FORKPILOT: SELF-EVOLVING POLICY FOR RETROSPECTIVE SEARCH IN LONG-HORIZON AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Interactive language-model agents increasingly solve complex tasks through long-horizon, multi-call reasoning, where errors in beliefs or actions can compound across tool interactions. Retrospective search can recover from such failures but is prone to misallocation. Delayed outcomes obscure the contribution of intermediate search decisions, leading to *Attribution Complexity*, while evolving execution evidence leads to *Adaptation Complexity*, where previously learned estimates become stale. To address these challenges, we first introduce Search Value Dynamics (SVD), which characterizes the evolving trade-off between the gain and cost of retrospective search. Building on SVD, we propose FORKPILOT, a self-evolving two-stage policy-learning framework. In the first stage, FORKPILOT learns a search-value policy offline from completed trajectories through automatically constructed outcome comparisons. In the second stage, it makes search decisions on the current observations and then self-evolves, incorporating newly completed trajectories into subsequent policy updates. We evaluate FORKPILOT across 6 diverse benchmarks and 7 widely used LLM backbone families, including four open-source families, GPT-5.6 Sol and Opus 4.8 in a production agentic system, against 9 competitive baselines, including real-world harness deployment used by hundreds of thousands of paid users. FORKPILOT achieves comparable state-of-the-art performance while reducing token usage by up to 59.2%, demonstrating its efficacy.

1 INTRODUCTION

Interactive language-model agents solve long-horizon tasks through tool calls, acting on evolving observations that only partially reveal the environment state (Liu et al., 2024; Zhou et al., 2024b). Failures can stem from incorrect beliefs about this state or poor actions despite accurate beliefs (Tang et al., 2026; Djuhera et al., 2026). Prior work has studied information gathering through tool use (Yao et al., 2023b), reflection on task feedback (Shinn et al., 2023), and correction with external verification (Gou et al., 2024). Retrospective test-time search examines failed or partially successful trajectories alongside grader or verifier feedback (Zhou et al., 2024a; Yu et al., 2025). Actions and observations in the trajectory, together with feedback on task completion, help the agent revisit beliefs and decisions, seek further observations, and explore alternatives. When earlier states can be restored, search can fork from a state, preserving the valid action prefix while trying another continuation (Koh et al., 2024). In Figure 1, an archive access failure prompts the agent to review its trajectory and inspect key bindings. Finding that the archive still uses the old key, the agent forks from an earlier state to migrate the archive and verify the bindings before disabling the old key.

Allocating retrospective search is difficult in long-horizon tool use: observations are partial, and rewards may arrive only when execution ends (Zhou et al., 2024b; Tan et al., 2026). Searching before enough evidence accumulates may waste forks (Holt et al., 2026), while waiting too long leaves less budget to recover. Different action prefixes also produce different observations and candidate actions, so another trajectory may require fewer corrections than one searched repeatedly (Zhou et al., 2024a; Koh et al., 2024; Yu et al., 2025). Even an improving fork does not establish that further search on the same trajectory is worth its cost (Snell et al., 2025). Its effects may emerge only after later tool calls, making its contribution difficult to separate from subsequent actions (Feng et al., 2025). New observations can reveal recovery actions or make further search unpromising, requiring repeated decisions about when to retrospect, which trajectory to explore, and how much to search (Yao et al.,

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.

2023b; Zhou et al., 2024a). Live environments may also change independently of the agent, making learned allocation policies stale (Pan et al., 2024; Xue et al., 2025).

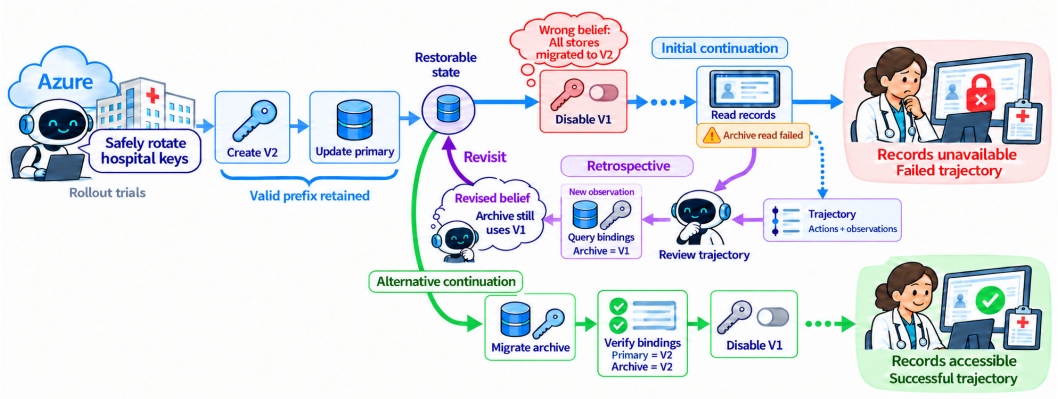


Figure 1: Retrospective search in long-horizon interaction. After an archive access failure, retrospection prompts the agent to query key bindings and revise its belief that migration is complete. The agent then explores an alternative continuation from an earlier state, preserving the valid prefix.

Existing work addresses computation allocation through input-adaptive budgets, confidence-based stopping, and complexity-based token allocation (Damani et al., 2025; Fu et al., 2025; Lin et al., 2025), while learned planning policies decide when to deliberate during interaction (Paglieri et al., 2025). Complementarily, experience-based methods such as Reflexion (Shinn et al., 2023), ReasoningBank (Ouyang et al., 2026), and ACE (Zhang et al., 2026) leverage previous trajectories and feedback to improve reasoning and guide exploration through evolving reflections, memory, and context. However, retrospective search in long-horizon, multi-call reasoning requires more than adjusting reasoning budgets or improving action selection through accumulated experience. Delayed, entangled outcomes obscure the incremental benefit of additional search and whether it justifies its computational cost, creating **Attribution Complexity**, while evolving tool observations and changes in the environment can render previously learned search-value estimates stale, creating **Adaptation Complexity**. These challenges motivate two research questions: **Q1:** *How can we theoretically understand the value of retrospective search under delayed rewards and evolving observations in long-horizon, multi-turn tool interaction?* **Q2:** *Can models self-evolve through accumulated experience to dynamically allocate inference resources, balancing effectiveness and efficiency?*

To answer these questions, we first introduce **Search Value Dynamics** (SVD), which characterizes the evolving trade-off between the gain and cost of retrospective search. Building on SVD, we propose **ForkPilot**, a self-evolving two-stage policy-learning framework. ForkPilot can bootstrap its search-value policy from completed trajectories using automatically constructed outcome comparisons. During deployment, it cit makes search decisions on the current observations and then self-evolves, incorporating newly completed trajectories into subsequent policy updates. This self-evolving refinement provides a mechanism for adapting search allocation to live real-world environments that evolve independently of the agent's actions.

We evaluate ForkPilot across 6 diverse benchmarks and 7 widely used LLM backbone families. We also deploy ForkPilot in a production agentic system, where the agent uses GPT-5.6 Sol and Claude Opus 4.8 served through production inference platforms employing state-of-the-art caching. We compare against 9 competitive baselines, including a real-world harness deployment used by hundreds of thousands of paid users. ForkPilot achieves comparable state-of-the-art performance while reducing token usage by up to 59.2%, demonstrating its efficacy.

2 Preliminaries

2.1 Challenges in Long-horizon Retrospective Search

Given task x and computation budget b, a retrospective-search host restores an earlier environment state and forks a new branch with the black-box agent policy π to sample alternative action sequences, preserving the action-observation prefix up to that state. At the fork, π conditions on the

preserved prefix including x, previously completed trajectory, and permitted feedback available at the retrospective decision (Zhou et al., 2024a; Yu et al., 2025).

At decision t, the search allocation controller observes $z_t = \phi(x, e_{1:\ell_t}, \mathcal{T}_t, b_t)$, which summarizes observed action-observation events, explored branches and their available outcomes, and remaining budget. The controller cannot observe the full environment state or future events. From the feasible menu $\mathcal{U}_t$, it chooses an allocation u, such as fork count and rollout depth. u^0 signifies that the rollout execution continues without any forking from the current state.

We define the value of u relative to u^0 using paired replay Γ. Both actions start from the same restored state under coordinated randomness, a common subsequent execution rule, and the same total-budget constraint (Appendix B.2). Let τ_t^u and τ_t^0 denote the completed continuations and their search records, including any later forks and branch rollouts. Then $Y_t(u) = R(x, \tau_t^u) - R(x, \tau_t^0)$, $\Delta C_t(u) = C(\tau_t^u) - C(\tau_t^0)$, $G_t(u) = \mathbb{E}_\Gamma[Y_t(u) - \lambda \Delta C_t(u) \mid z_t]$, $G_t(u^0) = 0$, where $\lambda \geq 0$ and C includes the generation and evaluation of unselected branches. Completed pairs provide learning and evaluation targets; deployment does not require paired replay at every decision.

Attribution Complexity. A completed reward reflects both the current fork and later tool actions. Sampling and environment responses therefore make $Y_t(u)$ variable even at a fixed z_t. We quantify this variability as $\mathcal{A}_t^\Gamma(u) = \mathrm{Var}_\Gamma[Y_t(u) \mid z_t]$. For n conditionally independent paired replays at fixed (z_t, u), the sample-mean variance is $\mathcal{A}_t^\Gamma(u)/n$. Larger attribution complexity thus requires more completed pairs for the same sampling precision.

Adaptation Complexity. New observations and search outcomes change z_t; changes in the environment may also alter reward effects at a fixed input. Let P and Q be the current and source conditional laws of $Y_t(u)$ at the same z_t, defined under a common Γ and subsequent execution rule.

Since $R \in [0,1]$ implies $Y_t(u) \in [-1,1]$, $|\mathbb{E}_P Y_t(u) - \mathbb{E}_Q Y_t(u)| \leq \mathcal{S}_t(u) := 2\,\mathrm{TV}(P,Q) \leq \sqrt{2 D_{\mathrm{KL}}(P\|Q)}$, where the last inequality assumes finite KL divergence and natural logarithms. This bound describes source-to-current shift at a fixed input.

2.2 Problem Definition

The host admits only feasible actions and reserves resources for subsequent execution, enforcing $C(\tau) \leq B$ for each complete task execution τ. We select λ using source validation only: B imposes the hard constraint, while λ governs the reward-cost trade-off among feasible decisions. Self-evolution proceeds across completed source tasks. The controller uses θ_n, learned from $\mathcal{D}_n$, throughout the next task. Only after that task finishes does it incorporate its completed experience ξ_{n+1} and update: $\mathcal{D}_{n+1} = \mathcal{D}_n \cup \{\xi_{n+1}\}, \theta_{n+1} = \mathsf{Update}(\theta_n, \mathcal{D}_{n+1})$. Here, ξ_{n+1} contains the observed search decisions, continuations, outcomes, and costs; the update does not require labels specifying optimal search allocations.

Problem 1 (Efficient Self-evolving Policy on Retrospective Long-horizon Reasoning)**.**
Given. *A policy π, a retrospective-search host, completed experience $\mathcal{D}_n$ from n source tasks, and, at each eligible decision, the observable input z_t and feasible menu $\mathcal{U}_t$.*

Find. *A learning procedure and its induced sequence of causal policies $\{\mu_{\theta_n}\}_{n\geq 0}$, where each policy is fitted from permitted completed source experience and selects $\mu_{\theta_n}(z_t, \mathcal{U}_t) \in \mathcal{U}_t$ using only information available at that decision. At each stage, the policy approximates the constrained objective*

$$\max_{\mu \in \Pi_{\mathrm{causal}}} \quad \mathbb{E}[R(x,\tau) - \lambda C(\tau)] \quad \textit{s.t.} C(\tau) \leq B \quad \textit{a.s.} \tag{1}$$

Here, Π_{causal} is the class of policies that select actions from the current observable input and feasible menu, using parameters fitted only on permitted completed experience.

3 Learning Retrospective Search Allocation

Building on the problem formulation, we next characterize the uncertainty in learning paired effects and the changes incurred when using a source estimate at a later decision. These motivate the two stages of ForkPilot.

3.1 SEARCH-VALUE DYNAMICS

Following classical metareasoning (Russell & Wefald, 1991; Hay et al., 2012), fix a source-task cost scale $\bar{C}_{\mathrm{src}} > 0$. Suppressing decision subscripts when unambiguous, write $D(u) = \Delta C(u)/\bar{C}_{\mathrm{src}}$ and $\lambda_D = \lambda \bar{C}_{\mathrm{src}}$. The paired net effect is $W(u) = Y(u) - \lambda_D D(u) = Y(u) - \lambda \Delta C(u)$. Let z_{src} denote an input represented in completed source experience and z_t the current input. We use Q for the source conditional law of (Y, D) and P for the corresponding current law. Both laws use the same fixed Γ, including ρ; the comparison does not change the continuation rule while changing the input or task law.

Attribution complexity. The first challenge is estimating whether an allocation is worth its incremental cost. At (z_{src}, u), let $A^{\Gamma}_{\mathrm{src}}(u) = \mathrm{Var}_{\mathsf{Q}}[Y(u) \mid z_{\mathrm{src}}]$ and $V^{\Gamma}_{\mathrm{src}}(u) = \mathrm{Var}_{\mathsf{Q}}[W(u) \mid z_{\mathrm{src}}]$. Then $V^{\Gamma}_{\mathrm{src}}(u) = A^{\Gamma}_{\mathrm{src}}(u) + \lambda_D^2 \mathrm{Var}_{\mathsf{Q}}[D(u) \mid z_{\mathrm{src}}] - 2\lambda_D \mathrm{Cov}_{\mathsf{Q}}(Y(u), D(u) \mid z_{\mathrm{src}})$.

Thus both reward and cost variation matter. With n conditionally independent pairs at the same input and action, the variance of their mean net effect is $V^{\Gamma}_{\mathrm{src}}(u)/n$.

The controller representation can add another source of ambiguity. At a source decision, let H be the available history, $Z = \phi(H)$ its representation, and $g(H, u) = \mathbb{E}_{\mathsf{Q}}[W(u) \mid H]$ for an action feasible across the histories considered. Because Z is determined by H,

$$\mathrm{Var}_{\mathsf{Q}}(W(u) \mid Z) = \mathbb{E}_{\mathsf{Q}}[\mathrm{Var}_{\mathsf{Q}}(W(u) \mid H) \mid Z] + \mathrm{Var}_{\mathsf{Q}}(g(H, u) \mid Z). \tag{2}$$

The second term arises when different permitted histories with different search values map to the same input Z. Sampling noise and this representation ambiguity jointly complicate attribution.

