PhGPO: Pheromone-Guided Policy Optimization for Long-Horizon Tool Planning

Yu Li[1], Guangfeng Cai[1], Shengtian Yang[1], Han Luo[1], Shuo Han[2], Xu He[2], Dong Li[2], Lei Feng[1]*

[1]School of Computer Science and Engineering
Southeast University, Nanjing, China
[2]Huawei Noah's Ark Lab, China
{yuli11,fenglei}@seu.edu.cn

Abstract

Recent advancements in Large Language Model (LLM) agents have demonstrated strong capabilities in executing complex tasks through tool use. However, long-horizon multi-step tool planning is challenging, because the exploration space suffers from a combinatorial explosion. In this scenario, even when a correct tool-use path is found, it is usually considered an immediate reward for current training, which would not provide any reusable information for subsequent training. In this paper, we argue that *historically successful trajectories contain reusable tool-transition patterns*, which can be leveraged throughout the whole training process. Inspired by ant colony optimization where historically successful paths can be reflected by the *pheromone*, we propose Pheromone-Guided Policy Optimization (PhGPO), which learns a trajectory-based transition pattern (i.e., pheromone) from historical trajectories and then uses the learned pheromone to guide policy optimization. This learned pheromone provides explicit and reusable guidance that steers policy optimization toward historically successful tool transitions, thereby improving long-horizon tool planning.

1 Introduction

Large Language Model (LLM) agents have demonstrated strong capabilities in executing complex tasks through tool use [1–5]. By invoking external tools such as web search [6, 7], computer operation [8], and code execution [9], agents can access up-to-date information and take actions beyond static parametric knowledge. As the tool space expands, many real-world tasks require a tool-use trajectory, where early choices constrain later reachable states and determine whether the entire trajectory succeeds [10, 11]. When tool-use trajectories extend to ten to twenty steps, as in Toolathlon [12], long-horizon tool planning becomes difficult: agents must choose from hundreds of tools, yet even when a successful trajectory is discovered, its transition patterns are hard to explicitly store and reliably reuse, which hinders effective learning in such a large tool space [13, 14].

Reinforcement learning (RL) offers a natural way to leverage interaction experiences and feedback to update the policy for long-horizon tool planning [15–17]. However, interaction experiences are typically absorbed implicitly into the policy parameters, leaving no explicit record of historically successful tool transitions. As a result, even after a successful tool-use trajectory is discovered, its transition patterns remain implicit in the policy parameters and cannot be directly reused as guiding information. When successful trajectories are rare, the exploration of RL becomes inefficient because the agent must search again for similar long-horizon trajectories [18, 19].

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

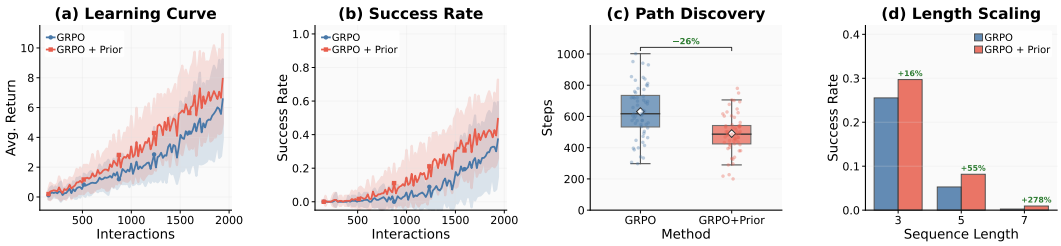

Figure 1: **Two-stage pilot experiment with an explicit transition prior for long-horizon tool planning.** We distill cross-trajectory transition memory from verified successful tool-use trajectories in an easier stage, then reuse it as a prior for GRPO optimization. Panels (a,b) show higher average return and success rate under matched interaction budgets, (c) fewer steps to the first successful tool-use trajectory, and (d) larger relative gains with longer trajectories.

In this paper, we argue that *an explicit transition prior* can be distilled from historically successful trajectories, which would be helpful to policy optimization. We validate our viewpoint with a two-stage pilot experiment (Figure 1): Firstly, we collect verified successful tool-use trajectories and distill their transition patterns into an explicit transition prior. Secondly, we use this prior to reweight the policy's next-tool distribution in accord with verified transitions. As shown in Figure 1, the explicit transition prior improves both sample efficiency and final success, with larger gains as trajectory length increases.

Motivated by the above analysis, we would like to seek an explicit transition prior that can be learned from historical trajectories and reused during policy optimization. To realize such a prior, we draw inspiration from Ant Colony Optimization (ACO), where the *pheromone* summarizes historically successful paths and encourages them to be taken again [20, 21]. Based on this inspiration, we propose **Pheromone-Guided Policy Optimization (PhGPO)**, which learns an explicit transition prior from historical trajectories, and this transition prior acts as the pheromone in ACO. Concretely, PhGPO organizes tools into a tool-transition graph based on the Model Context Protocol (MCP) [2] and associates each transition with a pheromone value. During policy optimization, we integrate the pheromone values into next-step tool selection, making historically successful tool transitions more likely to be reproduced. It is noteworthy that the pheromone is reinforced by verified successful tool-use trajectories and decays over time, which keeps the transition pattern reusable and up to date. We further construct a set of frequent argument invocations for each tool, enabling the agents to learn not only which tool to call but also how to invoke it effectively.

Based on the pheromone mechanism as described above, our PhGPO adopts a progressive training pipeline to equip an agent with long-horizon tool planning abilities. Firstly, PhGPO begins with supervised warm-up using a next-tool prediction objective to provide a stable initialization. Secondly, PhGPO performs pheromone-guided reinforcement learning with a progressive schedule: it injects the correct next tool with a decaying probability and progressively cedes control to the policy, which enables pheromone to be continually updated from accumulated policy-driven successes. As the accumulated pheromone becomes more reliable, it fully leverages pheromone guidance for trajectory generation to optimize long-horizon performance. We empirically evaluate PhGPO on three long-horizon tool-planning benchmarks, including Toolathlon [12], TOUCAN [13] and TRAJECT-Bench [14], which cover diverse real-world tool-use scenarios with long multi-step tool-use trajectories. Across these benchmarks, PhGPO consistently produces tool-use trajectories that more closely match the reference trajectories on complex tasks, demonstrating that pheromone enables effective reuse of historical successes and improves long-horizon tool planning performance. Our main contributions can be summarized as follows:

- **Explicit reuse in long-horizon tool planning.** We show that transition patterns in successful tool-use trajectories are typically absorbed implicitly into the policy parameters, making them hard to reuse and slowing learning when verified successes are rare.
- **PhGPO with pheromone as an explicit transition prior.** We propose PhGPO, which distills successful trajectories into an explicit transition prior working as pheromone, and uses it to guide policy optimization toward historically successful tool transitions.
- **MCP-grounded graph and progressive training.** We build an MCP-grounded tool-transition graph to maintain pheromone and introduce a progressive training pipeline to accumulate and

exploit it over long tool-use trajectories, achieving consistent gains on Toolathlon, TOUCAN, and TRAJECT-Bench.

2 Related Work

Retrieval-augmented tool use. Tool selection at scale is a central challenge for tool-using agents. Many methods use retrieval to narrow the candidate set by fetching relevant documentation, schemas, or demonstrations [10, 22–24]. Early prompting-based agents interleave reasoning and action selection over retrieved evidence [10, 22], while recent systems further train retrievers or couple retrieval with tool-calling models to improve robustness under large and evolving tool pools [25, 3, 26–28]. Dedicated benchmarks and pipelines also treat tool selection as an information retrieval problem [29–32]. While retrieval improves per-step grounding, long-horizon tool-use trajectories still require consistent transition patterns across multiple decisions.

Graph-structured tool planning and search. Beyond retrieval, many methods organize APIs and dependencies into graphs or trees, and then perform graph traversal, heuristic search, or learned graph-based planning [33–37]. Some approaches search or expand over tool graphs to produce multi-step tool-use trajectories [33–35], while others generate executable tool graphs or DAG-like plans [36]. Interactive search and self-updating mechanisms can also improve robustness to evolving tool interfaces [38]. These structured methods reduce the effective search space, but they usually rely on heuristic or supervised planning signals and do not maintain an adaptive transition prior distilled from accumulated successful trajectories.

Reinforcement learning for tool use. Reinforcement learning has been increasingly used to improve multi-step tool use under sparse outcomes [15, 16]. Recent work studies tool-integrated optimization with verifiable or execution-based rewards [39], hierarchical agent designs that separate high-level planning from tool execution [40], and executable-code agents that learn tool-use strategies from outcomes [41, 42]. Despite these advances, successful experience is often absorbed implicitly into policy parameters, which can slow learning when verified successful trajectories are rare.

Experience reuse and agent memory. Explicit experience reuse stores and replays successful behaviors instead of relying only on implicit parameter updates [43, 44]. Record-and-replay paradigms cache trajectories or routines to accelerate future problem solving [45, 46]. However, these methods rarely convert accumulated successes into structured transition statistics that can serve as an explicit transition prior for long-horizon tool-use trajectories.

Ant-inspired reinforcement learning and pheromone-based search. Prior ant-inspired methods use pheromone mechanisms for search, routing, multi-agent coordination, and optimal control [47–52], and recent LLM work applies similar signals to inference-time reasoning-path search [53]. PhGPO differs by treating pheromone as an explicit transition prior for long-horizon tool-use policy optimization, not as a stand-alone optimizer or value-like controller. This prior is distilled from verified successful tool-use trajectories and used as a log-space prior during rollout generation, while policy learning remains governed by GRPO.

3 Methodology

In this section, we present Pheromone-Guided Policy Optimization (**PhGPO**) for long-horizon tool planning. PhGPO is built around a simple principle: long-horizon tool use should explicitly *accumulate* and *reuse* historically successful tool transitions from verified tool-use trajectories. Inspired by *Ant Colony Optimization (ACO)* [20], PhGPO instantiates the explicit transition prior as pheromone on a tool-transition graph. Pheromone is reinforced by verified successful trajectories and decays over time, yielding an adaptive supervision signal that can be incorporated into next-step tool selection for policy optimization. In this way, PhGPO converts trajectory-level success into transition-level pheromone on the graph, which steers policy optimization toward historically successful tool transitions while avoiding stale bias through decay.

3.1 Tool-Transition Graph from MCP

The goal of long-horizon tool use is to first select the next tool and then specify an executable invocation. To this end, we build a tool-transition graph from MCP that captures tool transitions from

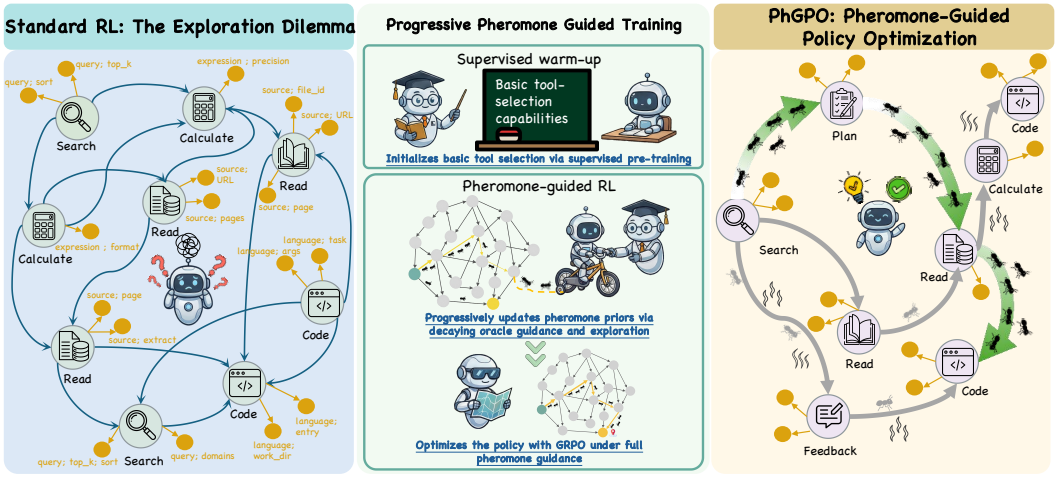

Figure 2: **Overview of PhGPO.** PhGPO converts verified successful tool-use trajectories into a pheromone-based explicit transition prior over tool-transition and tool-to-invocation edges. The prior is updated through deposition and evaporation, and incorporated into rollout generation to guide the policy toward historically successful transitions and argument invocations. Training proceeds from supervised next-tool warm-up to progressive pheromone-guided reinforcement learning and finally full pheromone-guided policy optimization.

verified trajectories and explicitly represents tools and their transitions, providing a structured space for modeling multi-step tool-use trajectories and their transition patterns. We also construct a set of frequent argument invocations for each tool, which captures how tool arguments are frequently invoked.

Tool-transition graph. Let $\mathcal{V}_{\text{Tool}} = \{v_1, \ldots, v_N\}$ be the base tools exposed by MCP. We view a tool-use trajectory as a path over $\mathcal{V}_{\text{Tool}}$ and define a directed edge $(v_i, v_j) \in \mathcal{E}$ when v_j follows v_i in a trajectory. This graph captures tool-to-tool dependencies and provides the substrate for accumulating transition experience on edges.

