor update	131.4	16.5%
Critic value forwards (all rounds)	56.5	7.1%
Old log-probabilities	39.2	4.9%
Search module: $\widehat{\Delta}$ path refresh + selection	3.2	0.4%
Search module: TreeGAE backup	2.7	0.3%
Reward, final tree processing, other	0.5	0.1%
Total	796.0	100%