Adaptation complexity. A source estimate may become stale when observations, explored branches, or the remaining budget change the controller input. The feasible menu can change as well. These changes motivate recomputing action scores at each eligible decision. They do not, by themselves, show that a newly fitted prediction is more accurate; the following bound only characterizes the discrepancy incurred by reusing a source-input estimate.

We can also bound the discrepancy between a source estimate and the current value. For $\mathsf{L} \in \{\mathsf{P}, \mathsf{Q}\}$, define $m_{\mathsf{L}}(z, u) = \mathbb{E}_{\mathsf{L}}[Y(u) \mid z]$ and $c_{\mathsf{L}}(z, u) = \mathbb{E}_{\mathsf{L}}[D(u) \mid z]$. For an action feasible at both z_{src} and z_t, define

$$\begin{aligned} \Omega_{\mathrm{src}\to t}(u) &= |m_{\mathsf{P}}(z_t, u) - m_{\mathsf{P}}(z_{\mathrm{src}}, u)|, \\ S_{\mathrm{src}}(u) &= 2\,\mathrm{TV}(\mathsf{P}^u_{z_{\mathrm{src}}}, \mathsf{Q}^u_{z_{\mathrm{src}}}), \\ \xi_{\mathrm{src},t}(u) &= |c_{\mathsf{P}}(z_t, u) - c_{\mathsf{Q}}(z_{\mathrm{src}}, u)|, \end{aligned}$$

where P^u_z and Q^u_z are the conditional laws of $Y(u)$ at z. The terms measure input change under the current law, source-to-current reward-law shift at the source input, and the combined change in expected normalized cost. Because $Y(u) \in [-1, 1]$ when $R \in [0, 1]$, $S_{\mathrm{src}}(u) \le \sqrt{2 D_{\mathrm{KL}}(\mathsf{P}^u_{z_{\mathrm{src}}} \| \mathsf{Q}^u_{z_{\mathrm{src}}})}$ whenever this KL divergence is finite.

Theorem 1 (Source-to-current search-value bound). *Let $\mathcal{F}_t = \mathcal{U}_t \setminus \{u^0\}$ contain $K \ge 1$ actions, each feasible at z_{src}. Fix Γ, the source and current laws, z_{src}, z_t, $\mathcal{F}_t$, and the sample counts $n_u \ge 1$ independently of the replay draws used below. Suppose $\mathsf{P}^u_{z_{\mathrm{src}}}$ and $\mathsf{Q}^u_{z_{\mathrm{src}}}$ are well defined for every $u \in \mathcal{F}_t$ (i.e. z_{src} has positive probability or density under the respective marginal law of Z), with the analogous current-law quantities well defined at z_t. For each $u \in \mathcal{F}_t$, draw n_u conditionally i.i.d. source pairs at (z_{src}, u) under Q, with finite $V^{\Gamma}_{\mathrm{src}}(u)$. Let $\overline{W}_{\mathrm{src}}(u)$ be their mean, $\widehat{G}_{\mathrm{src}}(u)$ any source-fitted predictor evaluated at z_{src}, and $e_{\mathrm{src}}(u) = |\widehat{G}_{\mathrm{src}}(u) - \overline{W}_{\mathrm{src}}(u)|$. If rewards lie in $[0, 1]$, then for $0 < \delta < 1$, with probability at least $1 - \delta$ over these source replay draws, simultaneously for all $u \in \mathcal{F}_t$,*

$$\begin{aligned} |G_t(u) - \widehat{G}_{\mathrm{src}}(u)| \le e_{\mathrm{src}}(u) &+ \sqrt{\frac{K V^{\Gamma}_{\mathrm{src}}(u)}{n_u \delta}} \\ &+ \Omega_{\mathrm{src}\to t}(u) + S_{\mathrm{src}}(u) \\ &+ \lambda_D \xi_{\mathrm{src},t}(u) =: \beta^{\delta}_{\mathrm{src},t}(u), \end{aligned} \tag{3}$$

where $G_t(u) = m_{\mathsf{P}}(z_t, u) - \lambda_D c_{\mathsf{P}}(z_t, u)$.

Additional pairs reduce the displayed sampling term. The condition $n_u \ge K V^{\Gamma}_{\mathrm{src}}(u)/(\delta \eta^2)$ is sufficient to make that term at most η; it does not remove input or law changes. Set $\widehat{G}_{\mathrm{src}}(u^0) =$

$\beta^{\delta}_{\mathrm{src},t}(u^0) = 0$ and define $L^{\delta}_t(u) = \widehat{G}_{\mathrm{src}}(u) - \beta^{\delta}_{\mathrm{src},t}(u)$ and $U^{\delta}_t(u) = \widehat{G}_{\mathrm{src}}(u) + \beta^{\delta}_{\mathrm{src},t}(u)$. On the theorem event, if $u_L \in \arg\max_{u\in\mathcal{F}_t} L^{\delta}_t(u)$ has $L^{\delta}_t(u_L) > 0$, then $G_t(u_L) > 0$ and its local net-value regret relative to $u^* \in \arg\max_{u\in\mathcal{U}_t} G_t(u)$ is at most $2\beta^{\delta}_{\mathrm{src},t}(u^*)$. If $\max_{u\in\mathcal{F}_t} U^{\delta}_t(u) \le 0$, skip is optimal. Appendix A gives the proof.

The bound concerns reusing $\widehat{G}_{\mathrm{src}}(u)$ at a fixed current input. Its probability statement is not uniform over inputs or menus chosen adaptively after observing the replay draws. At runtime, the learned predictor is instead evaluated on z_t; neither this bound nor $\beta^{\delta}_{\mathrm{src},t}(u)$ controls that refreshed prediction's error. The exact-input pairs and shift terms are also unavailable during deployment. Thus the theorem diagnoses why attribution and stale estimates matter under a fixed replay rule; it provides no certificate for the learned margin or the deployed controller.

3.2 FORKPILOT: TWO-STAGE SELF-EVOLVING POLICY

FORKPILOT implements these requirements in two stages. Stage I learns search effects and their reliability from paired completed trajectories, which may be collected both during an offline initialization phase and from tasks subsequently completed during deployment. These paired replays follow the same reference rule ρ and form a source pool $\mathcal{D}_n$. As additional deployment tasks are completed, newly available trajectory pairs extend $\mathcal{D}_n$ to $\mathcal{D}_{n'}$, where $n' > n$, and Stage I refits the controller parameters $\theta_{n'}$ on the expanded pool before a subsequent evaluation stage. Stage II operates within each evaluated task: the fitted controller parameters remain fixed, while search allocations are repeatedly recomputed as new trajectory evidence becomes available. Thus, Stage I enables the policy to evolve across completed tasks, whereas Stage II adapts its search decisions within a task without online parameter updates.

Stage I: Attribution from completed experience. At an eligible source state, paired replay compares u with u^0 from the same restorable state. The completed executions yield $\widetilde{Y} = R(x,\tau^u) - R(x,\tau^0)$ and $\widetilde{D} = [C(\tau^u) - C(\tau^0)]/\bar{C}_{\mathrm{src}}$. Each pair labels its selected action only. Training needs neither human allocation labels nor paired replay at every deployment decision.

An encoder produces $r_t = f_\psi(z_t)$. An action-conditioned ensemble of M task-bootstrap heads predicts reward effects $\widehat{m}^{(j)}_t(u)$ and normalized cost effects $\widehat{c}^{(j)}_t(u)$. Averaging across heads gives $\widehat{G}_t(u) = \widehat{m}_t(u) - \lambda_D \widehat{c}_t(u)$, with the fitted ensemble evaluated on the current z_t at deployment. Because the theorem's shift terms are unavailable at runtime, FORKPILOT estimates an empirical margin $\widehat{\beta}_t(u)$ using ensemble disagreement, source-representation support, and held-out paired residuals. A margin model predicts an upper quantile of $|\widetilde{Y} - \lambda_D \widetilde{D} - \widehat{G}_t(u)|$, with a correction calibrated on separate source data (Appendix B). We set $L_t(u) = \widehat{G}_t(u) - \widehat{\beta}_t(u)$ and $U_t(u) = \widehat{G}_t(u) + \widehat{\beta}_t(u)$. These are decision scores, not conditional-coverage certificates or estimates of the theorem's unavailable radius.

Stage II: Closed-loop adaptation. With the fitted artifact $\theta_{n'}$ fixed, the controller scores the current feasible menu using z_t. Let $\mathcal{F}_t = \mathcal{U}_t \setminus \{u^0\}$, and let $k(u)$ and $d(u)$ denote an allocation's fork count and rollout-depth limit. Fix probe-specific caps k_{pr} and d_{pr} before evaluation, stricter than the general host limits, and set $\mathcal{P}_t = \{u \in \mathcal{F}_t : k(u) \le k_{\mathrm{pr}},\ d(u) \le d_{\mathrm{pr}}\}$ (Appendix B). Define $L^{\max}_t = \max_{u\in\mathcal{F}_t} L_t(u)$ and $U^{\mathrm{probe}}_t = \max_{u\in\mathcal{P}_t} U_t(u)$, taking a maximum over an empty set to be $-\infty$. The controller chooses

$$u_t = \begin{cases} \arg\max_{u\in\mathcal{F}_t} L_t(u), & L^{\max}_t > 0, \\ \arg\min_{\substack{u\in\mathcal{P}_t \\ U_t(u)>0}} \widehat{c}_t(u), & L^{\max}_t \le 0 < U^{\mathrm{probe}}_t, \\ u^0, & \text{otherwise.} \end{cases} \tag{4}$$

Thus FORKPILOT selects a search action with positive lower score, chooses a capped probe with the lowest predicted normalized full-execution cost difference when an upper score is positive, and otherwise skips the current round.

4 EXPERIMENTS

We comprehensively evaluate FORKPILOT against 9 competitive baselines across 6 diverse benchmarks on 7 backbone families. Our experiments answer the following four questions: ***Q1:*** *Does* FORKPILOT *consistently achieve efficiency across diverse benchmarks?* ***Q2:*** *Does* FORKPILOT *consistently consistently achieve efficiency across different backbone model scales?* ***Q3:*** *Does* FORKPILOT *retain its efficiency gains on frontier black-box models?* ***Q4:*** *Does* FORKPILOT *require privileged feedback or large bootstrap samples?*

4.1 EXPERIMENTAL SETTINGS

Baselines. We compare against 9 competitive baselines, including three families of methods: (1) *test-time search*, including BoN-5 (Snell et al., 2025), full Retrospective Tree Search (RTS) and Retrospective Depth Search (RDS), where RTS uses a source-validation-selected heuristic to explore alternative continuations from earlier states for strong task reward and RDS reviews one initial rollout and continues only from its final resumable node; (2) *experience-based adaptation*, including ACE (Zhang et al., 2026), ReasoningBank (Ouyang et al., 2026), Reflexion (Shinn et al., 2023), GEPA (Agrawal et al., 2026), and Trace2Skill (Ni et al., 2026); and (3) *state-of-the-art agent harnesses*, represented by GitHub Copilot (GHCP) [1], which repeatedly conditions action selection on accumulated interaction history and tool results without explicit retrospective trajectory search.

Table 1: Efficiency comparison across 6 diverse benchmarks. S is task score (0-100), Higher S is better; R/U are raw/uncached token ratios to RTS. Lower R/U is better. The best is in **Bold** and the second best is in underline.