Argument-invocation set. The invocation of each tool $v_i \in \mathcal{V}_{\text{Tool}}$ also depends on the specific structure of tool arguments. For each tool $v_i \in \mathcal{V}_{\text{Tool}}$, we construct a set of frequent executable argument invocations $\mathcal{V}_{\text{Arg}}$ that captures how tool arguments are frequently invoked, denoted by $\mathcal{V}_{\text{Arg}}(v_i) = \{v_i^{(1)}, \ldots, v_i^{(K_i)}\}$ where $v_i^{(j)}$ denotes the j-th argument-invocation pattern among K_i key patterns of the tool v_i. To make the mapping explicit, we define tool-to-invocation association edges $\mathcal{E}_{\text{arg}} = \bigcup_{i=1}^{N} \{(v_i, v_i^{(k)}) \mid v_i^{(k)} \in \mathcal{V}_{\text{Arg}}\}$. An edge $(v_i, v_i^{(k)}) \in \mathcal{E}_{\text{arg}}$ indicates that $v_i^{(k)}$ is an argument-invocation associated with tool v_i.

Based on both the tool-transition graph and the argument-invocation set, we can have a factorized decision at step t: the agent first selects the next tool $a_t \in \mathcal{V}_{\text{Tool}}$, then selects an argument invocation $u_t \in \mathcal{V}_{\text{Arg}}(a_t)$ to instantiate an executable invocation template.

3.2 Pheromone as an Explicit Transition Prior

Our PhGPO integrates both *task-agnostic* pheromones with *task-dependent* pheromones from verified successful tool-use trajectories.

Task-agnostic pheromones. We maintain two task-agnostic pheromones, including $\tau^{\text{agn}}_{\text{Tool}}(v_i, v_j)$ over tool-transition edges and $\tau^{\text{agn}}_{\text{Arg}}(v_i, v_i^{(k)})$ over tool-to-invocation edges. After executing a trajectory ξ, we compute a bounded pheromone-deposition score $q_\tau(\xi) \in [0, 1]$, measured by the match ratio to the reference tool-use trajectory. This score is used only to determine how much pheromone is deposited on the traversed edges. It is not used as a direct step-level reward and does not directly enter the policy optimization loss. For each traversed tool transition and each selected argument invocation in ξ, we apply ACO-style deposition and evaporation:

$$\tau^{\text{agn}} \leftarrow \text{clip}\big((1-\rho)\tau^{\text{agn}} + \alpha\, q_\tau(\xi),\ [\tau_{\min}, \tau_{\max}]\big). \tag{1}$$

Here, $\rho \in (0,1)$ is the evaporation rate, $\alpha > 0$ is the deposition rate, and $q_\tau(\xi)$ scales the amount of pheromone deposited by trajectory ξ. Higher-quality tool-use trajectories contribute stronger pheromone updates, while evaporation reduces the influence of older statistics. We clip pheromones to $[\tau_{\min}, \tau_{\max}]$ to prevent excessive concentration on a small subset of transitions. It is noteworthy that the construction of both $\tau^{\text{agn}}_{\text{Tool}}$ and $\tau^{\text{agn}}_{\text{Arg}}$ follows the same way as shown in Eq. (1).

Task-dependent pheromones. For each transition (v_i, v_j), we maintain an edge memory bank $\mathcal{M}_{ij}$ that stores tuples $(\boldsymbol{e}, q_\tau)$, where $\boldsymbol{e}$ is a task embedding and q_τ is the pheromone-deposition score of a verified successful tool-use trajectory that includes this transition. Given a task description x, we compute $\boldsymbol{e}_x = \phi(x)$ using a sentence encoder ϕ and retrieve memories whose cosine similarity exceeds a threshold:

$$\mathcal{M}_{ij}(x) = \{(\boldsymbol{e}, q_\tau) \in \mathcal{M}_{ij} \mid \text{sim}(\boldsymbol{e}_x, \boldsymbol{e}) \geq \theta_{\text{sim}}, \forall \boldsymbol{e}\}. \tag{2}$$

Then, we estimate task-dependent pheromone by similarity-weighted averaging:

$$\tau^{\text{dep}}(v_i, v_j \mid x) = \tau_0 + \frac{\sum_{(\boldsymbol{e},q_\tau)\in\mathcal{M}_{ij}(x)} \text{sim}(\boldsymbol{e}_x, \boldsymbol{e})\, q_\tau}{\sum_{(\boldsymbol{e},q_\tau)\in\mathcal{M}_{ij}(x)} \text{sim}(\boldsymbol{e}_x, \boldsymbol{e}) + \epsilon_{\text{dep}}} \cdot (\tau_{\max} - \tau_0). \tag{3}$$

where τ_0 is a base value, ϵ_{dep} is a small constant for numerical stability, and the scaling $(\tau_{\max} - \tau_0)$ keeps τ^{dep} in a comparable range. It is noteworthy that the construction of both $\tau^{\text{dep}}_{\text{Tool}}(v_i, v_j \mid x)$ and $\tau^{\text{dep}}_{\text{Arg}}(v_i, v_i^{(k)} \mid x)$ follows the same way as shown in Eq. (3).

Nevertheless, as shown in Eq. (3), if the edge memory bank $\mathcal{M}_{ij}$ contains very few memories $(\boldsymbol{e}, q_\tau)$, the estimated task-dependent pheromone could be unreliable. Therefore, we need to assign a corresponding confidence $c_{ij}(x) \in [0,1]$ to each task-dependent pheromone for representing their reliability:

$$c_{ij}(x) = \min(1, n_{\text{act}}/n_{\min}) \cdot s_{\max} \cdot \bar{q}_\tau, \tag{4}$$

where n_{act} is the number of memories in $\mathcal{M}_{ij}$, $n_{\min}$ is a pre-defined minimum number of elements in $\mathcal{M}_{ij}$, $s_{\max}$ is the maximum similarity observed, and $\bar{q}_\tau$ is the mean pheromone-deposition score of the retrieved memories. In this way, the final pheromone can be fused by simultaneously considering task-agnostic pheromones and task-dependent pheromones:

$$\tau_{\text{Tool}}(v_i, v_j \mid x) = (1 - w \cdot c_{ij}(x)) \cdot \tau^{\text{agn}}_{\text{Tool}}(v_i, v_j) + w \cdot c_{ij}(x) \cdot \tau^{\text{dep}}_{\text{Tool}}(v_i, v_j \mid x). \tag{5}$$

where w controls the maximum contribution of task dependence. We apply the same fusion way as shown in Eq. (5) for calculating τ_{Arg}.

3.3 Pheromone-Guided Policy Optimization Training

Learning long-horizon tool planning from scratch is difficult because early training lacks reliable successes to build an explicit transition prior. We adopt a progressive pipeline that stabilizes basic behavior, gradually accumulates pheromone from verified tool-use trajectories, and then performs pheromone-guided policy optimization.

Supervised warm-up. We first pre-train the policy with a next-tool prediction objective:

$$\mathcal{L}^{\text{SL}} = -\log \pi_\theta(a_t^* \mid s_t), \tag{6}$$

where a_t^* is the reference next tool. This stage ignores pheromone guidance and moves the policy away from random tool selection. By establishing tool-selection capabilities, the model can generate plausible initial tool-use trajectories that facilitate future success.

Pheromone as guidance. Building on this initialization, we introduce a pheromone-guided exploration strategy to navigate long-horizon tool-use task completion. Specifically, we incorporate pheromone as an explicit transition prior during trajectory generation, increasing the likelihood of collecting informative tool-use trajectories for policy learning. This pheromone-guided trajectory generation aligns well with group-based policy optimization [54, 55], where learning signals depend on the quality of multiple rollouts per prompt. We adopt Group Relative Policy Optimization (GRPO) [56] as the optimization backbone, since GRPO computes relative advantages within a rollout group. In long-horizon tool planning with a large tool space, unguided sampling can yield groups dominated

by uniformly low-quality trajectories, weakening the relative learning signal. Pheromone-guided sampling increases the likelihood that each group contains higher-potential trajectories and meaningful contrasts, enabling GRPO to better leverage its group-based advantage to update the policy. We now present the pheromone-guided sampling rule and the objective used to update the policy.

Pheromone-guided sampling. At each step t, the policy network generates $\pi_\theta(a_t \mid s_t)$ given the current state s_t. To incorporate historical experience, we integrate the policy with pheromone as an explicit transition prior. Specifically, we first select the top-K candidates from π_θ as the sampling support, and then sample the next tool according to the pheromone-guided policy:

$$p(a_t \mid s_t, a_{t-1}) \propto \exp\big(\log \pi_\theta(a_t \mid s_t) + \beta \log \tau_{\text{Tool}}(a_{t-1}, a_t \mid x)\big). \tag{7}$$

Once the base tool a_t is selected, the agent selects an argument invocation $u_t \in \mathcal{V}_{\text{Arg}}(a_t)$ according to $\tau_{\text{Arg}}(a_t, u_t \mid x)$. This argument choice is not parameterized by π_θ and is not updated during policy optimization. Here (a_t, u_t) follows the tool-to-invocation association defined in Sec. 3.1.

Policy optimization. For each training instance, we sample M tool-use trajectories using the pheromone-guided distribution in Eq. (7), obtaining returns $\{R_m\}_{m=1}^{M}$. We compute group-relative advantages:

$$A_m = \frac{R_m - \bar{R}}{\sigma_R + \epsilon_{\text{adv}}}, \qquad \text{where} \quad \bar{R} = \frac{1}{M}\sum_{m=1}^{M} R_m, \tag{8}$$

where ϵ_{adv} is a small constant for numerical stability. The policy π_θ is updated by minimizing the following objective:

$$\mathcal{L}^{\text{PG}} = -\mathbb{E}\left[\min\left(r_t(\theta)A_m,\ \text{clip}(r_t(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})A_m\right)\right],$$

where $r_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ denotes the probability ratio at step t, and ϵ_{clip} is the clipping threshold.

Progressive pheromone updates. Having introduced pheromone-guided sampling and the policy optimization objective, we apply them through a progressive training schedule that blends supervision and exploration to accumulate pheromone as an explicit transition prior. We introduce a learning curriculum over horizon lengths, starting from short horizons and gradually increasing to longer ones, and employ an oracle probability p_{tf} decaying over time. Specifically, at each step t, the action a_t is determined as:

$$a_t = \begin{cases} a_t^*, & \text{with prob. } p_{\text{tf}}, \\ \text{sampled from Eq. (7)}, & \text{with prob. } 1 - p_{\text{tf}}, \end{cases} \tag{9}$$

where a_t^* is the reference action. This keeps rollouts close to reference trajectories in the early phase, increasing the chance of collecting verified successful tool-use trajectories for pheromone updates in longer horizons. We employ a mixed objective that shifts from imitation learning to policy optimization:

$$\mathcal{L}^{\text{mixed}} = \lambda \mathcal{L}^{\text{SL}} + (1-\lambda)\mathcal{L}^{\text{PG}} - \gamma \mathcal{H}[\pi_\theta], \tag{10}$$

where λ is annealed downwards and $\mathcal{H}[\pi_\theta]$ denotes the entropy of π_θ. Accordingly, the explicit transition prior (pheromone) is updated from partially autonomous verified successful trajectories, making it increasingly reflect policy-selected transitions, instead of being dominated by purely teacher-forced ones. We concurrently increase the pheromone-controlling hyperparameter β in Eq. (7) and the task-dependence weight w in Eq. (5) as the number of accumulated trajectories grows, turning the initial pheromone values into a reliable transition signal.

Full-pheromone rollouts. As p_{tf} decays and λ is annealed downwards, training transitions to fully autonomous rollouts with full pheromone influence:

$$\tilde{\pi}(a \mid s_t) \propto \exp\big(\log \pi_\theta(a \mid s_t) + \beta_{\max} \log \tau_{a_{t-1},a}\big), \tag{11}$$

where $\beta_{\max}$ is the maximum pheromone influence scheduled to be reached, and $\tau_{a_{t-1},a}$ is a shorthand for the fused tool-transition pheromone $\tau_{\text{Tool}}(a_{t-1}, a_t \mid x)$. Similarly, argument-invocation u_t is selected using the same pheromone-guided rule with $\tau_{\text{Arg}}(a_t, u_t \mid x)$. The updated pheromone serves as an explicit transition prior during trajectory generation, encouraging the policy to reproduce historically successful transitions and argument invocations. Policy optimization is performed on tool-use trajectories collected under pheromone guidance. In parallel, pheromone updates follow

the deposition and evaporation rule defined in Eq. (1), where evaporation reduces the influence of outdated statistics over time.

Collectively, training begins with supervised warm-up that provides a stable initialization for next-tool selection. It then proceeds with pheromone-guided reinforcement learning that uses pheromone as an explicit transition prior during trajectory generation and optimizes the policy with GRPO on tool-use trajectories collected under pheromone guidance. The training schedule gradually reduces oracle guidance via the decayed p_{tf} and anneals λ downward, and the horizon curriculum increases task difficulty from short to long. This progression supports progressive pheromone accumulation from partially guided verified successes and later enables fully pheromone-guided trajectory generation for complex long-horizon tool-use trajectories.

4 Experiments

4.1 Experiment Setup

Dataset. We conduct experiments on three long-horizon tool-use benchmarks: **Toolathlon** [12], a benchmark built on MCP servers with hundreds of tools across diverse domains; **TRAJECT-Bench** [14], grounded in executable production-style APIs; and **TOUCAN** [13], a large-scale tool-agent dataset synthesized from MCP environments. Each instance is converted into an episode containing a task description, a reference tool-use trajectory, and the corresponding execution outputs. We filter episodes with verified successful trajectories and map each recorded tool call to our action representation by separating the base tool from its argument invocation. We construct fixed-seed training, validation, and test splits and report results on the test set.