Method	Automation			τ^3-Bench			SWE-bench			MM-ToolSandbox			WorkArena++			VitaBench		
	S	R	U	S	R	U	S	R	U	S	R	U	S	R	U	S	R	U
DeepSeek-V4-Flash																		
BoN-5	49.5	1.17	1.26	74.1	1.18	1.24	72.0	1.10	1.22	28.2	1.07	1.26	49.4	1.08	1.29	28.6	1.08	1.29
RTS	56.8	1.00	1.00	82.1	1.00	1.00	78.8	1.00	1.00	35.2	1.00	1.00	52.1	1.00	1.00	**32.6**	1.00	1.00
ACE	55.0	0.62	0.66	81.2	0.69	0.74	76.7	0.64	0.72	33.1	0.65	0.73	51.9	**0.61**	**0.67**	31.5	0.60	0.67
ReasoningBank	55.0	0.61	0.67	**82.6**	**0.64**	0.73	78.6	**0.63**	**0.70**	35.4	0.65	0.74	51.6	0.62	0.70	32.0	0.59	**0.63**
Reflexion	53.8	0.72	0.79	78.0	0.76	0.87	76.4	0.84	0.89	32.2	0.79	0.88	49.9	0.74	0.80	31.1	0.81	0.92
GEPA	57.0	0.59	0.65	80.6	0.70	0.77	**79.5**	0.68	0.72	**35.7**	0.70	0.80	50.5	0.64	0.70	31.5	**0.58**	0.64
Trace2Skill	55.8	0.63	0.71	82.4	0.66	0.72	79.4	0.68	0.74	34.4	**0.63**	**0.71**	51.3	0.66	0.71	32.1	0.65	0.70
FORKPILOT	**57.2**	**0.56**	**0.64**	82.3	0.65	**0.69**	78.7	0.69	0.73	**35.7**	0.66	**0.71**	**52.4**	0.64	0.68	32.4	0.65	0.71
Kimi-K3																		
BoN-5	54.0	1.10	1.19	82.5	1.16	1.25	76.6	1.13	1.21	37.2	1.14	1.24	51.5	1.15	1.23	28.8	1.19	1.30
RTS	**64.5**	1.00	1.00	86.3	1.00	1.00	81.4	1.00	1.00	**43.4**	1.00	1.00	59.5	1.00	1.00	36.3	1.00	1.00
ACE	61.2	0.59	0.71	86.5	0.67	**0.67**	78.0	**0.62**	**0.73**	41.1	0.66	0.79	59.8	0.56	0.70	34.0	0.62	0.76
ReasoningBank	60.3	0.60	0.73	85.5	0.56	0.70	81.0	0.68	0.77	40.3	0.65	0.78	57.2	0.62	0.75	33.5	0.63	0.77
Reflexion	58.0	0.78	0.85	83.2	0.76	0.84	78.8	0.83	0.89	39.1	0.79	0.88	54.8	0.81	0.89	31.5	0.84	0.93
GEPA	62.8	0.62	0.72	87.0	0.55	**0.67**	80.0	0.67	0.78	41.5	0.64	0.75	59.0	0.58	0.69	34.8	0.62	0.73
Trace2Skill	62.3	**0.57**	0.71	85.9	0.58	0.70	79.6	0.64	0.77	42.6	**0.63**	0.76	**60.5**	**0.54**	**0.67**	35.5	**0.60**	**0.72**
FORKPILOT	64.2	0.63	**0.67**	**87.2**	**0.53**	0.72	**81.6**	0.70	0.74	43.0	0.68	**0.74**	59.8	0.64	0.70	**36.5**	0.66	**0.72**
DeepSeek-R1-671B																		
BoN-5	34.1	1.14	1.22	62.8	1.16	1.22	55.5	1.12	1.21	17.4	1.09	1.23	34.4	1.15	1.27	12.9	1.08	1.29
RTS	37.6	1.00	1.00	66.2	1.00	1.00	62.5	1.00	1.00	20.9	1.00	1.00	40.3	1.00	1.00	18.2	1.00	1.00
ACE	35.6	0.69	0.77	66.2	0.64	0.75	60.8	0.69	0.79	19.9	0.68	0.75	40.0	0.63	0.72	**18.4**	0.68	0.76
ReasoningBank	36.4	0.67	0.74	65.3	0.70	0.75	61.4	0.74	0.80	19.2	0.73	0.78	39.0	0.65	0.72	16.9	0.69	0.75
Reflexion	33.2	0.77	0.82	62.8	0.81	0.90	61.2	0.81	0.88	16.4	0.81	0.87	37.4	0.76	0.83	15.7	0.80	0.89
GEPA	36.4	**0.64**	0.73	64.6	0.67	0.72	62.5	0.67	0.77	**21.5**	**0.63**	**0.73**	39.2	0.67	0.77	17.9	**0.64**	**0.68**
Trace2Skill	36.7	0.71	0.80	66.4	**0.60**	**0.66**	**62.8**	**0.65**	**0.72**	19.8	0.68	0.77	40.5	**0.59**	**0.65**	18.3	0.73	0.79
FORKPILOT	**38.0**	**0.64**	**0.70**	**66.5**	0.66	0.72	62.2	0.69	0.73	21.3	0.68	**0.73**	**40.7**	0.65	0.71	**18.4**	0.66	0.71
Qwen3.8-27B																		
BoN-5	18.0	1.07	1.16	66.4	1.13	1.22	49.2	1.10	1.18	15.9	1.11	1.21	29.0	1.12	1.20	9.3	1.16	1.27
RTS	30.5	1.00	1.00	73.8	1.00	1.00	57.6	1.00	1.00	23.3	1.00	1.00	38.7	1.00	1.00	**17.5**	1.00	1.00
ACE	27.8	0.56	0.68	74.2	**0.50**	**0.64**	53.8	**0.59**	**0.70**	20.5	0.63	0.76	39.0	0.53	0.67	15.5	0.59	0.73
ReasoningBank	26.8	0.57	0.70	73.0	0.53	0.67	57.0	0.65	0.74	19.8	0.62	0.75	36.5	0.59	0.72	14.5	0.60	0.74
Reflexion	23.5	0.75	0.82	68.9	0.73	0.81	52.6	0.80	0.86	17.8	0.76	0.85	33.2	0.78	0.86	12.5	0.81	0.90
GEPA	29.0	0.59	0.69	**75.1**	0.52	**0.64**	56.0	0.64	0.75	21.3	0.61	0.72	38.3	0.55	0.66	16.5	0.59	0.70
Trace2Skill	28.2	**0.54**	0.68	73.4	0.55	0.67	55.2	0.61	0.74	22.1	**0.60**	0.73	39.1	**0.51**	**0.64**	17.0	**0.57**	0.69
FORKPILOT	**31.0**	0.58	**0.63**	74.4	0.61	0.66	**58.0**	0.65	**0.70**	**23.6**	**0.60**	**0.66**	**39.3**	0.59	0.65	17.3	0.62	**0.68**

[1] https://code.visualstudio.com/blogs/2026/05/15/agent-harnesses-github-copilot-vscode

Backbones. We evaluate on 7 backbone families, including both (1) *open-weight models*, including DeepSeek-V4-Flash (DeepSeek-AI, 2026), Kimi-K3 (Moonshot AI, 2026), DeepSeek-R1-671B (DeepSeek-AI, 2025), Qwen3.5(4B, 9B, 28B, 122B) (Qwen Team, 2026d) and Qwen3.8-27B (Qwen Team, 2026e) and (2) *black-box models*, including GPT-5.6 Sol[2] and Opus 4.8[3] with **extra-high** reasoning effort.

Benchmarks. We evaluate on 6 diverse benchmarks, covering workflow automation (AutomationBench (Zapier, 2026)), tool-user interaction (τ^3-Bench (Tau-Bench Contributors, 2026)), software engineering (SWE-bench Verified (SWE-bench)), multimodal tool use (MM-ToolSandbox (Apple, 2026)), web interaction (WorkArena++ (ServiceNow, 2026)) and general interactive tasks (VitaBench (Meituan LongCat, 2026)). Additional details are in Appendix B-C.

Table 2: Efficiency comparison across different backbone scales on six benchmarks. S is task score (0-100); R/U are raw/uncached token ratios to RTS. Lower R/U is better. The best is in **Bold** and the second best is in underline.

Method	Automation			τ^3-Bench			SWE-bench			MM-ToolSandbox			WorkArena++			VitaBench		
	S	R	U	S	R	U	S	R	U	S	R	U	S	R	U	S	R	U
Qwen3.5-4B																		
BoN-5	9.5	1.13	1.22	41.2	1.07	1.28	18.0	1.18	1.28	6.0	1.10	1.28	13.1	1.12	1.23	6.5	1.17	1.22
RTS	14.1	1.00	1.00	43.6	1.00	1.00	24.8	1.00	1.00	12.9	1.00	1.00	20.3	1.00	1.00	8.8	1.00	1.00
ACE	13.5	0.71	0.79	42.2	0.74	0.83	24.9	0.77	0.83	11.8	0.65	0.72	19.2	0.71	0.81	6.8	0.77	0.85
ReasoningBank	12.5	**0.69**	0.78	43.9	0.74	0.83	23.2	0.74	0.83	10.6	0.67	0.78	19.8	0.77	0.87	8.5	0.72	0.78
Reflexion	11.7	0.80	0.85	39.1	0.85	0.92	20.3	0.85	0.93	8.8	0.82	0.86	16.6	0.79	0.85	4.8	0.85	0.91
GEPA	12.9	0.70	0.77	**44.4**	**0.68**	**0.72**	**25.1**	**0.68**	**0.76**	11.3	0.65	0.75	**20.7**	0.67	**0.72**	8.2	0.77	0.85
Trace2Skill	13.1	0.72	0.77	44.0	0.77	0.82	24.1	0.69	**0.76**	**13.4**	**0.64**	**0.69**	19.8	**0.65**	0.73	**9.3**	**0.68**	**0.77**
FORKPILOT	**14.5**	0.70	**0.75**	44.2	0.72	0.76	25.0	0.74	0.79	13.2	0.69	0.74	20.6	0.71	0.76	8.6	0.73	0.78
Qwen3.5-9B																		
BoN-5	17.6	1.06	1.22	62.4	1.18	1.30	40.3	1.13	1.25	12.4	1.13	1.28	28.1	1.16	1.23	7.8	1.12	1.23
RTS	25.2	1.00	1.00	64.8	1.00	1.00	48.2	1.00	1.00	19.5	1.00	1.00	32.6	1.00	1.00	14.1	1.00	1.00
ACE	23.8	0.63	**0.68**	63.6	0.73	0.83	46.0	0.76	0.82	**19.7**	0.62	0.68	31.9	**0.64**	**0.72**	13.6	**0.63**	**0.69**
ReasoningBank	25.4	0.69	0.73	64.5	0.71	0.75	46.5	**0.62**	**0.68**	16.7	0.69	0.77	31.7	0.72	0.81	13.2	0.73	0.77
Reflexion	24.2	0.77	0.84	62.1	0.77	0.84	43.9	0.80	0.88	16.6	0.79	0.87	31.5	0.82	0.88	9.5	0.78	0.88
GEPA	24.4	0.70	0.79	**65.3**	0.69	0.78	**48.9**	0.63	0.71	18.5	**0.58**	**0.62**	31.2	0.69	0.78	13.2	0.73	0.82
Trace2Skill	24.5	**0.61**	**0.68**	63.5	**0.61**	**0.69**	47.7	0.71	0.78	19.0	0.64	0.68	32.9	0.69	0.78	13.3	0.70	0.79
FORKPILOT	**25.6**	0.65	0.70	65.0	0.66	0.72	48.5	0.69	0.73	19.4	0.64	0.69	**33.2**	0.67	**0.72**	**14.3**	0.67	0.71
Qwen3.5-27B																		
BoN-5	23.5	1.07	1.28	63.2	1.18	1.26	48.3	1.13	1.29	18.9	1.18	1.22	33.9	1.08	1.28	11.6	1.07	1.28
RTS	28.7	1.00	1.00	70.6	1.00	1.00	53.8	1.00	1.00	22.1	1.00	1.00	35.9	1.00	1.00	**15.9**	1.00	1.00
ACE	27.8	0.64	0.73	68.5	0.59	0.68	53.0	0.70	0.74	20.4	0.68	0.78	33.7	0.59	**0.64**	14.4	0.66	0.72
ReasoningBank	27.8	0.65	0.75	70.0	0.64	0.70	51.3	**0.64**	0.74	21.2	**0.59**	0.68	33.5	0.67	0.74	14.9	**0.64**	0.75
Reflexion	26.2	0.78	0.89	68.8	0.72	0.82	49.2	0.78	0.88	19.5	0.75	0.82	33.7	0.70	0.77	13.1	0.78	0.83
GEPA	28.8	**0.60**	0.70	69.4	**0.57**	**0.64**	**54.3**	0.65	0.74	21.2	0.66	0.72	35.3	0.60	0.68	14.5	0.70	0.81
Trace2Skill	**29.2**	0.66	0.74	69.5	0.65	0.70	53.7	0.66	0.76	20.8	0.68	0.74	35.5	**0.58**	0.66	15.0	0.66	0.72
FORKPILOT	**29.2**	0.61	**0.66**	**71.0**	0.63	0.67	54.0	0.66	**0.70**	**22.5**	0.62	**0.66**	**36.4**	0.62	0.66	15.7	**0.64**	**0.70**
Qwen3.5-122B-A10B																		
BoN-5	25.1	1.08	1.20	66.0	1.14	1.25	49.1	1.11	1.23	19.5	1.16	1.24	34.9	1.09	1.22	12.1	1.13	1.25
RTS	29.8	1.00	1.00	72.0	1.00	1.00	53.4	1.00	1.00	23.2	1.00	1.00	37.0	1.00	1.00	16.6	1.00	1.00
ACE	28.6	0.63	0.72	70.9	0.61	0.69	52.4	0.69	0.76	21.6	0.66	0.75	35.6	0.62	0.69	15.4	0.66	0.73
ReasoningBank	28.8	0.65	0.74	71.5	0.62	0.71	51.9	0.66	0.74	21.8	**0.62**	0.71	35.2	0.65	0.73	15.1	**0.64**	0.73
Reflexion	27.3	0.77	0.85	69.8	0.75	0.83	49.8	0.80	0.87	20.0	0.77	0.85	34.1	0.74	0.82	14.3	0.80	0.87
GEPA	29.3	**0.60**	0.69	**72.4**	**0.59**	**0.66**	**53.6**	**0.65**	0.73	22.5	0.64	0.71	36.9	0.61	0.68	16.0	0.67	0.75
Trace2Skill	29.5	0.64	0.72	71.6	0.62	0.70	53.1	0.66	0.74	22.7	0.65	0.72	37.1	**0.60**	**0.67**	16.3	**0.64**	**0.71**
FORKPILOT	**30.0**	0.63	**0.68**	**72.4**	0.62	0.67	53.5	0.68	**0.72**	**23.4**	**0.62**	**0.67**	**37.3**	0.64	0.69	**16.8**	0.66	**0.71**

4.2 CONSISTENT EFFICIENCY ACROSS VARIOUS BENCHMARKS

Across six diverse agentic benchmarks, FORKPILOT consistently achieves a favorable quality-efficiency tradeoff, substantially reducing retrospective-search cost while preserving task quality within Table 1. On average, FORKPILOT reduces raw and uncached inference tokens by 36.2% and 30.3%, respectively, while slightly improving task score by $+0.23$ points relative to full RTS. The gains remain consistent across heterogeneous environments: benchmark-level raw-token savings range from 31.8% on SWE-bench to 39.8% on AutomationBench, while the corresponding mean

[2] https://deploymentsafety.openai.com/gpt-5-6

[3] https://www.anthropic.com/news/claude-opus-4-8

score changes remain within $+0.50$ points. Overall, FORKPILOT removes roughly one-third of retrospective inference across diverse agentic tasks without sacrificing overall task performance.

4.3 ROBUST EFFICIENCY ACROSS DIFFERENT BACKBONE SCALES

As illustrated in Table 2, across backbone scales, FORKPILOT consistently preserves task quality while reducing retrospective-search cost. As the backbone scales from 4B to 9B and 27B, average raw-token savings increase from 28.5% to 33.7% and 37.0%, while uncached-token savings increase from 23.7% to 28.8% and 32.5%. At 122B-A10B, the savings remain comparable at 35.8% and 31.0%, indicating that the efficiency benefit largely saturates at the largest scale rather than disappearing. Meanwhile, mean score changes relative to full RTS remain stable at $+0.27$, $+0.27$, $+0.30$, and $+0.23$ points across the four scales. These results show that FORKPILOT's quality-efficiency advantage is not tied to a particular model size and persists from small to substantially larger backbones.

4.4 CONSISTENT EFFICIENCY ON FRONTIER BLACK-BOX MODELS

Deployed in a production agentic system where the agent uses GPT-5.6 Sol and Claude Opus 4.8 served through production inference platforms employing state-of-the-art caching, FORKPILOT retains substantial efficiency gains (Table 3). Across AutomationBench and τ^3-Bench, it uses only 58.8-77.8% of RTS raw tokens and 40.8-76.3% of uncached tokens, corresponding to 22.2-41.2% and 23.7-59.2% savings, respectively, while remaining comparable in performance to RTS. The gains are not reproduced by simply extending a single retrospective trajectory: compared with RDS, FORKPILOT achieves higher task score in all four settings while reducing raw inference cost by 53.1-78.9%. It also exceeds BoN-5 in task quality investing much less tokens. GHCP occupies a substantially cheaper but lower-quality operating point in three settings. Overall, the results show that FORKPILOT's adaptive retrospective allocation transfers to frontier black-box backbones, preserving much of full-search quality while substantially reducing token consumption. Appendix E reports the corresponding absolute inference-token totals and RTS-normalized costs for all black-box settings. In our deployment, FORKPILOT also demonstrates robustness to changes in rollout scheduling of host production agentic system while achieving further gains in token efficiency (Appendix C.4).

Table 3: Black-box evaluation on proprietary backbones. S is task score (0-100); R/U are raw/uncached inference-token ratios to full RTS. Lower R/U indicates lower deployment cost. The best is in **Bold** and the second best is in underline.

	GPT-5.6 Sol						Opus 4.8					
	Automation			τ^3-Bench			Automation			τ^3-Bench		
Method	S	R	U	S	R	U	S	R	U	S	R	U
BoN-5	59.83	2.030	1.689	59.28	0.989	1.138	55.17	1.491	1.025	44.44	1.137	2.743
RDS	68.83	1.632	0.697	55.67	2.783	0.775	62.50	3.188	1.192	29.90	1.685	0.454
GHCP	33.50	0.846	0.481	43.30	0.208	0.186	39.83	0.487	0.189	41.24	0.171	0.251
RTS	**71.83**	1.000	1.000	**67.53**	1.000	1.000	**70.83**	1.000	1.000	44.67	1.000	1.000
FORKPILOT	69.00	0.764	0.556	64.43	0.588	0.408	66.67	0.778	0.763	**48.31**	0.611	0.569

4.5 LEAKAGE-FREE FEEDBACK AND SAMPLE-EFFICIENT SELF-EVOLUTION

Feedback-leakage audit. We examine whether the observed performance depends on evaluator feedback in the retrospective prompt for agent. The *feedback-masked* only retains task execution history, but removes explicit rewards, rubric/grader-specific fields, and reference-answer cues. Across the four audited settings, feedback-masked scores are similar to the original (Table 4). We also shuffle the feedback across tasks, introducing task-mismatched noise. Scores fall by 1.4-2.8 points relative to feedback-masked, yet remain broadly comparable across different backbone scales and variate benchmarks. These results indicate that the observed gains are not driven by evaluator-information leakage through retrospective prompt.

Table 4: Feedback-leakage audit on two representative backbones.

Backbone	Feedback	Automation	τ^3-Bench
Qwen3.5-9B	Original	14.5	44.2
	Masked	14.4	44.0
	Shuffled	13.0	41.7
Qwen3.5-27B	Original	25.6	65.0
	Masked	25.5	64.8
	Shuffled	23.2	62.0

Sample-efficient self-evolution. We also study the sample efficiency of FORKPILOT's self-evolution on Automation Bench by varying the fraction of the training split available for policy learning (Fig. 2). We report the fraction of the full-training improvement recovered, $R(f) = (S_f - S_0)/(S_{100} - S_0)$. Remarkably, with only 25% of the training split(13.3% of the total dataset), FORKPILOT already captures 81-85% of the improvement obtained using all training samples across the evaluated settings; the remaining 75% contributes only another 15-19%. This strongly front-loaded scaling shows that FORKPILOT can rapidly bootstrap most of its eventual search-allocation improvement from limited completed experience, making its self-evolution highly sample efficient rather than dependent on a large training pool.

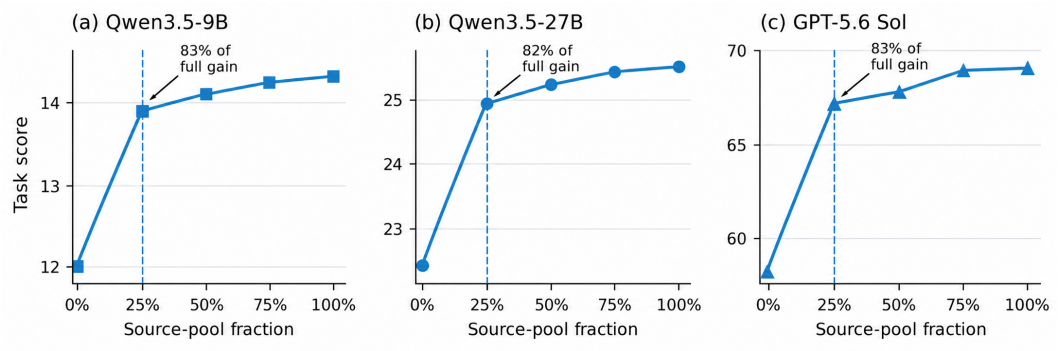


Figure 2: Fraction of the full-training improvement recovered as the amount of training experience available for policy learning increases on Automation Bench. The percentages on the x-axis denote the fraction of the total training split used for policy learning.

5 RELATED WORK

Retrospective Search and Adaptive Computation. Retrospective search improves reasoning by revisiting earlier decisions and exploring alternative trajectories. Tree of Thoughts (Yao et al., 2023a), Language Agent Tree Search (Zhou et al., 2024a), and ExACT (Yu et al., 2025) instantiate this idea for reasoning and interactive agents. More broadly, metareasoning and resource-rational planning study which computations are worth performing under limited resources (Russell & Wefald, 1991; Hay et al., 2012; Callaway et al., 2018). Recent LLM methods further adapt test-time computation according to query difficulty, intermediate confidence, or subproblem complexity, including compute-optimal scaling (Snell et al., 2025), input-adaptive allocation (Damani et al., 2025), Certaindex (Fu et al., 2025), Learning When to Plan (Paglieri et al., 2025), and Plan-and-Budget (Lin et al., 2025).

Learning to Control Search. A complementary line of work learns from completed interactions to improve future reasoning. Reflexion (Shinn et al., 2023), ReasoningBank (Ouyang et al., 2026), ACE (Zhang et al., 2026), GEPA (Agrawal et al., 2026), and Trace2Skill (Ni et al., 2026) use prior trajectories to construct reflections, memories, optimized context, or reusable skills. These methods primarily improve what information is available to the agent rather than learning how much retrospective computation should be allocated during execution.

6 CONCLUSION

Long-horizon, multi-call reasoning makes retrospective search a dynamic computation-allocation problem: delayed outcomes obscure the contribution of intermediate search decisions, while evolving execution evidence can make previously learned estimates stale. We characterize these challenges as *Attribution Complexity* and *Adaptation Complexity*, and introduce Search Value Dynamics (SVD) to characterize the evolving trade-off between the gain and cost of retrospective search. Building on SVD, we propose FORKPILOT, a self-evolving two-stage policy-learning framework. FORKPILOT first learns a search-value policy offline from completed trajectories, then, in the second stage, makes search decisions on the current observations and then self-evolves, incorporating newly completed trajectories into subsequent policy updates. Across six benchmarks and multiple widely used LLM backbone families, FORKPILOT achieves comparable state-of-the-art performance while reducing deployment-time token usage by up to 59.2%.

Reproducibility Statement

We provide detailed information to support reproduction of both the theoretical and empirical results. Appendix A gives the derivations, assumptions, proofs, and scope of the Search Value Dynamics analysis. Appendix B specifies the controller architecture, action representation, paired-replay data construction, empirical margin, runtime decision rule, update protocol, and baseline implementations. Appendix C documents the evaluated model checkpoints, benchmark revisions, task-grouped source/test splits, evaluation manifests, resource constraints, and token accounting. Appendix D further specifies the cold-start, self-evolution, stopping, and mechanism ablation protocols. Hyperparameter and selection rules are determined using source data and validation splits without access to held-out task outcomes. Together, these specifications are intended to make the experimental protocol, learning procedure, and reported efficiency measurements independently reproducible.

References

Lakshya A. Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J. Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

Apple. MM-ToolSandbox: A unified framework for evaluating visual tool-calling agents. https://github.com/apple/ml-mmtoolsandbox, 2026. Official benchmark repository.

Frederick Callaway, Falk Lieder, Priyam Das, Sayan Gul, Paul M. Krueger, and Thomas L. Griffiths. A resource-rational analysis of human planning. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pp. 178–183, 2018.

Mehul Damani, Idan Shenfeld, Andi Peng, Andreea Bobu, and Jacob Andreas. Learning how hard to think: Input-adaptive allocation of LM computation. In *International Conference on Learning Representations*, 2025.

DeepSeek-AI. DeepSeek-R1: Official model card and weights. https://huggingface.co/deepseek-ai/DeepSeek-R1, 2025. Full original release; accessed 2026-09-18.

DeepSeek-AI. DeepSeek-V4-Flash: Official model card and weights. https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash, 2026. Accessed 2026-09-18.

Aladin Djuhera, Swanand Ravindra Kadhe, Farhan Ahmed, Syed Zawad, Heiko Ludwig, and Holger Boche. Tsr: Trajectory-search rollouts for multi-turn rl of llm agents. *arXiv preprint arXiv:2602.11767*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training, 2025. *URL https://arxiv. org/abs/2505.10978*, 2025.

Yichao Fu, Junda Chen, Siqi Zhu, Zheyu Fu, Zhongdongming Dai, Yonghao Zhuang, Yian Ma, Aurick Qiao, Tajana Rosing, Ion Stoica, and Hao Zhang. Efficiently scaling LLM reasoning programs with certaindex. In *Advances in Neural Information Processing Systems*, 2025.

Isaac Gibbs and Emmanuel Candès. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34, 2021.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/fef126561bbf9d4467dbb8d27334b8fe-Abstract-Conference.html.

Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 346–355, 2012.

Samuel Holt, Max Ruiz Luyten, Thomas Pouplin, and Mihaela van der Schaar. Fact-augmented lookahead planning for llm agents, 2026. URL https://arxiv.org/abs/2506.09171.

Jing Yu Koh, Stephen McAleer, Daniel Fried, and Ruslan Salakhutdinov. Tree search for language model agents. *arXiv preprint arXiv:2407.01476*, 2024.

Junhong Lin, Xinyue Zeng, Jie Zhu, Song Wang, Julian Shun, Jun Wu, and Dawei Zhou. Plan and budget: Effective and efficient test-time scaling on large language model reasoning. *arXiv preprint arXiv:2505.16122*, 2025. URL https://arxiv.org/abs/2505.16122.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Meituan LongCat. VitaBench: Benchmarking LLM agents with versatile interactive tasks. https://huggingface.co/datasets/meituan-longcat/VitaBench, 2026. Official dataset card. Accessed 2026-09-10.

Moonshot AI. Kimi-K3: Official model card and weights. https://huggingface.co/moonshotai/Kimi-K3, 2026. Accessed 2026-09-18.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Erchao Zhao, Xiaoxi Jiang, and Guanjun Jiang. Trace2skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

Siru Ouyang, Jun Yan, I-Hung Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, Vishy Tirumalashetty, George Lee, Mahsan Rofouei, Hangfei Lin, Jiawei Han, Chen-Yu Lee, and Tomas Pfister. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *The Fourteenth International Conference on Learning Representations*, 2026.

Davide Paglieri, Bartłomiej Cupiał, Jonathan Cook, Ulyana Piterbarg, Jens Tuyls, Edward Grefenstette, Jakob Nicolaus Foerster, Jack Parker-Holder, and Tim Rocktäschel. Learning when to plan: Efficiently allocating test-time compute for LLM agents. *arXiv preprint arXiv:2509.03581*, 2025.

Yichen Pan, Dehan Kong, Sida Zhou, Cheng Cui, Yifei Leng, Bing Jiang, Hangyu Liu, Yanyi Shang, Shuyan Zhou, Tongshuang Wu, et al. Webcanvas: Benchmarking web agents in online environments. *arXiv preprint arXiv:2406.12373*, 2024.

Qwen Team. Qwen3.5-122B: Official model card and weights. https://huggingface.co/Qwen/Qwen3.5-122B-A10B, 2026a. Accessed 2026-09-18.

Qwen Team. Qwen3.5-27B: Official model card and weights. https://huggingface.co/Qwen/Qwen3.5-27B, 2026b. Accessed 2026-09-18.

Qwen Team. Qwen3.5-4B: Official model card and weights. https://huggingface.co/Qwen/Qwen3.5-4B, 2026c. Accessed 2026-09-18.

Qwen Team. Qwen3.5-9B: Official model card and weights. https://huggingface.co/Qwen/Qwen3.5-9B, 2026d. Accessed 2026-09-18.

Qwen Team. Qwen3.8-27B: Official model card and weights. https://huggingface.co/Qwen/Qwen3.8-27B, 2026e. Accessed 2026-09-18.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49(1–3):361–395, 1991. doi: 10.1016/0004-3702(91)90015-C.

ServiceNow. WorkArena and WorkArena++: Official benchmark repository. https://github.com/ServiceNow/workarena, 2026. Official benchmark repository.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *International Conference on Learning Representations*, 2025.

SWE-bench. SWE-bench: Official benchmark and evaluation harness. https://github.com/swe-bench/SWE-bench, 2026. Official implementation. Accessed 2026-09-10.

Hui-Ze Tan, Xiao-Wen Yang, Hao Chen, Jie-Jing Shao, Yi Wen, Yuteng Shen, Weihong Luo, Xiku Du, Lan-Zhe Guo, and Yu-Feng Li. Hindsight credit assignment for long-horizon llm agents. *arXiv preprint arXiv:2603.08754*, 2026.

Wenjie Tang, Minne Li, Sijie Huang, Liquan Xiao, and Yuan Zhou. Rewarding beliefs, not actions: Consistency-guided credit assignment for long-horizon agents. *arXiv preprint arXiv:2605.20061*, 2026.

Tau-Bench Contributors. τ^3-bench: Tool-agent-user interaction benchmark repository. https://github.com/codesque16/tau3-bench, 2026. Official benchmark repository; accessed 2026-09-20.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. *arXiv preprint arXiv:2504.01382*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=WE_vluYUL-X.

Xiao Yu, Baolin Peng, Vineeth Vajipey, Hao Cheng, Michel Galley, Jianfeng Gao, and Zhou Yu. ExACT: Teaching AI agents to explore with reflective-MCTS and exploratory learning. In *International Conference on Learning Representations*, 2025.

Zapier. AutomationBench: A benchmark for evaluating AI agents on realistic business workflows. https://github.com/zapier/AutomationBench, 2026. Official implementation. Accessed 2026-09-10.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

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

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024b.