Comparing Methods. We compare with a diverse set of representative tool-use baselines. **Standard Prompting and Planning** methods include the reasoning-action interleaving ReAct [10] and global-oriented Plan-and-Solve [57], alongside the Beyond-ReAct [58] planner-executor architecture. **Retrieval-Augmented Selection** baselines comprise Gorilla [25], ToolReAGt [30], and the GCN-based Tool Graph Retriever [59]. **RL-based Optimization** baselines consist of search-driven TreeRL [17], outcome-oriented ToRL [16], and process-level StepTool [15]. Finally, **Graph-based Frameworks** include GAP [60] as well as GTool [36], NaviAgent [35], and ToolNet [33], which leverage explicit tool-dependency structures for navigation or representation learning. All experiments are conducted using **Qwen2.5-7B** [61] and **Llama3.1-8B** [62] as backbones. All baselines operate in the same tool-and-invocation decision space and share the same execution interface.

Evaluation Metrics. We evaluate performance using two metrics: (1) **Match Ratio (**$M \in [0, 1]$**)**, a sequence-level metric that measures the similarity between generated tool-use trajectories and reference trajectories in tool identity and ordering; and (2) **Task Success Rate (**$\text{TSR} \in [0, 1]$**)**, an execution-level metric that measures whether the generated tool-use trajectory successfully completes the target task. We evaluate TSR using GPT-4o as an outcome judge under a fixed evaluation protocol, where the judge determines whether the final tool-use outcome satisfies the task requirement. Unlike Match Ratio, TSR evaluates the final task outcome and does not require the generated trajectory to exactly follow the reference ordering. We report both metrics as percentages in the experiments.

Implementation Details. We fine-tune the backbones using LoRA [63] ($r = 64$, $\alpha = 128$) to adapt the policy for long-horizon tool planning. Each state consists of the task description and a fixed-length truncated history of recent tool interactions. For decoding, we employ temperature 0.7, top-K sampling with $K = 20$, and random exploration probability $p_{\text{rand}} = 0.05$. The pheromone influence β is linearly scheduled to a maximum value $\beta_{\max} = 0.8$. We set the maximum trajectory length to 20 and sample $M = 5$ rollouts for optimization. Complete experimental details are provided in Appendix B.

4.2 Experimental Results

Overall performance compared with prior methods. Table 1 shows that **PhGPO achieves the best performance** on Toolathlon, TRAJECT-Bench, and TOUCAN under both Qwen2.5-7B and Llama3.1-8B. Compared with the strongest baselines, PhGPO consistently improves both Match Ratio and Task Success Rate, indicating that its gains appear in both reference-trajectory alignment and final task completion. The improvements are observed across different benchmarks and backbones, suggesting that the method is not tied to a specific task distribution or base model.

Table 1: **Main Results on Long-Horizon Tool-Use Benchmarks.** We report Match Ratio (Match R., %) and Task Success Rate (TSR, %) on Toolathlon, TRAJECT-Bench, and TOUCAN across two backbone models. Higher values (↑) indicate better performance.

Method	Toolathlon				TRAJECT-Bench				TOUCAN			
	Qwen2.5-7B		Llama3.1-8B		Qwen2.5-7B		Llama3.1-8B		Qwen2.5-7B		Llama3.1-8B	
	Match R.	TSR	Match R.	TSR	Match R.	TSR	Match R.	TSR	Match R.	TSR	Match R.	TSR
Standard Prompting & Planning Strategies												
ReAct	5.31	2.92	5.15	2.48	15.32	9.20	9.16	7.01	11.64	8.92	7.03	5.88
Plan-and-Solve	6.42	4.61	4.27	3.96	12.44	11.35	9.05	5.54	12.71	10.15	7.42	5.12
Beyond ReAct	3.85	3.10	5.23	4.18	11.33	10.74	15.61	14.20	12.92	10.84	7.35	6.54
Retrieval-Augmented Generation (RAG)												
Gorilla	6.60	3.26	5.78	3.65	29.04	22.62	19.38	11.42	22.32	16.05	18.99	16.94
ToolReAGt	4.05	4.42	3.45	3.17	26.69	20.11	26.84	18.67	23.49	18.77	15.79	10.76
Tool Graph Retriever	5.09	3.74	4.80	3.58	12.72	9.40	11.54	9.95	1.50	2.12	1.50	1.83
Reinforcement Learning												
ToRL	10.23	8.65	9.41	3.46	28.15	16.05	20.18	12.36	19.32	18.90	14.35	10.94
StepTool	14.33	12.08	11.58	7.87	35.22	23.18	26.31	20.12	19.75	20.86	14.46	15.30
TreeRL	21.26	18.91	18.78	11.42	35.45	27.82	26.33	20.04	30.65	18.74	20.20	18.62
Graph-based Navigation Models												
ToolNet	9.01	4.83	7.30	4.15	20.65	11.84	18.52	13.87	16.15	12.10	13.53	10.15
GAP	13.50	10.76	12.17	9.54	27.60	21.10	21.01	13.45	23.80	17.86	15.43	13.62
NaviAgent	15.28	13.94	11.60	10.86	25.33	20.78	19.40	14.83	23.10	20.44	16.20	16.05
GTool	20.60	17.82	16.50	14.22	32.50	28.06	26.20	21.78	27.56	19.35	19.50	13.33
PhGPO (Ours)	**25.25**	**20.34**	**22.83**	**19.76**	**41.62**	**32.18**	**31.84**	**22.66**	**35.17**	**26.28**	**24.67**	**19.86**

Compared with prompting and retrieval-augmented baselines. Prompting and planning baselines remain substantially below learning-based methods, reflecting the difficulty of maintaining coherent multi-step tool transitions through in-context reasoning alone. Retrieval-augmented methods can narrow the candidate tool space and improve local grounding, but their Task Success Rate remains limited on long-horizon episodes. This suggests that locally relevant tool retrieval does not necessarily lead to globally consistent tool-use trajectories, where later decisions must remain compatible with earlier tools, intermediate states, and argument invocations.

Compared with graph-based and RL-based optimization baselines. Graph-based navigation models improve over prompting baselines by using explicit tool structure, while RL-based methods further benefit from learning with execution outcomes. PhGPO outperforms these methods by using pheromone as an explicit transition prior during rollout generation and policy optimization. This prior biases sampling toward tool-transition edges and argument-invocation patterns that recur in verified successful trajectories, which improves multi-step consistency and reduces cascading errors. The higher Task Success Rate further shows that the improvement is not limited to reference-path imitation, but also leads to more reliable task completion.

4.3 Ablation Studies

We conduct ablation studies on Qwen2.5-7B using the same Toolathlon split, training budget, and evaluation protocol as the main experiment. All variants share the same backbone and optimization settings, with only one component modified at a time. We compare PhGPO with five variants: **(1) w/o Pheromone**, which sets $\beta{=}0$ throughout training; **(2) w/o Mixed Curriculum**, which removes the oracle-guided curriculum and accumulates pheromones only from fully policy-driven rollouts; **(3) Static Prior**, which freezes pheromones after the progressive training pipeline; **(4) w/o Evaporation**, which sets $\rho{=}0$ so pheromones only accumulate; and **(5) w/o Task-dependent Pheromones**, which uses only task-agnostic pheromones.

Table 2 shows that every component contributes to both reference-trajectory alignment and final task completion. Removing the Mixed Curriculum causes the largest drop, indicating that early guided rollouts help accumulate reliable pheromone statistics before training becomes fully policy driven. Disabling pheromone guidance also substantially degrades performance, showing that the explicit transition prior is important for long-horizon trajectory generation. Freezing pheromones or removing evaporation underperforms the full model, which supports the need for online updates and decay to keep the prior aligned with the evolving policy. Finally, removing task-dependent pheromones

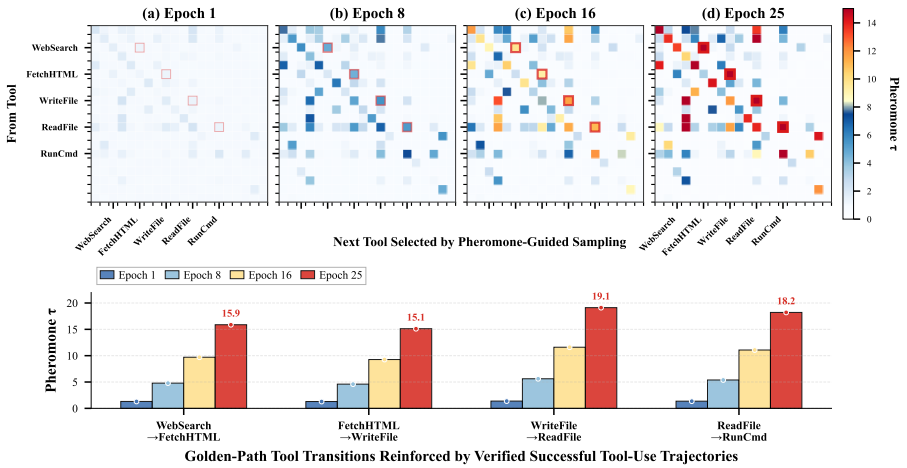

Figure 3: **Emergence of the reference chain in pheromone transitions.** As training proceeds, the transition matrix becomes increasingly concentrated on reference edges (red), while unrelated transitions fade due to evaporation, and pheromone values on the reference chain increase steadily.

Table 2: **Ablation studies on Toolathlon.** We report Match Ratio (Match R., %) and Task Success Rate (TSR, %) on the test set. Drop columns are computed relative to PhGPO (Full), where larger drops indicate stronger contribution of the removed component.

Variant	Variant role	Match R.↑	Drop M.↓	TSR↑	Drop TSR↓
w/o Pheromone (β=0 throughout)	removes explicit transition prior	21.23	4.02	16.72	3.62
w/o Mixed Curriculum	removes progressive oracle guidance	19.34	5.91	14.86	5.48
Static Prior	freezes pheromone updates	23.93	1.32	18.71	1.63
w/o Evaporation (ρ=0)	removes decay of stale statistics	21.72	3.53	16.28	4.06
w/o Task-dependent Pheromones	removes task-matched edge memory	22.45	2.80	17.35	2.99
PhGPO (Full)	**all components enabled**	**25.25**	–	**20.34**	–

reduces both Match Ratio and TSR, suggesting that task-matched transition evidence helps improve tool-use decisions for the current instance.

4.4 Visualization of transition structure

To qualitatively examine the transition structure learned by PhGPO, we present a representative long-horizon task involving sequential web retrieval and file manipulation. Figure 3 visualizes the evolution of the pheromone transition matrix during training.

At the beginning of training, the matrix is diffuse and no clear transition pattern is emphasized. As training proceeds, pheromone gradually concentrates on a small subset of transitions. In particular, the reference chain (*WebSearch* → *FetchHTML* → *WriteFile* → *ReadFile* → *RunCmd*) is repeatedly reinforced, while unsupported transitions fade under evaporation. By Epoch 25, the reference transitions are clearly highlighted, showing that PhGPO progressively accumulates and reuses validated transition patterns.

5 Conclusion and Limitations

We proposed PhGPO, a pheromone-guided policy optimization framework for long-horizon tool planning. PhGPO maintains a pheromone-based explicit transition prior over tool-transition edges and argument invocations, updates it through deposition and evaporation, and incorporates it into rollout sampling to reuse historically successful tool-use patterns. Experiments show consistent improvements in both Match Ratio and Task Success Rate, indicating benefits for reference-trajectory alignment and final task completion. We hope this work encourages future research on explicit experience reuse for tool-use agents. Future extensions may study broader verification settings, richer argument-invocation modeling, and more dynamic tool environments with evolving APIs.

Acknowledgement

This research work is supported by the Big Data Computing Center of Southeast University.

References

[1] Wentao Zhang, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *arXiv preprint arXiv:2506.12508*, 2025.

[2] Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. Mcp-universe: Benchmarking large language models with real-world model context protocol servers. *arXiv preprint arXiv:2508.14704*, 2025.

[3] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Proceedings of Advances in Neural Information Processing Systems*, pages 68539–68551, 2023.

[4] Kaibing Yang, Guangfeng Cai, Shengtian Yang, Shuo He, Yu Li, Mengyi Liu, Pengwei Chen, Jun Xu, and Lei Feng. Progress-conditioned group policy optimization for long-horizon agentic tasks. *arXiv preprint arXiv:2607.22724*, 2026.

[5] Shuozhe Li, Vaishnav Tadiparthi, Kwonjoon Lee, Nakul Agarwal, Hossein Nourkhiz Mahjoub, Ehsan Moradi Pari, Lizhang Chen, Amy Zhang, and Liu Leqi. Learning robust reasoning through guided adversarial self-play. *arXiv preprint arXiv:2602.00173*, 2026.

[6] Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *Proceedings of The Twelfth International Conference on Learning Representations*, 2024.

[7] Shengtian Yang, Yu Li, Shuo He, Yewen Li, Qingpeng Cai, Peng Jiang, and Lei Feng. Phase-aware mixture of experts for agentic reinforcement learning. *arXiv preprint arXiv:2602.17038*, 2026.

[8] Jiahao Qiu, Xuan Qi, Tongcheng Zhang, Xinzhe Juan, Jiacheng Guo, Yifu Lu, Yimin Wang, Zixin Yao, Qihan Ren, Xun Jiang, et al. Alita: Generalist agent enabling scalable agentic reasoning with minimal predefinition and maximal self-evolution. *arXiv preprint arXiv:2505.20286*, 2025.