A Derivation and Scope of Search Value Dynamics

A.1 Variance identities for paired search effects

Fix a source input z_{src}, a feasible action u, and the paired-replay rule Γ. As in Section 3.1, let $D(u) = \Delta C(u)/\bar{C}_{\mathrm{src}}$, $\lambda_D = \lambda \bar{C}_{\mathrm{src}}$, and $W(u) = Y(u) - \lambda_D D(u)$. The first calculation conditions on the fixed source input and action under the source law Q. Whenever the component second moments exist, expansion of the variance gives

$$\begin{aligned} V^{\Gamma}_{\mathrm{src}}(u) &= \mathrm{Var}_Q[W(u) \mid z_{\mathrm{src}}] \\ &= A^{\Gamma}_{\mathrm{src}}(u) + \lambda_D^2 \,\mathrm{Var}_Q[D(u) \mid z_{\mathrm{src}}] - 2\lambda_D \,\mathrm{Cov}_Q(Y(u), D(u) \mid z_{\mathrm{src}}), \end{aligned} \tag{5}$$

where $A^{\Gamma}_{\mathrm{src}}(u) = \mathrm{Var}_Q[Y(u) \mid z_{\mathrm{src}}]$. Thus paired reward variation alone does not determine the sampling variation of the net effect. If $W_1(u), \ldots, W_n(u)$ are conditionally independent draws from this same law, their mean $\overline{W}_n(u)$ satisfies

$$\mathbb{E}_Q[\overline{W}_n(u) \mid z_{\mathrm{src}}] = \mathbb{E}_Q[W(u) \mid z_{\mathrm{src}}], \qquad \mathrm{Var}_Q[\overline{W}_n(u) \mid z_{\mathrm{src}}] = \frac{V^{\Gamma}_{\mathrm{src}}(u)}{n}.$$

Independence across *actions* is not needed for this calculation or for the simultaneous bound below.

The other variance identity in Section 3.1 follows from the conditional law of total variance. Let H be the permitted history and $Z = \phi(H)$, and fix an action feasible across the histories being compared. Writing $g(H, u) = \mathbb{E}_Q[W(u) \mid H]$, the fact that Z is a function of H implies

$$\mathrm{Var}_Q[W(u) \mid Z] = \mathbb{E}_Q[\mathrm{Var}_Q(W(u) \mid H) \mid Z] + \mathrm{Var}_Q\big(g(H, u) \mid Z\big). \tag{6}$$

The first term reflects variation remaining after the full permitted history is known; the second reflects different conditional values assigned the same controller input. Neither identity by itself implies that pairing reduces variance relative to a particular unpaired estimator.

A.2 Proof of Theorem 1

We use the notation and fixed-quantity conditions of Theorem 1. In particular, $\mathcal{F}_t = \mathcal{U}_t \setminus \{u^0\}$ contains $K \geq 1$ actions, the source and current laws Q, P use the same Γ, and the action set, inputs, and sample counts are fixed independently of the replay draws. For $L \in \{P, Q\}$, write $m_L(z, u) = \mathbb{E}_L[Y(u) \mid z]$ and $c_L(z, u) = \mathbb{E}_L[D(u) \mid z]$, as in the main text. Define the population source net effect

$$g_Q(u) = m_Q(z_{\mathrm{src}}, u) - \lambda_D c_Q(z_{\mathrm{src}}, u) = \mathbb{E}_Q[W(u) \mid z_{\mathrm{src}}].$$

All these conditional expectations are understood at inputs where their specified conditional laws are defined. The same feasible action and replay rule are used on both sides of each comparison.

Step 1: simultaneous source sampling error. For each $u \in \mathcal{F}_t$, let $\overline{W}_{\mathrm{src}}(u)$ denote the mean of its n_u conditionally i.i.d. source paired net effects. Conditional on the fixed quantities in the theorem,

$$\mathbb{E}[\overline{W}_{\mathrm{src}}(u)] = g_Q(u), \qquad \mathrm{Var}[\overline{W}_{\mathrm{src}}(u)] = \frac{V^{\Gamma}_{\mathrm{src}}(u)}{n_u}.$$

When $V^{\Gamma}_{\mathrm{src}}(u) > 0$, Chebyshev's inequality with $a_u = \sqrt{K V^{\Gamma}_{\mathrm{src}}(u)/(n_u \delta)}$ yields

$$\Pr\big(\big|\overline{W}_{\mathrm{src}}(u) - g_Q(u)\big| > a_u\big) \leq \frac{V^{\Gamma}_{\mathrm{src}}(u)}{n_u a_u^2} = \frac{\delta}{K}.$$

If $V^{\Gamma}_{\mathrm{src}}(u) = 0$, the sample mean equals $g_Q(u)$ almost surely, so the same event bound holds with $a_u = 0$. A union bound over the K actions therefore gives an event $\mathcal{E}_\delta$ with $\Pr(\mathcal{E}_\delta) \geq 1 - \delta$ on which, simultaneously for all $u \in \mathcal{F}_t$,

$$\big|\overline{W}_{\mathrm{src}}(u) - g_Q(u)\big| \leq \sqrt{\frac{K V^{\Gamma}_{\mathrm{src}}(u)}{n_u \delta}}. \tag{7}$$

The replay samples for different actions need not be independent. The source-fitted predictor may also depend on these samples: its realized discrepancy from their mean is retained explicitly as $e_{\mathrm{src}}(u) = |\widehat{G}_{\mathrm{src}}(u) - \overline{W}_{\mathrm{src}}(u)|$.

Step 2: change of input and law. Add and subtract the current-law reward mean at z_{src}. Since $G_t(u) = m_P(z_t, u) - \lambda_D c_P(z_t, u)$, the following is an exact identity:

$$\begin{aligned} G_t(u) - g_Q(u) = &\big[m_P(z_t, u) - m_P(z_{\mathrm{src}}, u)\big] \\ &+ \big[m_P(z_{\mathrm{src}}, u) - m_Q(z_{\mathrm{src}}, u)\big] \\ &- \lambda_D\big[c_P(z_t, u) - c_Q(z_{\mathrm{src}}, u)\big]. \end{aligned} \tag{8}$$

The first bracket has magnitude $\Omega_{\mathrm{src}\to t}(u)$; the third has magnitude $\xi_{\mathrm{src},t}(u)$. To bound the second bracket, let $P^u_{z_{\mathrm{src}}}$ and $Q^u_{z_{\mathrm{src}}}$ be the corresponding conditional laws of $Y(u)$. Because $R \in [0,1]$, the paired reward difference satisfies $Y(u) \in [-1,1]$. With $\mathrm{TV}(\mu,\nu) = \sup_A |\mu(A) - \nu(A)|$,

$$\begin{aligned} |m_P(z_{\mathrm{src}}, u) - m_Q(z_{\mathrm{src}}, u)| &= \left| \int y \, d\big(P^u_{z_{\mathrm{src}}} - Q^u_{z_{\mathrm{src}}}\big)\,(y) \right| \\ &\le 2\,\mathrm{TV}\big(P^u_{z_{\mathrm{src}}}, Q^u_{z_{\mathrm{src}}}\big) = S_{\mathrm{src}}(u). \end{aligned} \tag{9}$$

Consequently, by the triangle inequality and $\lambda_D \ge 0$,

$$|G_t(u) - g_Q(u)| \le \Omega_{\mathrm{src}\to t}(u) + S_{\mathrm{src}}(u) + \lambda_D \xi_{\mathrm{src},t}(u).$$

When $P^u_{z_{\mathrm{src}}}$ is absolutely continuous with respect to $Q^u_{z_{\mathrm{src}}}$ and the KL divergence is finite, Pinsker's inequality additionally gives $S_{\mathrm{src}}(u) \le \sqrt{2 D_{\mathrm{KL}}(P^u_{z_{\mathrm{src}}} \| Q^u_{z_{\mathrm{src}}})}$.

Step 3: combine the errors. On the single event $\mathcal{E}_\delta$, for every $u \in \mathcal{F}_t$,

$$\begin{aligned} |G_t(u) - \widehat{G}_{\mathrm{src}}(u)| &\le |G_t(u) - g_Q(u)| + |g_Q(u) - \overline{W}_{\mathrm{src}}(u)| + |\overline{W}_{\mathrm{src}}(u) - \widehat{G}_{\mathrm{src}}(u)| \\ &\le e_{\mathrm{src}}(u) + \sqrt{\frac{K V^{\Gamma}_{\mathrm{src}}(u)}{n_u \delta}} + \Omega_{\mathrm{src}\to t}(u) \\ &\quad + S_{\mathrm{src}}(u) + \lambda_D \xi_{\mathrm{src},t}(u) = \beta^{\delta}_{\mathrm{src},t}(u). \end{aligned} \tag{10}$$

This proves Equation 1. For any $\eta > 0$, the condition

$$n_u \ge \frac{K V^{\Gamma}_{\mathrm{src}}(u)}{\delta \eta^2}$$

is sufficient to make the displayed sampling term for that action at most η; it does not control the other terms.

A.3 Decision consequences and scope

For non-skip actions, set $L^\delta_t(u) = \widehat{G}_{\mathrm{src}}(u) - \beta^\delta_{\mathrm{src},t}(u)$ and $U^\delta_t(u) = \widehat{G}_{\mathrm{src}}(u) + \beta^\delta_{\mathrm{src},t}(u)$, and extend both quantities to skip by $L^\delta_t(u^0) = U^\delta_t(u^0) = G_t(u^0) = 0$. Equation 10 implies $L^\delta_t(u) \le G_t(u) \le U^\delta_t(u)$ simultaneously over $\mathcal{U}_t$ on $\mathcal{E}_\delta$.

If $u_L \in \arg\max_{u \in \mathcal{F}_t} L^\delta_t(u)$ and $L^\delta_t(u_L) > 0$, then $G_t(u_L) \ge L^\delta_t(u_L) > 0$; in particular, skip cannot maximize the true local net value. Let $u^\star \in \arg\max_{u \in \mathcal{U}_t} G_t(u)$, so $u^\star \in \mathcal{F}_t$. Because $L^\delta_t(u_L) \ge L^\delta_t(u^\star)$,

$$\begin{aligned} G_t(u^\star) - G_t(u_L) &\le U^\delta_t(u^\star) - L^\delta_t(u_L) \\ &\le U^\delta_t(u^\star) - L^\delta_t(u^\star) = 2\beta^\delta_{\mathrm{src},t}(u^\star). \end{aligned} \tag{11}$$

If instead $\max_{u \in \mathcal{F}_t} U^\delta_t(u) \le 0$, then $G_t(u) \le 0 = G_t(u^0)$ for every non-skip action. Skip is therefore a local maximizer, possibly tied with another action. These are precisely the positive-LCB, skip-optimality, and local-regret implications stated after Theorem 1.

For completeness, a related information inequality follows without a sampling argument. Let $\mathcal{U}$ be a fixed finite action set feasible for all histories under comparison, and let $g(H,u)$ be integrable for every $u \in \mathcal{U}$. Since $g(H,u) \le \max_{v \in \mathcal{U}} g(H,v)$ for each fixed u, conditioning on $Z = \phi(H)$ and then maximizing gives

$$\max_{u \in \mathcal{U}} \mathbb{E}[g(H,u) \mid Z] \le \mathbb{E}\left[\max_{u \in \mathcal{U}} g(H,u) \,\middle|\, Z \right].$$

Taking expectations preserves the inequality. The comparison holds under the same history distribution and fixed menu; it only describes the possible value of more informative inputs.

Theorem 1 conditions on fixed inputs, menu, counts, and paired-replay rule. Its event need not cover menus or inputs chosen adaptively after inspecting these replay samples. Moreover, $e_{\mathrm{src}}(u)$ is a realized source-sample fitting discrepancy, not an independently validated bound on prediction error; it may be large. At deployment the controller evaluates its predictor at the refreshed input z_t, while Theorem 1 bounds reuse of $\widehat{G}_{\mathrm{src}}(u)$ at z_{src}. Neither the theorem nor the information inequality certifies the learned operational margin or a closed-loop regret guarantee for the deployed policy, whose subsequent decisions may differ from the fixed continuation rule ρ.

B Model Configuration

This section provides the action-level specification omitted from the system walkthrough. We fix a task and omit task index i. Source collection, feature construction, and controller fitting use the task boundaries and information permissions defined in Section 2.

B.1 State and action representation

Each permitted event is converted into categorical and scalar features: event type, tool identity, branch relations, success or error category, completed-branch score summaries, depth, search count, and accumulated token usage. Bounded deterministic features represent tool-argument and observation structure, including key names, value types, and length bins. Features are constructed from the prefix available at the recorded decision; later events never overwrite an earlier example.

A recurrent encoder maps the observable controller state to a representation $r_t = f_\psi(z_t)$. The host supplies the state, feasible-action mask, branch relations, and remaining resources. Action embeddings describe each host-provided search configuration. Resource features use source-only normalization. Feasibility is enforced from the host's actual reservation rule rather than the learned mean-cost prediction.

Table 5: Default controller configuration. Host caps and selected trade-off parameters are recorded separately in each evaluation manifest.

Component	Setting
Event embedding / recurrent state	64 / 128 dimensions; one GRU layer
Prediction ensemble	5 action-conditioned heads; task-bootstrap weights
Head architecture	One 64-unit hidden layer; separate gain and cost outputs
Source fitting	Adam, learning rate 10^{-3}, batch size 64; inner-source validation stopping
Regularization	Gradient norm cap 1; task-balanced loss weights
Source partitions	80% predictor fit, 10% margin fit, 10% residual calibration, grouped by task
Action menu	$k \in \{1, k_{\max}\}$; $d \in \{\lceil d_{\max}/2 \rceil, d_{\max}\}$; skip
Probe reservation	At most 10% of remaining budget; one fork and short cap
Runtime support	≥ 20 distinct source tasks and five-nearest-task distance at or below the source 90th percentile
Unsupported-state fallback	Source-selected Efficient RTS configuration under the same hard cap
Margin target quantile	$1 - \delta_{\mathrm{cal}} = 0.9$
Cost normalization	Source median host task cost, fixed at deployment
Trade-off selection	$\lambda \in \{0, 0.01, 0.03, 0.1, 0.3\}$ on source validation

B.2 Paired replay protocol, targets, and value prediction

Source states are sampled from a mixture of host budgets, including full-budget search, so that low apparent progress does not censor all recovery examples. The state rule is fixed before downstream reward is read. Candidate actions are sampled with logged nonzero probabilities.

For each candidate action u, the sandbox restores the same state for u and the skip reference u^0. Both arms use the same permissions, remaining budget, total-budget rule, and subsequent execution policy;

either arm may search at later opportunities. Let $\bar{C}_{\mathrm{src}} > 0$ be the cost scale fixed from source tasks. For each paired replicate q, the targets are

$$\widetilde{Y}_q = R(\tau_q^u) - R(\tau_q^0), \qquad \widetilde{D}_q = \frac{C(\tau_q^u) - C(\tau_q^0)}{\bar{C}_{\mathrm{src}}}. \tag{12}$$

Each $C(\tau_q^a)$ includes computation used by all search branches generated within arm a, including branches that are not selected. When two continuations per arm are collected, corresponding replicate indices form two paired records.

A record stores the state identifier, permitted-prefix hash, candidate action and its sampling probability, paired outcomes and costs, replicate index, replay-protocol identifier, continuation-policy identifier, and source-task identifier. Protocol Γ shares randomness only for exogenous events whose meaning and distribution do not depend on the chosen action. Such draws are keyed by task, state, replicate, and event identity. Backbone sampling and action-dependent environment draws use arm-specific keys, preserving the marginal execution law of each arm. Source collection and paired evaluation use the same replay protocol; controller fitting uses the resulting records. A pair is excluded from fitting if restoration fails to reproduce the state hash, permissions, or tool state, or if its effects cannot be isolated within the sandbox.

Table 6: Paired-supervision collection statistics. Candidate action–skip pairs use two continuations per arm. Rejected pairs are excluded before fitting; restoration agreement compares restored decision state and permission hashes before either continuation runs.

Benchmark	Checkpoints	Candidates	Accepted	Reject (%)	Restore (%)
AutomationBench	8,640	17,280	16,812	2.7	99.8
τ^3-Bench	6,480	12,960	12,532	3.3	99.7
SWE-bench Verified	5,940	11,880	11,215	5.6	99.4
MM-ToolSandbox	6,220	12,440	12,079	2.9	99.8
WorkArena++	5,760	11,520	10,967	4.8	99.5
VitaBench	7,040	14,080	13,587	3.5	99.6

Isolated execution of parallel rollouts in real-world live environment. Parallel multi-turn rollouts can interfere through shared filesystem or database assets in a live real-world environment: writes by one rollout may alter another's observations, overwrite its changes, or invalidate subsequent tool operations. This cross-rollout interference also undermines the intended comparison between paired action and skip executions. Our production agentic system (host) therefore routes supported tool calls through a session-scoped virtualization layer. Each rollout receives a distinct session identifier and a private execution view; paired replay arms are initialized from the same base snapshot. Writes are staged in session-local overlays, and subsequent reads and tool outputs reflect the session's own updates without exposing other rollouts' writes. Retrospective forks inherit the parent's state at the chosen fork point while isolating subsequent modifications. Replay-generated changes remain sandboxed and are discarded after outcome collection. During production execution, only the final selected trajectory's staged changes are eligible for publication to live assets; unselected overlays are discarded. Publication requires validation of relevant state dependencies and permissions against the current live environment, with conflicts requiring revalidation or re-execution. Isolation covers only resources mediated by the virtualization layer, and commit guarantees follow the supported storage transaction boundaries. Irreversible external effects must be deferred or excluded from speculative execution.

For bootstrap head j, the fitting loss is

$$\mathcal{L}_j = \frac{1}{|\mathcal{D}_j|} \sum_{q \in \mathcal{D}_j} \left[\left(\widehat{m}^{(j)}(r_q, u_q) - \widetilde{Y}_q \right)^2 + \left(\widehat{c}^{(j)}(r_q, u_q) - \widetilde{D}_q \right)^2 \right]. \tag{13}$$

Whole tasks, rather than prefixes, are bootstrapped. The ensemble's predicted net value is

$$\widehat{G}_t(u) = \frac{1}{M} \sum_{j=1}^{M} \left[\widehat{m}^{(j)}(r_t, u) - \lambda_D \widehat{c}^{(j)}(r_t, u) \right], \qquad \lambda_D = \lambda \bar{C}_{\mathrm{src}}. \tag{14}$$

Here λ is the raw-cost coefficient in Section 2, and $\widehat{c}^{(j)}$ predicts $\Delta C/\bar{C}_{\mathrm{src}}$. Head disagreement $s_t(u)$ is an empirical uncertainty feature, not a posterior standard deviation. Source-data collection cost is reported separately and counts both executed arms, failed runs, and state-preparation calls; these extra collection runs are not added to either arm's deployment cost $C(\tau_q^a)$. Head disagreement $s_t(u)$ is an empirical uncertainty feature, not a posterior standard deviation. Ordinary unpaired traces can supply auxiliary targets, but they do not replace Eq. 12 without an explicit attribution-bias term. Source collection cost includes every executed arm, failed run, and state-preparation call.

B.3 Empirical margin and interval-consistent stopping rule

Reference support is computed in source-standardized representation space for each action configuration. The novelty score $d_t(u)$ is the mean distance to nearby source representations from distinct tasks, and $n_t(u)$ counts distinct supporting tasks. A positive monotone calibrator forms

$$a_t(u) = \mathrm{softplus}\Big(b + w_s s_t(u) + w_d d_t(u) + w_n[1 + n_t(u)]^{-1/2}\Big), \quad w_s, w_d, w_n \geq 0. \tag{15}$$

It minimizes pinball loss at quantile $1 - \delta_{\mathrm{cal}}$ on the margin-fitting split for the absolute paired net-target residual. On the residual-calibration split, one state-action pair is selected per task by a prespecified random rule. With $\widetilde{W}_q = \widetilde{Y}_q - \lambda \widetilde{D}_q/\bar{C}_{\mathrm{src}}$, define

$$r_q = |\widetilde{W}_q - \widehat{G}_q| - a_q, \qquad q_{\mathrm{cal}} = \max\big\{0, r_{(\lceil (N+1)(1-\delta_{\mathrm{cal}})\rceil)}\big\}. \tag{16}$$

The order statistic includes the conventional additional $+\infty$ value if its index exceeds N. Insufficient support therefore produces fallback instead of an artificially narrow margin. The resulting operational scores are

$$\widehat{B}_t(u) = a_t(u) + q_{\mathrm{cal}}, \qquad L_t(u) = \widehat{G}_t(u) - \widehat{B}_t(u), \qquad U_t(u) = \widehat{G}_t(u) + \widehat{B}_t(u). \tag{17}$$

Under exchangeability, this construction gives marginal coverage for a new *realized paired target*. Because the residual also contains continuation randomness, this target differs from the conditional mean $G_t(u)$. Closed-loop action selection further changes the encountered state distribution; adaptive conformal inference studies related online distribution shift (Gibbs & Candès, 2021). Accordingly, L_t and U_t are operational decision scores whose closed-loop effect is measured through held-out rollouts and the no-margin ablation in Section 4.4.

Let $\widehat{m}_t(u)$ and $\widehat{c}_t(u)$ denote the ensemble mean predictions. The lower and upper operational scores imply two distinct action-level boundaries:

$$\begin{aligned} L_t(u) > 0 &\iff \widehat{m}_t(u) > \lambda \widehat{c}_t(u) + \widehat{B}_t(u) \quad \text{(commit-eligible)}, \\ U_t(u) > 0 &\iff \widehat{m}_t(u) > \lambda \widehat{c}_t(u) - \widehat{B}_t(u) \quad \text{(probe-eligible)}. \end{aligned} \tag{18}$$

Both boundaries change with the current trajectory, candidate action, predicted cost, and empirical reliability; the global λ remains source-selected. They are not interchangeable: the conservative $+\widehat{B}_t$ boundary authorizes regular execution, whereas the permissive $-\widehat{B}_t$ boundary only keeps a low-cost probe eligible. Equations 1-4 aggregate these action-level tests. The system skips exactly when every feasible action fails the commit boundary and every feasible probe fails the probe boundary.

B.4 Runtime rule and pseudocode

The host provides low and full fork/depth configurations, removes duplicates, and masks infeasible actions through hard reservations. Equations 1-4 select the highest positive-lower-score action, otherwise the least-cost positive-upper-score probe, and otherwise skip the current retrospective round. A skip does not end the task. If the artifact or support check is unavailable, the declared host fallback applies within the same hard budget.

Algorithm 1 FORKPILOT: Closed-Loop Retrospective Search Allocation

Input: retrospective-search host $\mathcal{H}$, controller θ_n, source-support index $\mathcal{S}$, hard task budget B, trade-off coefficient λ_D

Initialize observable history h_0, search record T_0, and remaining budget $b_0 \leftarrow B$
$t \leftarrow 0$
while the host has not terminated **do**
  Advance $\mathcal{H}$ to the next retrospective-search opportunity
  Append newly permitted events to h_t
  Obtain the host-feasible action menu $\mathcal{U}_t$
  $\mathcal{F}_t \leftarrow \mathcal{U}_t \setminus \{u^0\}$
  Let $\mathcal{P}_t \subseteq \mathcal{F}_t$ denote bounded probe actions
  Form the observable state $z_t \leftarrow \phi(h_t, T_t, b_t)$
  Compute representation $r_t \leftarrow f_\psi(z_t)$
  if θ_n is unavailable **or** $\mathcal{S}(r_t, \mathcal{F}_t)$ fails **then**
    $u_t \leftarrow \text{HOSTFALLBACK}(\mathcal{H}, \mathcal{U}_t, b_t)$
  else
    for $u \in \mathcal{F}_t$ **do**
      Predict reward effects $\{\widehat{m}_t^{(j)}(u)\}_{j=1}^M$
      Predict normalized cost effects $\{\widehat{c}_t^{(j)}(u)\}_{j=1}^M$
      $\widehat{G}_t(u) \leftarrow \frac{1}{M}\sum_{j=1}^{M}\left[\widehat{m}_t^{(j)}(u) - \lambda_D \widehat{c}_t^{(j)}(u)\right]$
      Compute empirical margin $\widehat{B}_t(u)$
      $L_t(u) \leftarrow \widehat{G}_t(u) - \widehat{B}_t(u)$
      $U_t(u) \leftarrow \widehat{G}_t(u) + \widehat{B}_t(u)$
    end for
    $C_t \leftarrow \max_{u \in \mathcal{F}_t} L_t(u)$
    $Q_t \leftarrow \max_{u \in \mathcal{P}_t} U_t(u)$
    if $C_t > 0$ **then**
      $u_t \leftarrow \arg\max_{u \in \mathcal{F}_t} L_t(u)$
    else if $Q_t > 0$ **then**
      $u_t \leftarrow \arg \min_{\substack{u \in \mathcal{P}_t \\ U_t(u) > 0}} \widehat{c}_t(u)$
    else
      $u_t \leftarrow u^0$ ▷ skip this retrospective round
    end if
  end if
  $(h_{t+1}, T_{t+1}, b_{t+1}) \leftarrow \text{EXECUTE}(\mathcal{H}, u_t)$
  $t \leftarrow t + 1$
end while
if the completed task is eligible source experience **then**
  Add its valid paired records to $\mathcal{D}_{n+1}$
  $\theta_{n+1} \leftarrow \text{LEARN}(\mathcal{D}_{n+1})$
end if
Freeze the fitted controller before evaluating the next held-out block

Table 7: One end-to-end decision record (AutomationBench held-out trace AB-f2-0174). $\widehat{G}$, $\widehat{B}$, and L refer to the selected non-skip candidate; the probe at $t = 1$ is chosen by its positive upper score. The trace illustrates that a skip at $t = 3$ does not prevent renewed search after later evidence.

t	Newly visible evidence	Feasible actions	$\widehat{G}$	$\widehat{B}$	L	Decision
1	Lookup failed; no canonical record ID; 68% budget remains	probe / full / skip	0.018	0.026	-0.008	Probe
2	Probe returns canonical ID and matching account	full / low / skip	0.091	0.024	0.067	Full
3	Recovered branch raises permitted score from .61 to .83	full / low / skip	0.007	0.019	-0.012	Skip
4	Later tool return exposes a recoverable permission path	full / low / skip	0.054	0.021	0.033	Full

B.5 Online updates and censoring

A task can enter the next fitting block only after all required downstream outcomes are complete. Online replay calls are logged as learning cost. A retained full-budget source panel preserves recovery opportunities that an adaptive policy might otherwise stop observing. No-future-improvement labels from truncated executions are censored rather than treated as zero search value.

Task roles for predictor fitting, margin fitting, and residual calibration remain disjoint and stable as the source prefix grows. The encoder, heads, normalization, support index, and margin are versioned and frozen as one artifact before held-out evaluation. Within a task, changing tool observations refresh the controller input and decision scores, but do not update these fitted parameters. Across completed source tasks, a later artifact may be fitted and deployed to a subsequent block. Inner source validation can select update hyperparameters; the outer test panel cannot determine whether an update is deployed. At cold start, none of these fitted artifacts or statistics is available, so the system uses the host fallback until fitting and calibration become possible.

B.6 Baseline definitions

RTS and RDS. Full Retrospective Tree Search (RTS) uses the common retrospective-search host with a configuration selected on source validation to favor task score. Its three beams retain alternative continuations from earlier resumable states for exploration. The common scorer prioritizes promising branches for further expansion, providing exploitation within the search. At each eligible state, RTS executes the complete selected rollout allocation, subject to the hard host cap shared by all search-control methods. Retrospective Depth Search (RDS) reviews one initial rollout and continues only from its final resumable node. It does not fork from earlier states. RDS therefore tests how much additional depth along a single trajectory can achieve without exploring alternative retrospective branches.

The eight comparators in the seven-backbone matrix are BoN-5, RTS, RDS, ACE, ReasoningBank, Reflexion, GEPA, and Trace2Skill. BoN-5 samples five complete candidates and selects by the common scorer. ACE, ReasoningBank, Reflexion, GEPA, and Trace2Skill use their declared context, memory, or prompt intervention while sharing the evaluation harness and token accounting. These are end-to-end baselines with a different intervention interface from rollout allocation; Table 8 evaluates their complementarity through a ReasoningBank+ ForkPilot composition. BoN-1 and GHCP appear only in the separate frontier-model comparison and are not counted among the eight matrix comparators.

Table 8: Composition with ReasoningBank on AutomationBench. The combined system supplies ReasoningBank context to the actor while ForkPilot independently controls retrospective fork/depth allocation. Ratios are relative to RTS; paired intervals compare the composition against ForkPilot alone.

Backbone	Method	Score	R	U	ΔS [95% CI]	ΔR [95% CI]
Qwen3.5-9B	ReasoningBank	12.50	0.690	0.780	–	–
	ForkPilot	14.53	0.700	0.750	–	–
	ReasoningBank + ForkPilot	14.91	0.586	0.663	+0.38 [+0.09,+0.71]	-0.114 [-0.132,-0.098]
Qwen3.5-27B	ReasoningBank	25.40	0.690	0.730	–	–
	ForkPilot	25.61	0.650	0.700	–	–
	ReasoningBank + ForkPilot	26.08	0.552	0.621	+0.47 [+0.12,+0.83]	-0.098 [-0.117,-0.080]

C Evaluation Protocol and Accounting

C.1 Released model checkpoints

Table 9 maps every backbone label to a public weight release. No proprietary backbone is part of the evaluation matrix. The controller uses observable events and does not inspect model activations or gradients. Each run must pin the release revision, tokenizer, serving configuration, reasoning setting, decoding parameters, and tool-call serialization; an unpinned model family name is insufficient for reproduction.