[9] Jian Yang, Wei Zhang, Shark Liu, Jiajun Wu, Shawn Guo, and Yizhi Li. From code foundation models to agents and applications: A practical guide to code intelligence. *arXiv preprint arXiv:2511.18538*, 2025.

[10] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *Proceedings of The eleventh international conference on learning representations*, 2022.

[11] Yanming Liu, Xinyue Peng, Jiannan Cao, Shi Bo, Yuwei Zhang, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. Tool-planner: Task planning with clusters across multiple tools. In *Proceedings of The Thirteenth International Conference on Learning Representations*, 2025.

[12] Junlong Li, Wenshuo Zhao, Jian Zhao, Weihao Zeng, Haoze Wu, Xiaochen Wang, Rui Ge, Yuxuan Cao, Yuzhen Huang, Wei Liu, et al. The tool decathlon: Benchmarking language agents for diverse, realistic, and long-horizon task execution. *arXiv preprint arXiv:2510.25726*, 2025.

[13] Zhangchen Xu, Adriana Meza Soria, Shawn Tan, Anurag Roy, Ashish Sunil Agrawal, Radha Poovendran, and Rameswar Panda. Toucan: Synthesizing 1.5 m tool-agentic data from real-world mcp environments. *arXiv preprint arXiv:2510.01179*, 2025.

[14] Pengfei He, Zhenwei Dai, Bing He, Hui Liu, Xianfeng Tang, Hanqing Lu, Juanhui Li, Jiayuan Ding, Subhabrata Mukherjee, Suhang Wang, et al. Traject-bench: A trajectory-aware benchmark for evaluating agentic tool use. *arXiv preprint arXiv:2510.04550*, 2025.

[15] Yuanqing Yu, Zhefan Wang, Weizhi Ma, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. Steptool: Enhancing multi-step tool usage in llms via step-grained reinforcement learning. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pages 3952–3962, 2025.

[16] Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl. *arXiv preprint arXiv:2503.23383*, 2025.

[17] Zhenyu Hou, Ziniu Hu, Yujiang Li, Rui Lu, Jie Tang, and Yuxiao Dong. Treerl: LLM reinforcement learning with on-policy tree search. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pages 12355–12369, 2025.

[18] Fanbin Lu, Zhisheng Zhong, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Arpo: End-to-end policy optimization for gui agents with experience replay. *arXiv preprint arXiv:2505.16282*, 2025.

[19] Jiazheng Li, Yawei Wang, David Yan, Yijun Tian, Zhichao Xu, Huan Song, Panpan Xu, and Lin Lee Cheong. Salt: Step-level advantage assignment for long-horizon agents via trajectory graph. *arXiv preprint arXiv:2510.20022*, 2025.

[20] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(1):29–41, 1996.

[21] Anirudh Chari, Aditya Tiwari, Richard Lian, Suraj Reddy, and Brian Zhou. Pheromone-based learning of optimal reasoning paths. *arXiv preprint arXiv:2501.19278*, 2025.

[22] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems*, 2023.

[23] Dehai Min, Kailin Zhang, Tongtong Wu, and Lu Cheng. Quco-rag: Quantifying uncertainty from the pre-training corpus for dynamic retrieval-augmented generation. In *Findings of the Association for Computational Linguistics, ACL*, pages 16482–16500. Association for Computational Linguistics, 2026.

[24] Dehai Min, Giovanni Vaccarino, Huiyi Chen, Yongliang Wu, Gal Yona, and Lu Cheng. Stop when reasoning converges: Semantic-preserving early exit for reasoning models. *arXiv preprint arXiv:2605.17672*, 2026.

[25] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: large language model connected with massive apis. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 126544–126565, 2024.

[26] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

[27] Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. Stabletoolbench: Towards stable large-scale benchmarking on tool learning of large language models. *arXiv preprint arXiv:2403.07714*, 2024.

[28] Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023.

[29] Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 17754–17762, 2024.

[30] Norbert Braunschweiler, Rama Doddipatla, and Tudor-Catalin Zorila. Toolreagt: Tool retrieval for llm-based complex task solution via retrieval augmented generation. In *Proceedings of the 3rd Workshop on Towards Knowledgeable Foundation Models*, pages 75–83, 2025.

[31] Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. Toolgen: Unified tool retrieval and calling via generation. In *Proceedings of The Thirteenth International Conference on Learning Representations*, 2025.

[32] Yu Pan, Xiaocheng Li, and Hanzhao Wang. Online-optimized rag for tool use and function calling. *arXiv preprint arXiv:2509.20415*, 2025.

[33] Xukun Liu, Zhiyuan Peng, Xiaoyuan Yi, Xing Xie, Lirong Xiang, Yuchen Liu, and Dongkuan Xu. Toolnet: Connecting large language models with massive tools via tool graph. *arXiv preprint arXiv:2403.00839*, 2024.

[34] Zhaoyang Liu, Zeqiang Lai, Zhangwei Gao, Erfei Cui, Ziheng Li, Xizhou Zhu, Lewei Lu, Qifeng Chen, Yu Qiao, Jifeng Dai, et al. Controlllm: Augment language models with tools by searching on graphs. In *European Conference on Computer Vision*, pages 89–105, 2024.

[35] Yan Jiang, Hao Zhou, LiZhong GU, Ai Han, and TianLong Li. Naviagent: Bilevel planning on tool dependency graphs for function calling. *arXiv preprint arXiv:2506.19500*, 2025.

[36] Wenjie Chen, Wenbin Li, Di Yao, Xuying Meng, Chang Gong, and Jingping Bi. Gtool: Graph enhanced tool planning with large language model. *arXiv preprint arXiv:2508.12725*, 2025.

[37] Yuchen Zhuang, Xiang Chen, Tong Yu, Saayan Mitra, Victor S. Bursztyn, Ryan A. Rossi, Somdeb Sarkhel, and Chao Zhang. Toolchain*: Efficient action space navigation in large language models with a* search. In *Proceedings of The Twelfth International Conference on Learning Representations*, 2024.

[38] Guoxin Chen, Zhong Zhang, Xin Cong, Fangda Guo, Yesai Wu, Yankai Lin, Wenzheng Feng, and Yasheng Wang. Learning evolving tools for large language models. In *Proceedings of The Thirteenth International Conference on Learning Representations*, 2025.

[39] Yifan Wei, Xiaoyan Yu, Yixuan Weng, Tengfei Pan, Angsheng Li, and Li Du. Autotir: Autonomous tools integrated reasoning via reinforcement learning. *arXiv preprint arXiv:2507.21836*, 2025.

[40] Yanfei Zhang. Agent-as-tool: A study on the hierarchical decision making with reinforcement learning. *arXiv preprint arXiv:2507.01489*, 2025.

[41] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.

[42] Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*, 2025.

[43] Xiangru Tang, Tianrui Qin, Tianhao Peng, Ziyang Zhou, Daniel Shao, Tingting Du, Xinming Wei, Peng Xia, Fang Wu, He Zhu, et al. Agent kb: Leveraging cross-domain experience for agentic problem solving. *arXiv preprint arXiv:2507.06229*, 2025.

[44] Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of Forty-second International Conference on Machine Learning*, 2025.

[45] Jiahao Qiu, Xinzhe Juan, Yimin Wang, Ling Yang, Xuan Qi, Tongcheng Zhang, Jiacheng Guo, Yifu Lu, Zixin Yao, Hongru Wang, et al. Agentdistill: Training-free agent distillation with generalizable mcp boxes. *arXiv preprint arXiv:2506.14728*, 2025.

[46] Erhu Feng, Wenbo Zhou, Zibin Liu, Le Chen, Yunpeng Dong, Cheng Zhang, Yisheng Zhao, Dong Du, Zhichao Hua, Yubin Xia, et al. Get experience from practice: Llm agents with record & replay. *arXiv preprint arXiv:2505.17716*, 2025.

[47] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, 26(1):29–41, 1996.

[48] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1 (1):53–66, 2002.

[49] Luca Maria Gambardella and Marco Dorigo. Ant-q: A reinforcement learning approach to the traveling salesman problem. In *Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995*, pages 252–260. Morgan Kaufmann, 1995.

[50] Gianni Di Caro and Marco Dorigo. Antnet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.

[51] N Monekosso and Paolo Remagnino. The analysis and performance evaluation of the pheromone-q-learning algorithm. *Expert Systems*, 21(2):80–91, 2004.

[52] Jelmer Marinus van Ast, Robert Babuška, and Bart De Schutter. Ant colony learning algorithm for optimal control. In *Interactive Collaborative Information Systems*, pages 155–182. Springer, 2010.

[53] Anirudh Chari, Aditya Tiwari, Richard Lian, Suraj Reddy, and Brian Zhou. Pheromone-based learning of optimal reasoning paths. *arXiv preprint arXiv:2501.19278*, 2025.

[54] Oron Anschel, Alon Shoshan, Adam Botach, Shunit Haviv Hakimi, Asaf Gendler, Emanuel Ben Baruch, Nadav Bhonker, Igor Kviatkovsky, Manoj Aggarwal, and Gerard Medioni. Group-aware reinforcement learning for output diversity in large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32382–32403, 2025.

[55] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *arXiv preprint arXiv:2505.10978*, 2025.

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

[57] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 2609–2634, 2023.

[58] Xiaolong Wei, Yuehu Dong, Xingliang Wang, Xingyu Zhang, Zhejun Zhao, Dongdong Shen, Long Xia, and Dawei Yin. Beyond react: A planner-centric framework for complex tool-augmented llm reasoning. *arXiv preprint arXiv:2511.10037*, 2025.

[59] Linfeng Gao, Yaoxiang Wang, Minlong Peng, Jialong Tang, Yuzhe Shang, Mingming Sun, and Jinsong Su. Tool graph retriever: Exploring dependency graph-based tool retrieval for large language models. *arXiv preprint arXiv:2508.05152*, 2025.

[60] Jiaqi Wu, Qinlao Zhao, Zefeng Chen, Kai Qin, Yifei Zhao, Xueqian Wang, and Yuhang Yao. Gap: Graph-based agent planning with parallel tool use and reinforcement learning. *arXiv preprint arXiv:2510.25320*, 2025.

[61] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[62] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[63] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations*, 2022.

[64] Yu Li, Shenyu Zhang, Rui Wu, Xiutian Huang, Yongrui Chen, Wenhao Xu, Guilin Qi, and Dehai Min. Mateval: A multi-agent discussion framework for advancing open-ended text evaluation. In *Database Systems for Advanced Applications - 29th International Conference, DASFAA*, pages 415–426, 2024.

[65] Yu Li, Yi Huang, Guilin Qi, Junlan Feng, Nan Hu, Songlin Zhai, Haohan Xue, Yongrui Chen, Ruoyan Shen, and Tongtong Wu. Harnessing diverse perspectives: A multi-agent framework for enhanced error detection in knowledge graphs. In *Database Systems for Advanced Applications - 30th International Conference, DASFAA*, volume 15991, pages 446–458, 2025.

[66] Yu Li, Zhe Yang, Yi Huang, Xin Liu, and Guilin Qi. C^3tg: Conflict-aware, composite, and collaborative controlled text generation. In *Fortieth AAAI Conference on Artificial Intelligence, Thirty-Eighth Conference on Innovative Applications of Artificial Intelligence, Sixteenth Symposium on Educational Advances in Artificial Intelligence, AAAI*, pages 31834–31842, 2026.

A Theoretical Motivation of Pheromone-guided Sampling

This appendix provides a principled view of the pheromone-guided sampling distribution in Eq. (7). The key idea is that the sampling rule can be derived as the closed-form solution of an entropy-regularized consensus problem that combines the neural policy with an *explicit transition prior* induced by pheromone.

A.1 Setup

At decision step t, the policy produces a distribution $\pi_\theta(a \mid s_t)$ over tools. Following Sec. 3.3, we restrict sampling to a finite support $\mathcal{A}_t$ constructed from the Top-K candidates under $\pi_\theta(\cdot \mid s_t)$. We use pheromone as an explicit transition prior through the fused tool-transition pheromone $\tau_{\text{Tool}}(a_{t-1}, a \mid x)$ defined in Sec. 3.2, which already incorporates task-agnostic and task-dependent evidence via Eq. (5). To ensure $\log \tau_{\text{Tool}}$ is well-defined, we clip pheromone values:

$$\bar{\tau}(a_{t-1}, a \mid x) = \text{clip}\big(\tau_{\text{Tool}}(a_{t-1}, a \mid x), [\tau_{\min}, \tau_{\max}]\big), \qquad \tau_{\min} > 0. \tag{12}$$

We define the pheromone-induced explicit transition prior over the candidate set as

$$p_\tau(a \mid a_{t-1}, x) \propto \big[\bar{\tau}(a_{t-1}, a \mid x)\big]^\beta, \qquad a \in \mathcal{A}_t, \tag{13}$$

where $\beta \geq 0$ controls the strength of the explicit transition prior. The pheromone-guided sampling distribution used in our rollouts is

$$\tilde{\pi}(a \mid s_t, a_{t-1}, x) \propto \exp\Big(\log \pi_\theta(a \mid s_t) + \beta \log \bar{\tau}(a_{t-1}, a \mid x)\Big), \qquad a \in \mathcal{A}_t, \tag{14}$$

which is equivalent to the multiplicative form $\tilde{\pi} \propto \pi_\theta(\cdot \mid s_t)\, p_\tau(\cdot \mid a_{t-1}, x)$ on the support $\mathcal{A}_t$.