Table 9: Open-weight backbone identities. Licenses refer to the listed releases. DeepSeek-R1-671B denotes the full original R1 release, not a distilled variant.

Backbone	Public checkpoint	Release license
DeepSeek-V4-Flash	`deepseek-ai/DeepSeek-V4-Flash` (DeepSeek-AI, 2026)	MIT
Kimi-K3	`moonshotai/Kimi-K3` (Moonshot AI, 2026)	Kimi K3 License
DeepSeek-R1-671B	`deepseek-ai/DeepSeek-R1` (DeepSeek-AI, 2025)	MIT
Qwen3.5-4B	`Qwen/Qwen3.5-4B` (Qwen Team, 2026c)	Apache 2.0
Qwen3.5-9B	`Qwen/Qwen3.5-9B` (Qwen Team, 2026d)	Apache 2.0
Qwen3.5-27B	`Qwen/Qwen3.5-27B` (Qwen Team, 2026b)	Apache 2.0
Qwen3.5-122B-A10B	`Qwen/Qwen3.5-122B-A10B` (Qwen Team, 2026a)	Apache 2.0
Qwen3.8-27B	`Qwen/Qwen3.8-27B` (Qwen Team, 2026e)	Apache 2.0

For image-bearing observations, the shared adapter is a frozen Qwen3.5-9B instance with a fixed prompt that serializes only the public observation into text and structured tool fields. Every evaluated backbone, including those with native visual capability, receives this same adapter interface. The adapter cannot read answers, grader state, or a future trajectory; it is not trained on held-out tasks. Its input and output tokens count toward each method's cost, including unsuccessful branches. The manifest records the adapter checkpoint and prompt hash. This design permits a consistent all-open-weight comparison including text-only backbones, but its visual-task scores describe the adapted interface and must not be conflated with native-interface benchmark results.

C.2 Task groups, outer folds, and source roles

Every comparison is tied to a manifest containing benchmark revision, exact task identifiers, task groups, environment seeds, model checkpoint revisions, adapter configuration, host-search settings, cache policy, feedback permissions, and controller artifact identifier. The label τ^3-Bench identifies the text configuration used in the matrix; its suite revision and evaluator settings must be fixed in the manifest rather than inferred from the label alone. AutomationBench's public evaluation panel is distinct from any private leaderboard.

Table 10: Evaluation manifest summary. Source/test counts are per outer fold; $k_{\max}$ and $d_{\max}$ are host fork and continuation-depth caps; the hard budget is relative to full RTS. All runs use the source-selected Efficient RTS fallback and the common support rule stated in text.

Benchmark	Source	Test	$k_{\max}$	$d_{\max}$	Hard cap
AutomationBench	400	200	4	6	1.00
τ^3-Bench	320	160	4	5	1.00
SWE-bench Verified	300	200	3	6	1.00
MM-ToolSandbox	320	160	4	5	1.00
WorkArena++	300	150	3	6	1.00
VitaBench	400	200	4	5	1.00

The common host offers a retrospective decision after a completed search branch, a recoverable tool error, or an explicit host retrospection request. It applies the $k_{\max}$ and $d_{\max}$ settings in Table 10 before presenting feasible actions. The controller support check requires at least 20 distinct source tasks for that action and a five-nearest-task representation distance no greater than the source 90th percentile. Failure of either condition invokes the recorded Efficient RTS fallback. These trigger, cap, support, and fallback settings are frozen from source validation.

For AutomationBench, the 600-task panel is divided into three grouped outer folds. Each run uses 400 tasks as its source pool and the remaining 200 as its held-out panel. Near-duplicate instances, common cloned states, all prefixes, and both replay arms remain within a task group. The same outer split is shared by every method and source-size condition. The aggregate contains out-of-fold predictions, not predictions on tasks used to train the corresponding controller. An instance can be a source task in another independent fold, but artifacts, memories, and thresholds cannot be transferred across those folds. No score from one fold selects another fold's controller.

Within each source pool, task roles are 80% predictor fitting, 10% margin fitting, and 10% residual calibration. Hyperparameter and manual-threshold selection use inner task-grouped validation inside the predictor-fitting portion; they do not consume the residual-calibration or outer test panel. Role assignments are made before outcomes are read and preserved as nested source prefixes grow. All roles count toward N. At $N = 100$, the allocation is 80/10/10 tasks, not 100 fitting tasks plus unreported calibration experience. Before fitting is possible, the fallback needs no source statistics.

The other benchmarks use the task-grouped outer counts in Table 10. Training never uses the scored task's group. Such within-benchmark learning is distinct from frozen cross-environment or unseen-backbone transfer; neither transfer claim follows from the main matrix. For any separate transfer analysis, encoder, heads, normalization, action semantics, support representation, margin, and λ must be frozen together and fallback rates reported.

C.3 Token costs, learning cost, and amortization

For task i and complete-method repetition r, let I_{ir}, O_{ir}, and H_{ir} be input, output, and cache-hit input tokens, summed across actor, search, permitted grading, observation adaptation, and auxiliary calls. Then $C_{ir}^{\mathrm{raw}} = I_{ir} + O_{ir}, \qquad C_{ir}^{\mathrm{uncached}} = I_{ir} - H_{ir} + O_{ir}$. Unknown cache-hit counts are missing data, not zeros. For method m and reference b, the ratios are $R_m = \frac{\sum_{i,r} C_{ir}^{\mathrm{raw},m}}{\sum_{i,r} C_{ir}^{\mathrm{raw},b}}, \qquad U_m = \frac{\sum_{i,r} C_{ir}^{\mathrm{uncached},m}}{\sum_{i,r} C_{ir}^{\mathrm{uncached},b}}$. They are ratios of totals, not averages of task ratios. The Q1 RTS denominator remains fixed across source-size checkpoints and early-stopping settings. Cache policy is shared across methods, and every branch is charged regardless of whether it is ultimately selected.

Let C_{src} include source collection and all required paired continuations, and let C_{update} include additional online replay calls. For N_{deploy} deployed tasks, learning-inclusive average raw inference cost is

$$\overline{C} = \frac{C_{\mathrm{src}} + C_{\mathrm{update}} + \sum_{i=1}^{N_{\mathrm{deploy}}} C_i^{\mathrm{deploy}}}{N_{\mathrm{deploy}}}. \tag{19}$$

With positive per-task deployment savings $\Delta\overline{C}$ and fixed learning costs, the approximate token break-even point is $(C_{\mathrm{src}} + C_{\mathrm{update}})/\Delta\overline{C}$. Update frequency changes the numerator.

C.4 Robustness to Host Rollout Scheduling

The host controls when trajectories are launched, while ForkPilot allocates retrospective search during their execution. We examine whether its efficiency benefits persist when the host changes from three parallel trajectories to sequential execution, evaluating both RTS and ForkPilot under each schedule. Parallel execution explores diverse action sequences early, potentially finding a trajectory that requires fewer corrections. Sequential execution instead launches another trajectory only after abandoning the current attempt without solving the task, avoiding unnecessary launches when one trajectory suffices. Both schedules retain retrospective search within each trajectory, but sequential execution may increase latency when several attempts are needed.

Table 11: Host-scheduling comparison with GPT-5.6 Sol at extra-high reasoning effort. S is task score (0–100); all R/U values are raw/uncached deployment-token ratios to RTS (parallel) within each benchmark. Higher S and lower R/U are better. Bold and underline indicate the best and second-best values per column.

Method	AutomationBench			τ^3-Bench		
	S	R	U	S	R	U
RTS (parallel)	71.83	1.000	1.000	<u>67.53</u>	1.000	1.000
ForkPilot (parallel)	69.00	0.764	<u>0.556</u>	64.43	<u>0.588</u>	<u>0.408</u>
RTS (sequential)	**77.00**	<u>0.680</u>	0.634	**69.59**	0.775	0.814
ForkPilot (sequential)	<u>75.00</u>	**0.679**	**0.402**	64.95	**0.505**	**0.292**

Sequential scheduling provides additional token savings. Both RTS and ForkPilot achieve higher observed scores with fewer tokens under sequential execution (Table 11). Relative to parallel ForkPilot, its sequential variant reduces raw tokens by 11.1% and 14.2%, and uncached tokens

by 27.6% and 28.5%, on AutomationBench and τ^3-Bench, respectively. Scores increase from 69.00 to 75.00 and from 64.43 to 64.95. Thus, deferring trajectory launches offers further savings beyond the default parallel configuration. Latency is not measured here; the potential delay from successive attempts remains unquantified.

FORKPILOT retains uncached-token savings under both host schedules. Against schedule-matched RTS, FORKPILOT reduces uncached tokens by 44.4%/59.2% under parallel execution and 36.5%/64.1% under sequential execution on AutomationBench/τ^3-Bench. Under the sequential schedule, raw-token usage is nearly identical to RTS on AutomationBench and 34.8% lower on τ^3-Bench, with respective score differences of -2.00 and -4.64 points. These operating points show that FORKPILOT's uncached-token efficiency benefit is not restricted to three-way parallel rollout, although the quality–cost trade-off varies with the host schedule.

C.5 SCOPE OF THE FRONTIER-MODEL COMPARISON

The main frontier-model Pareto comparison uses GPT-5.6 Sol on AutomationBench and τ^3-Bench and considers only methods with both task score and uncached inference-token totals. A reported GEPA score without comparable token accounting cannot be assigned a Pareto rank. The separate completed Opus 4.8 (MAX) AutomationBench runs are reported in Table 12 as a descriptive operating-point check. The raw and uncached ratios use full Opus RTS inference tokens as their respective denominators; source-training tokens are excluded from these deployment ratios.

Table 12: Completed Opus 4.8 (MAX) AutomationBench operating points. Scores are on the $[0, 1]$ scale. R/U are total raw/uncached inference-token ratios to full Opus RTS; lower token ratios are better. These aggregate records do not provide task-paired intervals.

Method	Score ↑	R/RTS ↓	U/RTS ↓
BoN-1	0.4233	0.295	0.195
BoN-5	0.5517	1.491	1.025
RTS	0.7083	1.000	1.000
Efficient RTS	0.6067	0.513	0.547
FORKPILOT	0.6667	0.778	0.763

At this operating point, FORKPILOT uses 22.2% fewer raw and 23.7% fewer uncached inference tokens than Opus RTS, with a 0.0416 lower task score. Its reported source-training totals are 1.541 billion raw and 0.457 billion uncached tokens and are not amortized in Table 12. No complete second-benchmark Opus comparison is available in the supplied run log, so no additional Opus Pareto claim is made. The aggregate Opus records also do not establish a noninferiority interval or a matched-cost superiority result.

D SELF-EVOLVING AND STOPPING PROTOCOLS

D.1 TRUE COLD START AND NESTED SOURCE-SIZE CURVES

Each of five Q2 source orders is drawn independently of task outcomes and shared across training regimes; three controller seeds are fit per order. Nested prefixes contain 0, 100, 200, 300, and 400 completed source tasks per outer fold. At zero, there are no trained weights, calibration residuals, source normalizers, or experience-memory entries. Every request is handled by the host fallback under the predeclared deployment cap. A hand-initialized predictor with previously calibrated thresholds would instead constitute a warm start and is excluded from the cold-start condition.

The Q2 deployment cap is held fixed across all source sizes and regimes within a backbone and uses the same host reservation rule as Q1. The full-source batch endpoint is therefore directly aligned with the corresponding Q1 FORKPILOT operating point rather than being evaluated under a second, undocumented cap. Training and replay budgets are logged separately from deployment cost.

For *batch fitting*, the predictor is initialized afresh at each nonzero N and fitted on that prefix's fitting tasks, with its margin and residual correction trained only on their assigned roles. For *online updates*, the first nonzero artifact is fitted after the first completed block; subsequent artifacts warm-

start from the preceding version and update on the expanded prefix, including retained fitting tasks. Recalibration uses all eligible completed tasks in the fixed margin and calibration roles. The source self-evolving policy used for paired targets remains the same frozen host across regimes, so different target definitions do not explain a curve difference.

Any paired replay evidence exposed to one learned regime is also available to the other at the corresponding source-size state. The comparison therefore isolates incremental fitting under an expanding source pool with matched evidence availability. Each artifact is frozen before the common held-out panel is evaluated. Held-out tasks and their feedback never enter updates, and the state schedule does not depend on their scores. The frozen cold-start control retains its initial fallback without updates.

D.2 Saturation and order sensitivity

Because the mean curves are mildly nonmonotone, define

$$N_{0.9} = \min\left\{ N > 0 : \frac{S(N) - S(0)}{\max_{N' \in \mathcal{N}} S(N') - S(0)} \geq 0.9 \right\}. \tag{20}$$

Because the displayed mean curves can reverse slightly at larger N, Eq. 20 normalizes by the best observed state on the evaluated source-size grid rather than by $S(N_{\max})$. On the mean curves, the first 90% crossings are $N = 300$ for 9B batch, $N = 300$ for 9B online, $N = 200$ for 27B batch, and $N = 300$ for 27B online. We therefore use the curves to support early, noisy saturation rather than a universal 200-task threshold. Source-order and controller-seed variation remains represented in the plotted hierarchical intervals; no monotonicity or concavity constraint is imposed.

D.3 Manual and learned stopping under the same action scope

The manual rule skips the current retrospective round when the permitted incumbent score $q_t \in [0, 1]$ reaches ρ. Otherwise it permits the host's fixed fork and depth configuration, subject to the common resource interface. It does not terminate the underlying task or remove future reconsideration opportunities. The primary settings are $\rho \in \{0.6, 0.8, 0.9, 1.0\}$. A stronger manual control selects ρ from $\{0.50, 0.55, \ldots, 1.00\}$ on inner source validation using the same quality-cost objective. The displayed selected values are 0.85 for Qwen3.5-9B and 0.75 for Qwen3.5-27B; held-out outcomes cannot choose them.

The learned-stopping-only policy is the singleton-action special case of Eqs. 1-4: $\mathcal{F}_t = \{u_{\text{fix}}\}$ and $\mathcal{P}_t = \varnothing$. It therefore executes the common fixed action if and only if $L_t(u_{\text{fix}}) > 0$ and otherwise skips the round. It cannot change fork count or depth and has no probe action. All self-evolving policies use identical host generation, permitted feedback, and restoration rules. Source-only resource-cap selection targets the common cost range while leaving the fixed fork/depth setting unchanged. Thresholds that cannot attain the target range remain identified as unmatched.