A.2 Entropy-regularized consensus objective

We derive $\tilde{\pi}$ by solving for a distribution q over the candidate set $\mathcal{A}_t$ that agrees with both the neural policy and the explicit transition prior, while retaining entropy:

$$\max_q \ \mathbb{E}_{a\sim q}\Big[\log \pi_\theta(a \mid s_t) + \beta \log \bar{\tau}(a_{t-1}, a \mid x)\Big] + \eta\, \mathcal{H}(q), \quad \text{s.t.} \sum_{a\in\mathcal{A}_t} q(a) = 1,\ q(a) \geq 0, \tag{15}$$

where $\mathcal{H}(q) = -\sum_{a\in\mathcal{A}_t} q(a) \log q(a)$ is Shannon entropy and $\eta > 0$ is an entropy weight.

A.3 Closed-form solution via Lagrange multipliers

Introduce a Lagrange multiplier γ for the normalization constraint and write the Lagrangian

$$\begin{aligned}\mathcal{L}(q, \gamma) = &\sum_{a\in\mathcal{A}_t} q(a)\Big(\log \pi_\theta(a \mid s_t) + \beta \log \bar{\tau}(a_{t-1}, a \mid x)\Big) \\ &+ \eta\Big(-\sum_{a\in\mathcal{A}_t} q(a) \log q(a)\Big) \\ &+ \gamma\Big(\sum_{a\in\mathcal{A}_t} q(a) - 1\Big).\end{aligned} \tag{16}$$

Taking partial derivatives with respect to $q(a)$ and setting them to zero gives

$$\log \pi_\theta(a \mid s_t) + \beta \log \bar{\tau}(a_{t-1}, a \mid x) - \eta\big(1 + \log q(a)\big) + \gamma = 0. \tag{17}$$

Rearranging yields

$$\log q(a) = \frac{1}{\eta}\Big(\log \pi_\theta(a \mid s_t) + \beta \log \bar{\tau}(a_{t-1}, a \mid x)\Big) + \frac{\gamma}{\eta} - 1, \tag{18}$$

which implies the optimal solution

$$q^*(a) \propto \exp\Big(\tfrac{1}{\eta}\big[\log \pi_\theta(a \mid s_t) + \beta \log \bar{\tau}(a_{t-1}, a \mid x)\big]\Big), \qquad a \in \mathcal{A}_t. \tag{19}$$

Setting $\eta = 1$ recovers Eq. (14), which matches our implementation in Eq. (7).

A.4 Equivalent KL projection view

The same objective admits an information-geometric interpretation. Let

$$r(a) \propto \exp\Big(\log \pi_\theta(a \mid s_t) + \beta \log \bar{\tau}(a_{t-1}, a \mid x) \Big), \qquad a \in \mathcal{A}_t. \tag{20}$$

Expanding $\mathrm{KL}(q\|r) = \sum_{a\in\mathcal{A}_t} q(a) \log \frac{q(a)}{r(a)}$ gives

$$-\mathrm{KL}(q\|r) = \mathbb{E}_{a\sim q}\big[\log \pi_\theta(a \mid s_t) + \beta \log \bar{\tau}(a_{t-1}, a \mid x) \big] + \mathcal{H}(q) + \text{const}. \tag{21}$$

For $\eta = 1$, maximizing Eq. (15) is equivalent, up to constants, to the KL projection

$$\min_q \ \mathrm{KL}\big(q \,\big\|\, \tfrac{1}{Z} \exp\big(\log \pi_\theta(\cdot \mid s_t) + \beta \log \bar{\tau}(a_{t-1}, \cdot \mid x) \big)\big) , \tag{22}$$

where Z is the normalizer over $\mathcal{A}_t$. This view clarifies the role of pheromone as an explicit transition prior in log space: actions must be supported by the neural policy and by the accumulated transition statistics encoded in $\bar{\tau}$, which increases the probability of historically successful tool-transition edges under the current task context.

B Algorithm Implementation Details and Training Mechanisms

B.1 Base model adaptation

We instantiate PhGPO on instruction-tuned backbones used in our main experiments, including Qwen2.5-7B and Llama3.1-8B. We apply LoRA for parameter-efficient adaptation by freezing backbone weights and inserting low-rank adapters with rank r=64 and scaling α=128 into standard projection modules. All stages use the tokenizer and chat template of the corresponding backbone, and tool planning is trained as a generative decision process over the same two-layer action space described in Sec. 3.1. Our code is available at `https://anonymous.4open.science/r/PhGPO-Pheromone-Guided-Policy-Optimization-for-Long-Horizon-Tool-Planning-CB1C`.

B.2 Progressive training pipeline

Training follows the progressive pipeline described in Sec. 3.3. It begins with supervised warm-up using the next-tool prediction objective in Eq. (6), which provides a stable initialization for tool selection. After warm-up, training uses a progressive schedule that mixes oracle guidance with pheromone-guided sampling. At each decision step, the agent takes the reference action with probability p_{tf}; otherwise it samples actions from the pheromone-guided distribution in Eq. (7), as specified in Eq. (9). Policy updates use the mixed objective in Eq. (10), where λ is annealed downward to shift learning from imitation toward policy optimization. In parallel, a horizon curriculum increases the rollout horizon from shorter to longer trajectories. We also schedule β in Eq. (7) and w in Eq. (5) upward as more verified trajectories are accumulated, which increases the influence of the explicit transition prior and task-dependent evidence. When p_{tf} approaches zero and λ becomes small, rollouts become fully policy-driven under pheromone guidance, and optimization follows the policy objective described in Sec. 3.3. For group-based learning, we use M=5 rollouts per instance, consistent with Sec. 4.1.

B.3 Mixed training objective under teacher forcing

During progressive training pipeline, rollouts are generated under a mixed behavior rule that combines oracle actions and pheromone-guided sampling. Let a_t^* denote the reference action provided by the benchmark instance, and let $\tilde{\pi}$ denote the pheromone-guided sampling distribution in Eq. (7). The behavior distribution used for rollout generation is

$$\pi_{\text{mix}}(a_t \mid s_t) = p_{\text{tf}} \cdot \delta(a_t = a_t^*) \ + \ (1 - p_{\text{tf}}) \cdot \tilde{\pi}(a_t \mid s_t), \tag{23}$$

where $\delta(\cdot)$ is a point mass on the oracle action. We record the executed action a_t at every step, together with its log-probability under the current policy at rollout time, $\log \pi_{\theta_{\text{old}}}(a_t \mid s_t)$.

Policy optimization uses GRPO with a PPO-style clipped surrogate. For each training instance, we sample a group of M tool-use trajectories and compute their returns $\{R_m\}_{m=1}^M$ (Sec. B.5). We

Algorithm 1 PhGPO Training Loop with Task-dependent Pheromone Fusion

Input: dataset $\mathcal{D}$ with task x and reference trajectory $\{a_t^*\}$; horizon curriculum $\{L_s\}_{s=1}^{S}$; schedules $p_{\text{tf}}(s), \lambda(s), \beta(s), w(s)$; group size M; tool graph $\mathcal{G}{=}(\mathcal{V}_{\text{Tool}}, \mathcal{E})$ and invocation edges $\mathcal{E}_{\text{arg}}$; sentence encoder $\phi(\cdot)$; ACO parameters $(\rho, \alpha, \tau_{\min}, \tau_{\max}, \tau_0)$; $\theta_{\text{sim}}, n_{\min}, \epsilon_{\text{dep}}$.
Output: policy π_θ; pheromones $\tau^{\text{agn}}_{\text{Tool}}, \tau^{\text{agn}}_{\text{Arg}}$; memory banks $\{\mathcal{M}\}$.
Initialize π_θ; set $\tau^{\text{agn}}_{\text{Tool}}, \tau^{\text{agn}}_{\text{Arg}} \leftarrow \tau_0$; initialize all memory banks as empty.
Subroutine $\text{ComputeFusedPheromone}(\tau^{\text{agn}}, \mathcal{M}, \boldsymbol{e}_x, w)$:
Retrieve $\mathcal{M}(x) = \{(\boldsymbol{e}, q_\tau) \in \mathcal{M} \mid \text{sim}(\boldsymbol{e}_x, \boldsymbol{e}) \geq \theta_{\text{sim}}\}$.
Compute $\tau^{\text{dep}}(\cdot \mid x)$ by Eq. (3) and confidence $c(x)$ by Eq. (4).
Return fused $\tau(\cdot \mid x)$ by Eq. (5), clipped to $[\tau_{\min}, \tau_{\max}]$.
for epochs **do**
  Update π_θ by minimizing $\mathcal{L}^{\text{SL}}$ in Eq. (6).
end for
for $s = 1$ **to** S **do**
  $L \leftarrow L_s$; $p_{\text{tf}} \leftarrow p_{\text{tf}}(s)$; $\lambda \leftarrow \lambda(s)$; $\beta \leftarrow \beta(s)$; $w \leftarrow w(s)$.
  for each $(x, \{a_t^*\}) \in \mathcal{D}$ **do**
    $\boldsymbol{e}_x \leftarrow \phi(x)$.
    for $m = 1$ **to** M **do**
      $a_0 \leftarrow \langle \texttt{START} \rangle$; initialize ξ_m.
      for $t = 1$ **to** L **do**
        Build state s_t from (x, ξ_m).
        if $\text{rand}() < p_{\text{tf}}$ **then**
          $a_t \leftarrow a_t^*$ {Eq. (9)}
        else
          Form candidate set $\mathcal{A}_t$ as Top-K under $\pi_\theta(\cdot \mid s_t)$.
          for each $a \in \mathcal{A}_t$ **do**
            $\tau_{\text{Tool}}(a_{t-1}, a \mid x) \leftarrow \text{ComputeFusedPheromone}\big(\tau^{\text{agn}}_{\text{Tool}}(a_{t-1}, a), \mathcal{M}_{a_{t-1},a}, \boldsymbol{e}_x, w\big)$.
          end for
          Sample a_t using Eq. (7) with β on support $\mathcal{A}_t$.
        end if
        Let $\mathcal{U}_t \leftarrow \mathcal{V}_{\text{Arg}}(a_t)$.
        for each $u \in \mathcal{U}_t$ **do**
          $\tau_{\text{Arg}}(a_t, u \mid x) \leftarrow \text{ComputeFusedPheromone}\big(\tau^{\text{agn}}_{\text{Arg}}(a_t, u), \mathcal{M}_{a_t,u}, \boldsymbol{e}_x, w\big)$.
        end for
        Sample u_t by the analogous pheromone-guided rule; execute (a_t, u_t) and append to ξ_m.
      end for
      Compute return R_m and pheromone-deposition score $q_\tau(\xi_m) \in [0, 1]$ (match ratio).
    end for
    Compute advantages by Eq. (8).
    Update π_θ using the mixed objective in Eq. (10).
    Global pheromone evaporation:
    $\tau^{\text{agn}}_{\text{Tool}} \leftarrow \text{clip}\big((1-\rho)\tau^{\text{agn}}_{\text{Tool}}, [\tau_{\min}, \tau_{\max}]\big)$.
    $\tau^{\text{agn}}_{\text{Arg}} \leftarrow \text{clip}\big((1-\rho)\tau^{\text{agn}}_{\text{Arg}}, [\tau_{\min}, \tau_{\max}]\big)$.
    Pheromone deposition from successful rollouts:
    for each rollout ξ_m that is verified successful **do**
      for each traversed tool-transition edge (v_i, v_j) in ξ_m **do**
        $\tau^{\text{agn}}_{\text{Tool}}(v_i, v_j) \leftarrow \text{clip}\big(\tau^{\text{agn}}_{\text{Tool}}(v_i, v_j) + \alpha q_\tau(\xi_m), [\tau_{\min}, \tau_{\max}]\big)$.
        Store $(\boldsymbol{e}_x, q_\tau(\xi_m))$ into $\mathcal{M}_{ij}$.
      end for
      for each traversed invocation edge (v_i, u) in ξ_m **do**
        $\tau^{\text{agn}}_{\text{Arg}}(v_i, u) \leftarrow \text{clip}\big(\tau^{\text{agn}}_{\text{Arg}}(v_i, u) + \alpha q_\tau(\xi_m), [\tau_{\min}, \tau_{\max}]\big)$.
        Store $(\boldsymbol{e}_x, q_\tau(\xi_m))$ into $\mathcal{M}_{i,u}$.
      end for
    end for
  end for
end for

normalize returns within the group to obtain advantages A_m as in Eq. (8). The policy-gradient loss is computed on executed actions under π_{mix}:

$$\mathcal{L}_{\text{PG}} = -\mathbb{E}_{m,t}\Big[\min\Big(r_{m,t}(\theta)A_m,\ \text{clip}(r_{m,t}(\theta), 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})A_m\Big)\Big], \\ r_{m,t}(\theta) = \frac{\pi_\theta(a_{m,t} \mid s_{m,t})}{\pi_{\theta_{\text{old}}}(a_{m,t} \mid s_{m,t})}. \tag{24}$$

When an action is oracle-selected, Eq. (24) still evaluates its probability under π_θ, since that action is the one executed in the environment under π_{mix}. In parallel, the supervised term in Eq. (10) is computed as a next-action cross-entropy against a_t^*. As p_{tf} is annealed toward zero, the rollout distribution approaches fully policy-driven pheromone-guided sampling, and updates become dominated by autonomous decisions.