The primary contrast is learned-stopping-only versus the source-selected manual control at matched cost. Full ForkPilot adds the action-conditioned fork/depth menu and bounded probing; its additional score difference measures those allocation changes jointly. Access to history is permitted for every arm, while the scalar manual policy intentionally uses only the incumbent score. The dense validation-selected threshold is the primary hand-designed stopping control.

D.4 Same-host mechanism controls

The *current-score learned controller* is a two-layer gain-cost MLP trained on the same paired records and invoked at every opportunity, but its state contains only incumbent score, remaining budget, action descriptor, and feasibility mask. The *one-shot learned controller* uses the full prefix at the first retrospective opportunity and holds that allocation for the remainder of the task. The *frozen-prefix controller* recomputes feasibility and remaining budget but reuses the first recurrent representation. The *no-margin* controller uses the full refreshed representation and action menu with $\widehat{B}_t(u) = 0$. All variants use the same source-role assignments, optimization budget, candidate actions, self-evolving protocol, and outer test tasks. Action frequencies in Table 15 are measured at the common retrospective opportunities before fallback filtering.

For Table 16, a state-changing event class is assigned without reading downstream paired outcomes. The hindsight action is the feasible action with greatest realized paired net target under protocol Γ, with skip assigned zero. "Sign flip" records a change in the sign of the best non-skip paired target from the state immediately before to the state immediately after the event. This analysis tests whether new evidence changes the allocation target; it does not claim that hindsight actions are available online.

E ADDITIONAL RESULTS

Table 13: Stopping-rule ablation on AutomationBench. Panel (a) reports operating points; manual and learned-stopping-only policies share a singleton fork/depth action and no probe, while full FORKPILOT uses the complete action menu. Panel (b) reports simultaneous 95% hierarchical paired intervals over held-out task groups and three matched controller seeds per fold for the two predeclared comparisons within each backbone. Raw-cost equivalence requires the entire interval to lie inside the predeclared $\pm 2\%$ band.

(a) Operating points and absolute costs

Policy	Score	Raw (k)	R	Uncached (k)	U
Qwen3.5-9B; RTS raw/uncached = 142.63k/101.33k tokens per task					
Manual: $\rho = 0.6$	12.10	98.42	0.690	75.79	0.748
Manual: $\rho = 0.8$	13.10	99.56	0.698	76.10	0.751
Manual: $\rho = 0.9$	13.40	101.11	0.709	77.01	0.760
Manual: $\rho = 1.0$	13.00	99.84	0.700	76.20	0.752
Validation-selected manual ($\rho = 0.85$)	13.61	99.80	0.700	75.99	0.750
Learned stopping only	14.02	100.00	0.701	76.09	0.751
Full FORKPILOT	14.53	99.84	0.700	75.99	0.750
Qwen3.5-27B; RTS raw/uncached = 181.20k/128.40k tokens per task					
Manual: $\rho = 0.6$	23.00	117.42	0.648	89.62	0.698
Manual: $\rho = 0.8$	24.40	118.01	0.651	90.60	0.706
Manual: $\rho = 0.9$	24.00	119.59	0.660	91.16	0.710
Manual: $\rho = 1.0$	23.80	118.08	0.652	90.95	0.708
Validation-selected manual ($\rho = 0.75$)	24.58	117.72	0.650	89.88	0.700
Learned stopping only	25.01	117.94	0.651	90.01	0.701
Full FORKPILOT	25.61	117.78	0.650	89.88	0.700

(b) Paired contrasts to the validation-selected manual control

Backbone	Policy	ΔS [simult. 95% CI]	Δ raw cost (%) [simult. 95% CI]	$\pm 2\%$ equiv.
Qwen3.5-9B	Learned stopping only	+0.41 [+0.07, +0.74]	+0.21 [-0.72, +1.13]	Yes
	Full FORKPILOT	+0.92 [+0.55, +1.31]	+0.03 [-0.84, +0.93]	Yes
Qwen3.5-27B	Learned stopping only	+0.43 [+0.06, +0.79]	+0.18 [-0.71, +1.05]	Yes
	Full FORKPILOT	+1.03 [+0.63, +1.44]	+0.04 [-0.76, +0.86]	Yes

Table 14: Paired inference for the four predeclared mechanism contrasts on AutomationBench. Effects are full FORKPILOT minus the named ablation. Intervals are simultaneous 95% hierarchical max-$|t|$ bootstrap intervals over held-out task groups and three matched controller seeds per fold within each backbone; positive ΔS and negative ΔR favor full FORKPILOT. Fold signs are descriptive.

Backbone	Ablation comparator	ΔS simult. 95% CI	ΔR simult. 95% CI	Positive folds
Qwen3.5-9B	Current-score learned controller	+0.91 [+0.31, +1.50]	-0.013 [-0.022, -0.005]	3/3
	One-shot learned controller	+0.79 [+0.21, +1.36]	-0.009 [-0.018, -0.001]	3/3
	Frozen-prefix controller	+0.62 [+0.11, +1.14]	-0.006 [-0.014, +0.001]	3/3
	FORKPILOT without margin	+0.41 [+0.04, +0.79]	-0.081 [-0.095, -0.069]	3/3
Qwen3.5-27B	Current-score learned controller	+1.12 [+0.44, +1.79]	-0.016 [-0.026, -0.007]	3/3
	One-shot learned controller	+0.98 [+0.32, +1.65]	-0.014 [-0.024, -0.005]	3/3
	Frozen-prefix controller	+0.71 [+0.13, +1.30]	-0.008 [-0.017, -0.001]	3/3
	FORKPILOT without margin	+0.40 [+0.02, +0.80]	-0.089 [-0.105, -0.075]	3/3

F AI STATEMENT

During revision, LLMs assisted with drafting and language editing. The authors reviewed, verified, and finalized the mathematical claims, proofs, methodology, experimental design, data processing,

Table 15: Mechanism ablation on AutomationBench. Every learned variant uses the same host, feasible actions, paired source records, held-out tasks, and three matched controller seeds per outer fold. Full/probe/skip are percentages of retrospective opportunities and sum to 100 within a row.

Backbone	Controller	Score	R	Full	Probe	Skip
Qwen3.5-9B	Current-score learned controller	13.62	0.713	26	13	61
	One-shot learned controller	13.74	0.709	25	12	63
	Frozen-prefix controller	13.91	0.706	24	15	61
	FORKPILOT without margin	14.12	0.781	37	18	45
	Full FORKPILOT	14.53	0.700	23	19	58
Qwen3.5-27B	Current-score learned controller	24.49	0.666	27	12	61
	One-shot learned controller	24.63	0.664	26	12	62
	Frozen-prefix controller	24.90	0.658	25	15	60
	FORKPILOT without margin	25.21	0.739	38	17	45
	Full FORKPILOT	25.61	0.650	23	20	57

Table 16: Event-conditioned evidence that search value changes after tool interaction. A sign flip compares the paired net value immediately before and after the new event. Hindsight action agreement uses the best action under completed paired continuations; utility is normalized paired net value of refreshed over frozen-prefix decisions.

Event class	n	Flip (%)	Fresh (%)	Frozen (%)	Δ utility
Recoverable error followed by new identifier	1,840	41.7	72.6	55.8	+0.071
Permission or schema clarification	1,310	36.9	70.8	56.4	+0.062
Routine success or unchanged observation	2,790	11.8	73.1	69.4	+0.014
All audited transitions	5,940	26.8	72.4	62.7	+0.041

Table 17: Identifier-masking control on AutomationBench. "Fresh, IDs masked" recomputes the trajectory representation after each tool event but replaces newly revealed literal identifiers with typed placeholders while preserving event type, schema, success/error status, and value length.

Backbone	Representation	Score	R	Fraction of refresh gain retained
Qwen3.5-9B	Frozen prefix	13.91	0.706	0%
	Fresh, IDs masked	14.32	0.703	66.1%
	Fresh (full FORKPILOT)	14.53	0.700	100%
Qwen3.5-27B	Frozen prefix	24.90	0.658	0%
	Fresh, IDs masked	25.37	0.654	66.2%
	Fresh (full FORKPILOT)	25.61	0.650	100%

reported numerical results, citations, and conclusions, and take responsibility for the final content of the work.

G ETHICS STATEMENT

This study is based exclusively on publicly available datasets and large language models, and does not involve the use of private data. All scientific concepts, methodological designs, experimental implementations, and resulting conclusions remain entirely the responsibility of the authors.

Table 18: Cross-benchmark replication on WorkArena++ with Qwen3.5-27B. The same source-only selection, grouped outer split, and information boundary are used as in AutomationBench.

Question	Replication result	Outcome
Q2: 25% source pool	Batch/online recover 64.3%/60.8% of the full-source score gain; first 90% crossing is at 75%/75% of the source pool.	Early saturation
Q3: learned stopping	$\Delta S = +0.37\,[+0.08, +0.69]$ vs. validation-selected manual; Δ raw cost $= +0.26\%\,[-0.61, +1.12]$.	Within $\pm 2\%$ cost band
Q3: refreshed state	Full vs. frozen: $\Delta S = +0.56\,[+0.13, +1.02]$, $\Delta R = -0.011\,[-0.019, -0.004]$.	Refresh gain repeats
Q3: operational margin	Full vs. no-margin: $\Delta S = +0.35\,[+0.06, +0.67]$, $\Delta R = -0.078\,[-0.091, -0.066]$.	Margin remains main efficiency effect
Q4: boundary audit	4,860 controller/hint counterfactuals and 960 protected-field canaries; 0 violations; 180/180 injected positives detected.	Same audited boundary

Table 19: Detailed black-box evaluation on frontier proprietary backbones. S denotes task score on a 0-100 scale, and ΔS is the score difference relative to full RTS under the same backbone-benchmark setting. Raw and Uncached report inference-token totals in millions. R/U are the corresponding token ratios to full RTS, and Raw/U Save report percentage reductions relative to RTS; negative savings indicate higher cost than RTS. For GHCP, S uses the strict task-success metric, consistent with the task-score definition used throughout the paper. Bold and underline denote the best and second-best task scores within each backbone-benchmark setting.

Backbone	Benchmark	Method	$S\uparrow$	ΔS	Raw (M)↓	Uncached (M)↓	$R\downarrow$	$U\downarrow$	Raw Save	U Save
GPT-5.6 Sol / xhigh	Automation	BoN-5	59.83	-12.00	1101.62	633.46	2.030	1.689	-103.0%	-68.9%
		RDS	68.83	-3.00	885.48	261.18	1.632	0.697	-63.2%	30.3%
		GHCP	33.50	-38.33	458.98	180.55	0.846	0.481	15.4%	51.9%
		RTS	**71.83**	0.00	542.72	374.98	1.000	1.000	0.0%	0.0%
		ForkPilot	<u>69.00</u>	-2.83	414.87	208.53	0.764	0.556	23.6%	44.4%
	τ^3-Bench	BoN-5	59.28	-8.25	704.56	290.99	0.989	1.138	1.1%	-13.8%
		RDS	55.67	-11.86	1982.05	198.15	2.783	0.775	-178.3%	22.5%
		GHCP	43.30	-24.23	147.96	47.63	0.208	0.186	79.2%	81.4%
		RTS	**67.53**	0.00	712.12	255.62	1.000	1.000	0.0%	0.0%
		ForkPilot	<u>64.43</u>	-3.10	418.92	104.40	0.588	0.408	41.2%	59.2%
Opus 4.8 / max	Automation	BoN-5	55.17	-15.66	1327.83	261.14	1.491	1.025	-49.1%	-2.5%
		RDS	62.50	-8.33	2838.48	303.76	3.188	1.192	-218.8%	-19.2%
		GHCP	39.83	-31.00	433.35	48.05	0.487	0.189	51.3%	81.1%
		RTS	**70.83**	0.00	890.29	254.76	1.000	1.000	0.0%	0.0%
		ForkPilot	<u>66.67</u>	-4.16	693.01	194.46	0.778	0.763	22.2%	23.7%
	τ^3-Bench	BoN-5	–	–	–	–	–	–	–	–
		RDS	29.90	-6.18	2724.05	27.52	1.685	0.454	-68.5%	54.6%
		GHCP	**41.24**	+5.16	276.56	15.18	0.171	0.251	82.9%	74.9%
		RTS	<u>36.08</u>	0.00	1616.57	60.56	1.000	1.000	0.0%	0.0%
		ForkPilot	33.51	-2.57	987.21	34.47	0.611	0.569	38.9%	43.1%

Table 20: Black-box comparison on frontier proprietary backbones. S is task score on a 0–100 scale; R/U denote raw/uncached inference-token ratios relative to full RTS under the same backbone–benchmark setting. Lower R/U is better. For GHCP and GEPA, S uses the reported Strict metric. GHCP denotes the production GitHub Copilot harness; RDS denotes depth-only retrospective search.

	GPT-5.6 Sol						Opus 4.8					
	Automation			τ^3-Bench			Automation			τ^3-Bench		
Method	S	R	U	S	R	U	S	R	U	S	R	U
BoN-1	43.67	0.363	0.288	44.33	0.192	0.200	42.33	0.295	0.195	20.10	0.328	0.223
BoN-5	59.83	2.030	1.689	59.28	0.989	1.138	55.17	1.491	1.025	–	–	–
Efficient RTS	60.17	0.555	0.530	**67.53**	0.711	0.738	60.67	0.513	0.547	32.47	0.761	0.683
RDS	68.83	1.632	0.697	55.67	2.783	0.775	62.50	3.188	1.192	29.90	1.685	0.454
GHCP	33.50	0.846	0.481	43.30	0.208	0.186	39.83	0.487	0.189	**41.24**	0.171	0.251
Trace2Skill	40.83	0.678	0.434	46.39	0.232	0.148	45.00	0.365	0.298	27.84	0.308	0.225
ReasoningBank	42.00	0.523	0.370	48.45	0.178	0.136	42.17	0.316	0.267	27.08	0.216	0.143
GEPA	46.50	0.480	0.329	46.39	0.230	0.238	46.83	0.280	0.202	25.77	0.370	0.242
RTS	**71.83**	1.000	1.000	**67.53**	1.000	1.000	**70.83**	1.000	1.000	36.08	1.000	1.000
FORKPILOT	69.00	0.764	0.556	64.43	0.588	0.408	66.67	0.778	0.763	33.51	0.611	0.569