B.4 Pheromone and memory

PhGPO maintains pheromone on two types of edges: tool-transition edges $(v_i, v_j) \in \mathcal{E}$ and tool-to-invocation edges $(v_i, v_i^{(k)}) \in \mathcal{E}_{\text{arg}}$ defined in Sec. 3.1. We maintain task-agnostic pheromones $\tau^{\text{agn}}_{\text{Tool}}$ and $\tau^{\text{agn}}_{\text{Arg}}$, and we clip pheromone values to $[\tau_{\min}, \tau_{\max}]$ as described in Sec. 3.2.

Task embeddings and sentence encoder. Task-dependent pheromones use a fixed sentence encoder ϕ to embed the task text x into a dense vector $\boldsymbol{e} = \phi(x)$. In our reference implementation, ϕ is instantiated with `SentenceTransformer(all-MiniLM-L6-v2)`. Edge memory banks store tuples $(\boldsymbol{e}, q_\tau)$ for each edge, where q_τ is the pheromone-deposition score of the corresponding tool-use trajectory (Sec. B.5).

We additionally maintain edge-indexed memory banks for task-dependent pheromones. For a tool-transition edge (v_i, v_j), its memory bank $\mathcal{M}_{ij}$ stores tuples $(\boldsymbol{e}, q_\tau)$, where $\boldsymbol{e}{=}\phi(x)$ is the task embedding and q_τ is the pheromone-deposition score associated with a verified tool-use trajectory that contains this edge. Given a task x, we compute task-dependent pheromones by retrieval and similarity-weighted aggregation as in Eq. (3), estimate confidence by Eq. (4), and then combine task-agnostic and task-dependent pheromones using Eq. (5). We apply the same mechanism to tool-to-invocation edges for τ_{Arg}.

Rationale and locality of the confidence score. The confidence score in Eq. (4) is computed *locally* from the retrieved set $\mathcal{M}_{ij}(x)$ for the queried task x. In particular, n_{act} counts the number of retrieved memories that pass activation thresholds, $s_{\max}$ is the maximum similarity among activated memories, and $\bar{q}_\tau$ is the mean pheromone-deposition score over the same activated subset. We incorporate $\bar{q}_\tau$ in confidence to gate task-dependent pheromones when retrieval returns semantically similar but consistently low-quality memories. This design keeps the fused explicit transition prior conservative, assigning higher weight to context-aware pheromones only when retrieved evidence is both relevant (high similarity) and reliable (high average quality), and otherwise falling back to task-agnostic pheromones.

B.5 Reward signals and pheromone-deposition score

We distinguish two quantities computed per rollout: the episode return R used for policy optimization, and the pheromone-deposition score $q_\tau(\xi)$ used for pheromone deposition and memory writing. They are computed together during rollout generation but serve different roles in training.

Reward structure and episode return. For each rollout, we compute an episode return R as the sum of step-level rewards and an outcome-level bonus. At each step t, the step reward decomposes into an intent component and an execution component,

$$r_t = r_t^{\text{intent}} + r_t^{\text{exec}}. \tag{25}$$

The intent reward r_t^{intent} encourages selecting the correct next base tool. Concretely, we assign $+0.5$ for an exact match to the reference next base tool, and $+0.2$ for a category-consistent selection (i.e., the predicted tool belongs to the same predefined tool category as the reference). To mitigate error propagation, we additionally provide a recovery bonus of $+0.1$ when the previous step is incorrect

but the current step returns to the reference tool.[2] Otherwise, $r_t^{\text{intent}} = 0$. The execution reward r_t^{exec} is assigned after simulating the selected tool invocation (Sec. B.6). We assign $+0.5$ when the simulator returns a valid execution without triggering an error pattern, and -0.5 when an execution error or timeout is detected. We treat invalid argument formatting as an execution failure and apply the same penalty.

After the rollout terminates, we add an outcome-level bonus that rewards completed episodes and, when completion is not achieved, provides graded bonuses based on the final Match Ratio. The episode return used by GRPO is

$$R = \sum_{t=1}^{T} r_t \; + \; r_{\text{outcome}}. \tag{26}$$

Pheromone-deposition score. The pheromone deposition mechanism in Eq. (1) relies on a pheromone-deposition score $q_\tau(\xi) \in [0, 1]$. In our benchmark setting, each instance provides a reference tool-use trajectory $\xi^* = (a_1^*, \ldots, a_T^*)$. Given a rollout trajectory $\xi = (a_1, \ldots, a_T)$, we define

$$q_\tau(\xi) \;=\; \text{Match}(\xi, \xi^*) \;=\; \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}[\text{tool}(a_t) = \text{tool}(a_t^*)]\,, \tag{27}$$

where $\text{tool}(\cdot)$ extracts the base tool identity from the two-layer action, and the indicator compares step-aligned tool identities. This definition uses the reference tool-use trajectory, and it is computed independently from the environmental return R in Eq. (26). We maintain both quantities since R provides the learning signal for policy optimization, while $q_\tau(\xi)$ only updates pheromone statistics.

Teacher forcing and model-driven quality. During mixed rollouts with oracle guidance, we record a per-step teacher-forcing mask indicating whether a_t was oracle-selected. We additionally report a model-driven Match Ratio computed on the subset of steps not teacher-forced:

$$q_{\text{model}}(\xi) = \frac{1}{|\mathcal{T}_{\text{model}}|} \sum_{t \in \mathcal{T}_{\text{model}}} \mathbb{I}[\text{tool}(a_t) = \text{tool}(a_t^*)]\,, \qquad \mathcal{T}_{\text{model}} = \{t : \; a_t \text{ is not oracle-selected}\}. \tag{28}$$

This quantity diagnoses how much alignment is attributable to autonomous decisions. In addition, we only collect elite replay trajectories after p_{tf} decays below a threshold, which limits the impact of heavily teacher-forced trajectories on subsequent pheromone reinforcement.

B.6 Tool execution simulation

Our training environment executes tool calls through a simulator, not by invoking external APIs. This design keeps rollouts reproducible, avoids network variability, and prevents unintended side effects. Given the selected tool and its argument invocation, the simulator produces a tool output and a completion signal that indicates whether the episode has reached a terminal success state. The simulator can replay cached outputs from an offline database, and it also supports a pluggable LLM-based simulator backend that generates tool outputs via an OpenAI-compatible interface. The execution reward in Sec. B.5 is computed from this simulated output, assigning positive reward to valid executions and penalizing execution errors or malformed outputs. This simulator-based reward is used for training and rollout feedback. For evaluation, Task Success Rate is measured by GPT-4o as an outcome judge under a fixed protocol, based on the task requirement, generated tool-use trajectory, and final tool outputs. The judge evaluates task completion rather than reference-order matching.

Fairness of comparisons. To ensure fair comparisons, all methods share the same backbones, tool set, decoding configuration, rollout horizon limits, and evaluation protocol, and they are trained under the same computational budget whenever applicable. For baseline methods that require tool-format awareness (e.g., producing valid tool names or argument fields), we apply the same lightweight tool-format fine-tuning procedure used in our approach (e.g., the same LoRA setup on identical tool-call supervision) to calibrate tool syntax and reduce avoidable execution failures. This control isolates the effect of the learning algorithm and sampling strategy, and it prevents differences in tool-call formatting from dominating the reported performance.

[2] This recovery term is only applied when $a_{t-1} \neq a_{t-1}^*$ and $a_t = a_t^*$.

C Datasets

We evaluate on three long-horizon tool-use benchmarks: Toolathlon, TRAJECT-Bench, and TOUCAN. Each benchmark provides (i) executable tools or tool interfaces, and (ii) a reference tool-use sequence, which enables fine-grained comparison between predicted tool-use sequences and the reference. Following our main experimental protocol, we use the official splits when available, and we only keep episodes whose reference tool-use sequences are verifiably successful.

To align with our Tool-Transition Graph action space (Sec. 3.1), we report dataset-scale statistics in terms of: **L1 Tools**, the number of base tools in $\mathcal{V}_{\text{Tool}}$; **L2 Tools**, the number of argument-invocation patterns aggregated across tools in $\mathcal{V}_{\text{Arg}}$; and **Avg Steps**, the average number of tool calls per verifiably successful reference tool-use sequence in the retained episodes.

Toolathlon. Toolathlon is a long-horizon benchmark designed for realistic, execution-based evaluation across diverse software applications. After mapping tool calls into our Tool-Transition Graph action representation, Toolathlon contains 618 L1 tools and 1,250 L2 argument-invocation patterns. Among the retained verifiably successful reference sequences, the average tool-call length is 13.48 steps.

TRAJECT-Bench. TRAJECT-Bench is a trajectory-aware benchmark that emphasizes correctness of tool selection, tool arguments, and dependency or ordering constraints along the full tool-use trajectory. In our Tool-Transition Graph representation, TRAJECT-Bench contains 381 L1 tools and 715 L2 argument-invocation patterns. Among the retained verifiably successful reference sequences, the average tool-call length is 6.36 steps.

TOUCAN (Multi-step only). TOUCAN is a large-scale tool-agent dataset synthesized from real-world Model Context Protocol (MCP) environments. We only consider the Multi-step subset, defined as instances with at least two tool calls in the reference sequence. In our Tool-Transition Graph representation, the Multi-step subset contains 850 L1 tools and 15,294 L2 argument-invocation patterns. Among the retained verifiably successful reference sequences, the average tool-call length is 3.16 steps.

Table 3: **Dataset Statistics.** We summarize the scale of the three benchmarks under our Tool-Transition Graph action representation. L1 Tools denotes the number of base tools, and L2 Tools denotes the number of argument-invocation patterns. Avg Steps reports the average number of tool calls per verifiably successful reference tool-use sequence in the retained episodes. For TOUCAN, we only consider Multi-step instances.

Dataset	Tool source	L1 Tools	L2 Tools	#Instances	Avg Steps
Toolathlon	MCP apps	618	1,250	108 tasks	13.48
TRAJECT-Bench	executable APIs	381	715	5,870 queries	6.36
TOUCAN (Multi-step)	MCP servers	850	15,294	1,646,546 traj.	3.16

D GPT-4o Judge Protocol for Task Success Rate

Task Success Rate (TSR) measures whether a generated tool-use trajectory successfully completes the target task. We evaluate TSR using GPT-4o as an outcome judge under a fixed evaluation protocol [64–66]. This evaluation is used only for reporting test performance and is not used during training or pheromone updating.

For each test instance, we provide the judge with the task instruction, the generated tool-use trajectory, and the final tool outputs. We do not provide the method name to the judge. The judge is asked to evaluate whether the final outcome satisfies the task requirement, independent of whether the generated trajectory follows the reference ordering. This design is important because a task can sometimes be completed through a valid alternative tool-use path even when its Match Ratio to the reference trajectory is not perfect.

The judge outputs a binary decision, where `success` indicates that the generated trajectory completes the task and `failure` indicates that it does not. A trajectory is labeled as successful only when the

final output satisfies the main task requirement and is supported by the executed tool outputs. Partial completion, missing outputs, invalid tool results, hallucinated results, or outputs that do not address the requested task are labeled as failures. Minor formatting differences are ignored when they do not affect task correctness. We use the same judge prompt and deterministic decoding setting for all methods.

Formally, for a test set with N instances, let $y_i \in \{0, 1\}$ denote the GPT-4o judge label for the i-th generated trajectory, where 1 indicates success. We compute TSR as

$$\mathrm{TSR} = \frac{1}{N} \sum_{i=1}^{N} y_i \times 100\%. \tag{29}$$

The judge prompt template is shown below.

> You are an impartial evaluator for long-horizon tool-use agents. Given a task instruction, a generated tool-use trajectory, and the final tool outputs, determine whether the agent successfully completed the task.
>
> Evaluate only the final task outcome. Do not require the generated trajectory to follow the reference tool ordering. A different tool-use path can be marked as successful if it produces a correct final outcome. Mark the trajectory as failure if the final output is missing, incomplete, unsupported by tool outputs, based on invalid tool execution, or does not satisfy the task requirement.
>
> Return your judgment in the following format:
>
> `{"success": 0 or 1, "reason": "brief explanation"}`

E Ablation Study on Pheromone Influence Parameter β

This appendix studies the pheromone influence parameter β in Eq. (7), which scales the explicit transition prior during pheromone-guided sampling. β controls how strongly historically verified tool-transition patterns bias the next tool choice, shaping the exploration exploitation trade-off in long-horizon planning.

Specifically, we sample tools from

$$p(a_t \mid s_t, a_{t-1}) \propto \exp\Big(\log \pi_\theta(a_t \mid s_t) + \beta \log \tau_{\text{Tool}}(a_{t-1}, a_t \mid x) \Big), \tag{30}$$

where $\beta \geq 0$ determines the strength of pheromone guidance. $\beta = 0$ removes pheromone, while larger β increasingly favors transitions with higher pheromone, which can reduce exploration if set too aggressively.

E.1 Experimental Setup

We ablate β on Toolathlon with Qwen2.5-7B. We compare: (i) $\beta = 0$ (no pheromone), (ii) $\beta = 1$ (fixed moderate), (iii) $\beta = 5$ (strong over-guidance), and (iv) our dynamic schedule, where β is linearly increased from 0 to 0.8 over the first 30% of epochs and then kept at 0.8. All variants share the same backbone, optimizer, learning rate schedule, rollout group size ($M = 5$), and evaluation protocol. We train for 25 epochs and report Match Ratio (Match R.) and Next-tool Accuracy (TAcc). We also measure *Exploration Diversity* as the normalized entropy of the action distribution over tools:

$$\text{Diversity} = \frac{H(\pi)}{\log |A|} \in [0, 1]. \tag{31}$$

E.2 Results

Table 4 shows that pheromone guidance improves performance over $\beta = 0$, while excessive guidance ($\beta = 5$) harms both Match R. and TAcc. Our dynamic schedule achieves the best results, indicating that gradually increasing pheromone influence matches the increasing reliability of accumulated pheromone statistics.

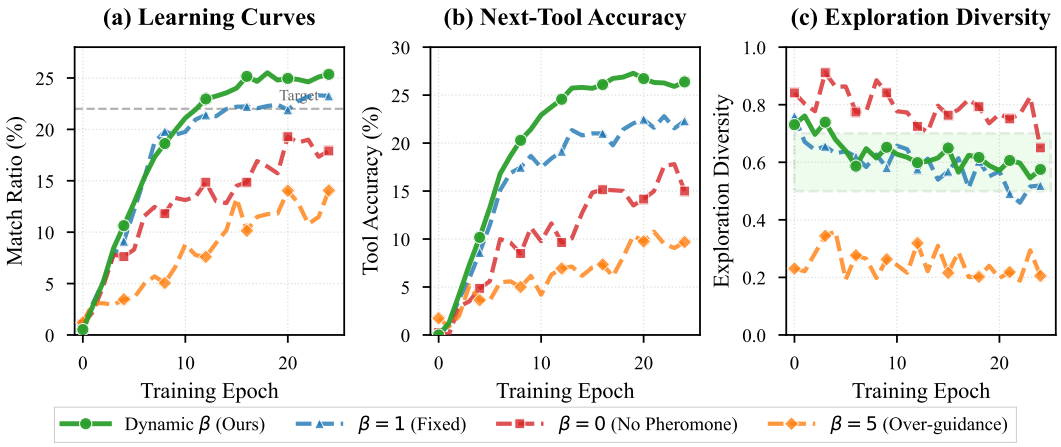

Figure 4: **Ablation study of pheromone influence parameter β on Toolathlon benchmark.** (a) **Learning curves** show that dynamic β annealing achieves the highest Match Ratio (25.25%), significantly outperforming fixed strategies. $\beta = 0$ (no pheromone) exhibits slow, noisy convergence, while $\beta = 5$ (over-guidance) becomes trapped in local optima. (b) **Next-tool accuracy** mirrors learning trends, with dynamic β reaching 27.16% through effective balance of exploration and exploitation. (c) **Exploration diversity** exhibits natural fluctuations due to finite-sample estimation from rollouts, but reveals clear trends: $\beta = 0$ maintains excessively high diversity (unfocused exploration), $\beta = 5$ shows extremely low diversity (trapped exploitation), while dynamic β demonstrates an optimal trajectory—starting with high exploration (0.75) and smoothly transitioning into the optimal range (0.55-0.65) that balances verified pattern reuse with adaptive exploration.

Table 4: Ablation on β for Toolathlon (Qwen2.5-7B). We report Match Ratio (Match R., %), Next-tool Accuracy (TAcc, %), and final Exploration Diversity.

Setting	Match R.	TAcc	Final Diversity
$\beta = 0$ (No pheromone)	21.23	19.86	0.76
$\beta = 1$ (Fixed)	23.93	22.05	0.53
$\beta = 5$ (Over-guidance)	17.80	16.50	0.25
Dynamic β (Ours)	**25.25**	**27.16**	0.58

Interpretation. With $\beta = 0$, the policy remains highly diverse but learns slowly, suggesting unfocused exploration. With $\beta = 5$, the policy becomes nearly deterministic and underperforms, consistent with premature over-exploitation of early transition patterns. Fixed $\beta = 1$ provides a stable improvement over $\beta = 0$, but it cannot adapt to the evolving quality of pheromone statistics. Dynamic β preserves early diversity for discovery and then reduces diversity to a stable mid-range as training progresses, aligning improved Match R. with higher TAcc. Variance in the diversity curve is expected because it is estimated from a small number of rollouts per epoch ($M = 5$), while Match R. and TAcc average across trajectories and evaluation instances.

Pheromone guidance benefits long-horizon tool planning, but its strength must be calibrated. Extreme settings either remove reusable transition information ($\beta = 0$) or suppress exploration ($\beta = 5$). A progressive schedule for β yields the best balance, improving both sequence-level Match R. and step-level TAcc.

F Training Dynamics and Pheromone Evolution

Training dynamics. To assess optimization stability, we visualize the evolution of policy performance together with the evolution of the pheromone structure in Figure 5. We report the *average return* measured over training rollouts for each backbone. Figure 5(a) shows a consistent upward trend across all four backbone models, indicating that PhGPO steadily improves the rollout-level objective. The smallest backbone, Qwen2.5-1.5B, reaches competitive returns compared with larger models, which

suggests that the reusable tool-transition patterns reduces the difficulty of long-horizon exploration for lower-capacity policies. The curves exhibit sustained improvement without abrupt collapses, which is consistent with stable optimization under our training configuration.

Pheromone graph growth. Figure 5(b) tracks the growth of the pheromone graph, measured by the number of discovered edges that receive pheromone statistics. These edges correspond to tool-transition edges, and they can also include argument-invocation edges when tool invocations are represented at the argument level. We observe rapid edge discovery during early epochs, followed by a clear saturation. This trend indicates that the agent quickly identifies a broad set of feasible tool transitions and then concentrates updates on a more stable subset. Under pheromone evaporation, older statistics gradually lose influence, which keeps the explicit transition prior responsive to recent verified tool-use trajectories while preventing unbounded accumulation.

Relationship between returns and pheromone. The increase in average return coincides with the period where the pheromone graph expands and then stabilizes. This alignment suggests that improving performance is accompanied by accumulating transition evidence on the pheromone graph. Once most frequently used transitions have been identified, performance improvements continue primarily through adjusting the relative strengths of already discovered edges and improving the policy probabilities on those transitions.

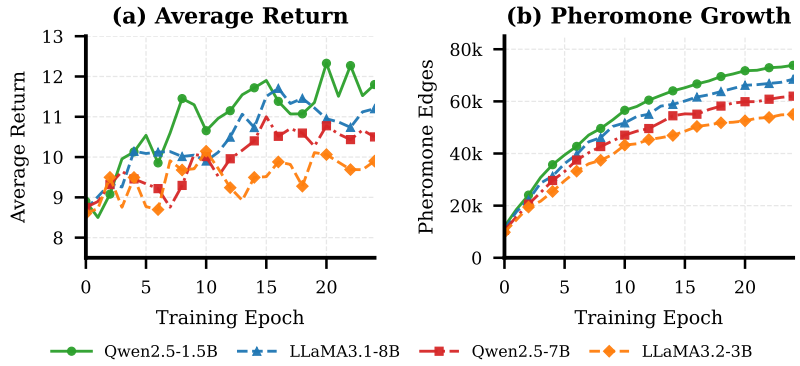

Figure 5: **Training Dynamics of PhGPO.** (a) **Average Return**: All backbone models show a consistent upward trend in average return, indicating stable policy improvement during training. (b) **Pheromone Graph Growth**: The number of discovered edges grows rapidly in early epochs and stabilizes later, reflecting fast discovery of feasible tool transitions followed by refinement on a stable set of edges.

G Hyperparameter Sensitivity Analysis

This appendix reports a sensitivity analysis of eight key hyperparameters in PhGPO on Toolathlon. We vary one hyperparameter at a time while keeping all others fixed to the default values in Table 5. We train with Qwen2.5-7B and report Match Ratio on the test set as a fine-grained sequence-level diagnostic. The main experiments report both Match Ratio and Task Success Rate, while this sensitivity analysis uses Match Ratio because it provides a continuous signal for comparing hyperparameter trends. Each configuration is evaluated with 3 random seeds, and Figure 6 shows the mean with ± 1 standard deviation. Red stars and dashed lines indicate our chosen values.

Hyperparameters. We group the parameters into pheromone guidance and policy optimization. Pheromone guidance includes the maximum pheromone influence $\beta_{\max}$ in Eq. (7), evaporation rate ρ and deposition rate α in Eq. (1), and the task-dependent pheromone weight w in Eq. (5). Policy optimization includes GRPO group size M, RL fine-tuning learning rate η, final teacher forcing probability p_{tf}, and entropy regularization coefficient γ.

Overall observations. Figure 6 shows that our defaults fall within, or very close to, the best-performing regions across all eight parameters. Among pheromone parameters, $\beta_{\max}$ and ρ are the most sensitive. A small $\beta_{\max}$ weakens the influence of the explicit transition prior, while an overly large $\beta_{\max}$ can suppress exploration. The evaporation rate ρ performs best around 0.01, where older

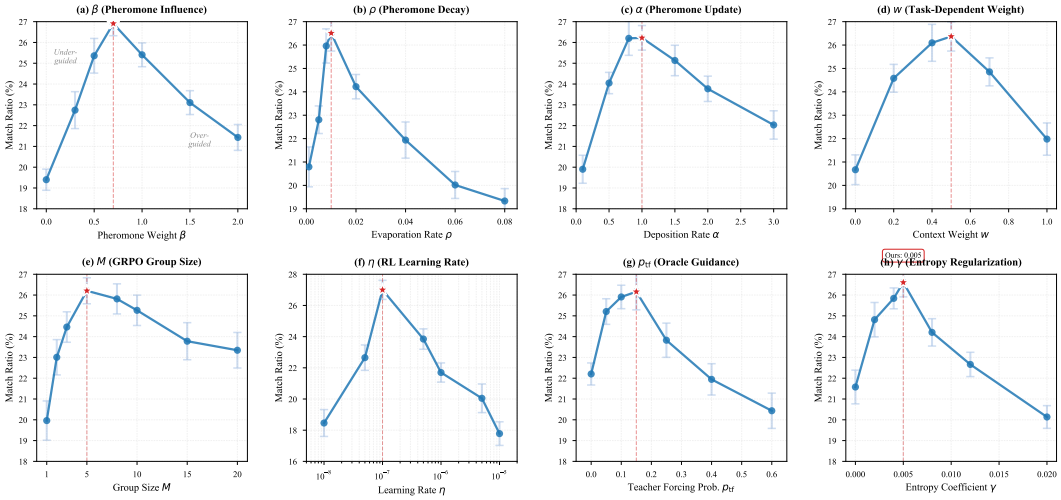

Figure 6: **Hyperparameter sensitivity analysis on Toolathlon.** We evaluate 8 hyperparameters for pheromone guidance and policy optimization. Red stars indicate our selected values; shaded regions show ± 1 standard deviation across 3 seeds. **Row 1:** (a) $\beta_{\max}$; (b) ρ; (c) α; (d) w. **Row 2:** (e) M; (f) η; (g) p_{tf}; (h) γ.

Table 5: Key hyperparameters in PhGPO and selected values.

Category	Parameter	Value	Description
Pheromone	$\beta_{\max}$	0.8	Maximum pheromone influence weight in Eq. (7)
	ρ	0.01	Evaporation rate in Eq. (1)
	α	1.0	Deposition rate in Eq. (1)
	w	0.5	Task-dependent pheromone weight in Eq. (5)
Optimization	M	5	GRPO group size, measured as rollouts per instance
	η	10^{-7}	Learning rate for RL fine-tuning
	p_{tf}	0.15	Final teacher forcing probability
	γ	0.005	Entropy regularization coefficient

statistics decay slowly enough to remain useful but not so slowly that early transitions dominate. In contrast, α is comparatively robust, with a broad near-optimal region around 1.0. The task-dependent pheromone weight w peaks around 0.5, indicating that combining task-dependent and task-agnostic pheromones is more effective than relying on either source alone.

For optimization parameters, M shows diminishing returns. Performance improves rapidly from $M = 1$ and saturates around $M = 5$. The RL learning rate η has a narrow optimum on a log scale, consistent with the need for conservative fine-tuning in the final stage. The final teacher forcing probability p_{tf} follows a U-shaped pattern, where too little oracle guidance destabilizes early pheromone accumulation and too much oracle guidance limits autonomous improvement. The entropy coefficient γ has a broad plateau around 0.005, maintaining exploration without preventing convergence.

Practical guidance. When adapting PhGPO, tuning ρ and $\beta_{\max}$ is most important because they directly control the strength and persistence of pheromone guidance. The remaining parameters have wider stable regions, and our defaults provide a strong starting point. Progressive schedules for β, w, and p_{tf} are consistent with these sensitivity trends because they avoid applying strong pheromone guidance before the corresponding statistics become reliable.

H Generality Across Policy Optimization Backbones

This appendix shows that PhGPO is not tied to a specific policy optimization backbone. PhGPO defines (i) pheromone as an explicit transition prior on tool-transition edges and tool-to-invocation edges (Sec. 3.2), (ii) pheromone-guided sampling during rollout generation (Eq. (7)), and (iii)

pheromone updates via deposition and evaporation (Eq. (1) and Eq. (5)). These components are independent of the choice of the policy-gradient optimizer used to update π_θ. In practice, we can replace GRPO with other on-policy optimizers, while keeping the same pheromone-guided rollout distribution and pheromone bookkeeping.

H.1 GRPO, PPO, and RLOO backbones

We describe three interchangeable optimization backbones under the same rollout protocol. For each task instance, we sample M tool-use trajectories from the pheromone-guided distribution in Eq. (7), obtain sequence-level returns $\{R_m\}_{m=1}^{M}$, and update the policy using a clipped policy-gradient objective. Across all three backbones, pheromone-guided sampling and pheromone updates remain unchanged.

GRPO (Group Relative Policy Optimization). GRPO uses group-relative normalization to construct a sequence-level advantage for each rollout:

$$A_m = \frac{R_m - \bar{R}}{\sigma_R + \epsilon}, \qquad \bar{R} = \frac{1}{M}\sum_{m=1}^{M} R_m, \tag{32}$$

and applies a PPO-style clipped ratio objective with A_m (Sec. 3.3). This backbone does not require an explicit value function and leverages within-group contrasts.

PPO (Proximal Policy Optimization). PPO constructs an advantage using a baseline $b(\cdot)$, which can be implemented as a learned value head or a running baseline. In our setting, we use a sequence-level advantage

$$A_m = R_m - b(x), \tag{33}$$

and optimize the standard clipped surrogate objective:

$$\mathcal{L}^{\text{PPO}} = -\mathbb{E}\Big[\min\big(r_t(\theta)A_m,\ \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)A_m\big)\Big], \tag{34}$$

where $r_t(\theta) = \pi_\theta(a_t \mid s_t)/\pi_{\theta_{\text{old}}}(a_t \mid s_t)$ is the probability ratio at tool-decision steps.

RLOO (Leave-one-out baseline). RLOO avoids learning a value function and uses a leave-one-out baseline computed from the other rollouts in the same group:

$$b_{-m} = \frac{1}{M-1}\sum_{j\neq m} R_j, \qquad A_m = R_m - b_{-m}. \tag{35}$$

The policy is then updated using the same clipped objective as Eq. (34), with A_m replaced by the leave-one-out advantage.

H.2 Experimental protocol on Toolathlon

We evaluate optimizer-agnosticity on Toolathlon using Qwen2.5-7B and Llama3.1-8B. All runs use the same action space, pheromone-guided sampling rule (Eq. (7)), pheromone update rule (Eq. (1)), and the same progressive pipeline in Sec. 3.3. The only change is the policy optimization backbone used to update π_θ. We report sequence-level Match Ratio (Match R., %) and Next-tool Accuracy (TAcc, %).

I Extended Visualization on Longer Tool-use Trajectories

This appendix provides an extended visualization showing that the pheromone-based explicit transition prior remains informative on longer tool-use trajectories. For readability, we visualize a *10-tool subset* of the tool-transition space, while the underlying episode involves a longer reference tool-use trajectory in the same MCP execution environment. All visualizations report the *fused* tool-transition pheromone $\tau_{\text{Tool}}(\cdot \mid x)$ defined in Eq. (5), where task-dependent evidence is retrieved by Eq. (3) and confidence is computed by Eq. (4). Pheromone updates follow deposition and evaporation in Eq. (1).

Table 6: **PhGPO with different policy optimization backbones on Toolathlon.** For each backbone optimizer, we report results with pheromone-guided sampling (Eq. (7)) and without pheromone guidance by setting β=0 in sampling. Higher is better (↑).

Optimizer variant	Qwen2.5-7B		Llama3.1-8B	
	Match R. ↑	TAcc ↑	Match R. ↑	TAcc ↑
PhGPO + GRPO	25.25	27.16	22.83	23.09
w/o pheromone guidance (β=0)	21.23	19.86	18.75	16.45
PhGPO + PPO	23.27	25.03	18.21	24.05
w/o pheromone guidance (β=0)	20.78	23.31	15.72	21.22
PhGPO + RLOO	20.21	24.47	18.51	22.69
w/o pheromone guidance (β=0)	18.61	23.26	17.39	21.51

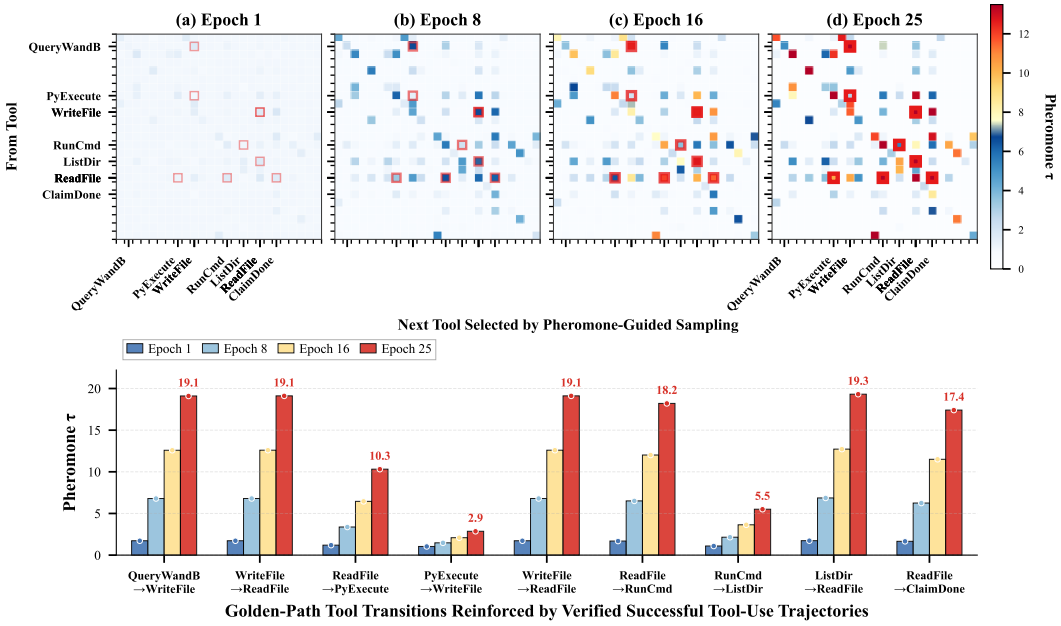

Figure 7: **Pheromone evolution on a longer tool-use trajectory, visualized on a 10-tool subset.** Top: fused tool-transition pheromone $\tau_{\text{Tool}}(\cdot \mid x)$ restricted to ten tools at checkpoint, with reference-chain edges highlighted in red. Bottom: pheromone values on the highlighted reference-chain transitions at the same checkpoint. The visualization shows that pheromone concentrates on reference-chain transitions while off-chain transitions fade under evaporation, indicating that the explicit transition prior remains selective on longer tool-use trajectories.

10-tool subset visualization Figure 7 visualizes pheromone evolution on a representative long-horizon instance using the following 10 tools:

`QueryWandB` → `WriteFile` → `ReadFile` → `PyExecute` → `WriteFile` → `ReadFile` → `RunCmd` → `ListDir` → `ReadFile` → `ClaimDone`

The heatmaps (top row) show the fused pheromone matrix restricted to these tools at several training checkpoints. Reference-chain transitions that appear in this instance are highlighted in red. As training proceeds, pheromone mass becomes increasingly concentrated on the reference-chain transitions within this 10-tool subset, while many off-chain transitions fade as evaporation reduces the influence of older or weakly supported statistics. This pattern indicates that the explicit transition prior becomes more selective over time, amplifying transitions that recur in verified successful tool-use trajectories.

The bar plot (bottom row) reports pheromone values along the reference-chain transitions within the same 10-tool subset at the corresponding checkpoints. Earlier transitions often exhibit stronger

reinforcement because they are visited more frequently across verified successful trajectories and because early tool choices constrain later reachable states in long tool-use trajectories. Later transitions remain distinguishable from the background, which indicates that the pheromone signal is not limited to prefix segments and can provide guidance throughout the chain.

J Time Efficiency: Path Discovery Speed

This appendix provides a qualitative analysis of training-time efficiency. We compare PhGPO against an ablation that removes the pheromone prior, and examine how quickly each method discovers the first high-quality tool-use trajectory on Toolathlon episodes. A trajectory is considered high-quality when its pheromone-deposition score satisfies $q_\tau(\xi) \geq 0.6$.

Experimental protocol We evaluate two configurations on Toolathlon with Qwen2.5-7B:

- **PhGPO (Full)**: rollouts are sampled from the fused distribution that incorporates pheromone as an explicit transition prior (Eq. (7)), with the progressive schedule described in Sec. 3.3.
- **GRPO w/o pheromone**: we set $\beta = 0$ throughout training, removing pheromone guidance during rollout generation while keeping the remaining pipeline identical (same curriculum, the same oracle-mixing rule via p_{tf}, and the same mixed objective).

For each Toolathlon episode, we record the earliest training step at which at least one generated trajectory reaches $q_\tau(\xi) \geq 0.6$, and summarize the trend across episodes and trajectory-length bins.

Results. Figure 8 summarizes the path discovery speed measured by the first occurrence of a high-quality trajectory ($q_\tau(\xi) \geq 0.6$). In panel (a), PhGPO maintains a higher high-quality match rate than the ablation without pheromone at matched training steps, and it reaches the same coverage level with fewer training steps. The gap remains visible across the training range shown in the figure, indicating more efficient discovery of high-quality trajectories under the same budget. Panel (b) reports the relative speedup across trajectory-length bins. The speedup stays positive across bins and varies within a narrow range, and the fitted linear trend is close to flat. This pattern suggests that the efficiency gain is not restricted to a specific horizon, and it remains broadly stable as the average trajectory length increases.

Interpretation The trend is consistent with the role of pheromone as an explicit transition prior during trajectory generation. Without pheromone ($\beta = 0$), rollout groups sampled only from the policy can be dominated by uniformly low-quality trajectories in a large tool space, which weakens group-relative learning signals. With pheromone guidance, historically verified tool-transition edges receive higher sampling probability, which reduces wasted exploration on low-potential transitions and increases the likelihood that each rollout group contains informative contrasts for group-based policy optimization. This mechanism can benefit from partial matches: trajectories that reach $q_\tau(\xi) \geq 0.6$ indicate reusable transition evidence, and verified successful ones are used for pheromone updates, even when they are not exact matches to the reference.

Implications These results indicate that PhGPO not only improves final performance, but also improves training-time efficiency by accelerating the discovery of high-quality tool-use trajectories. This property is particularly relevant in execution-based environments where tool calls are costly.

K Simulated Tool Execution

Our training and evaluation use a *simulated* tool environment. Tool calls are not executed against real external APIs or services. Instead, each tool invocation is routed to a tool-output simulator that returns a plausible response conditioned on the current context. This design provides a controlled and reproducible feedback channel for learning tool-use trajectories, while avoiding external side effects and service availability issues.

K.1 LLM-based tool-output simulator

We implement an LLM-based simulator that generates tool outputs from structured context. Given a tool variant name, its arguments, the task name, the user request, and a truncated history of recent tool interactions, the simulator constructs a prompt that includes: (i) the task and user request, (ii)

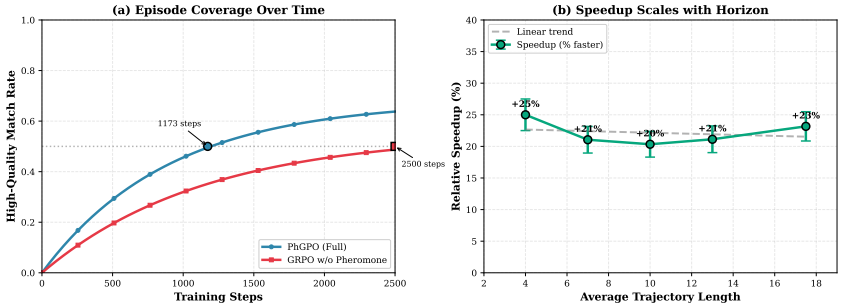

Figure 8: **Time efficiency analysis on Toolathlon.** We track the earliest training step at which a high-quality trajectory ($q_\tau(\xi) \geq 0.6$) is first generated for each episode. PhGPO reaches broad coverage earlier than the ablation without pheromone, and the relative improvement remains broadly consistent across trajectory-length bins.

the recent tool-call history (with any completion markers removed), (iii) the current tool name, its original tool name, and its natural-language description, (iv) the provided arguments (or expected keys when arguments are absent). The simulator then queries an instruction-following model to produce a short response that resembles a real tool return value (for example, a JSON snippet, a status string, or a concise result summary).

In our experiments, we use `qwen3-235b-a22b-thinking-2507` as the simulator model through an OpenAI-compatible chat-completions endpoint. We fix the simulator system prompt to enforce output-only behavior, and we strip common explanatory prefixes and markdown formatting during post-processing to keep outputs consistent with a tool API style.

K.2 Completion detection via an explicit marker

The simulator also provides a lightweight termination signal. After generating a tool output, the simulator decides whether the current interaction history already completes the user request. If it judges the task complete, it appends a dedicated marker `«END»` on a separate final line. We remove this marker from the returned tool output and store the completion flag separately. This enables consistent stopping behavior during rollout generation without requiring an additional model call.

K.3 Caching and fallback behavior

To reduce simulator calls and stabilize training, we cache simulator outputs using a content hash over the tool variant, task name, user request, truncated history, and arguments. When the LLM simulator is unavailable or fails, we optionally fall back to a pool-based simulator that samples from pre-defined outputs, and we use a simple default template as a final fallback. These mechanisms preserve the tool-use interface while keeping the environment feedback stable under large-scale training.

L Adaptation to Dynamic Tool Environments

A practical advantage of PhGPO is that the transition prior is maintained explicitly on the MCP-grounded graph and its edge memories, instead of being absorbed only into policy parameters. This design allows tool-environment changes to be handled through local graph and memory updates without global re-initialization.

When a new tool is introduced, we add the corresponding tool node, argument-invocation patterns, tool-transition edges, and tool-to-invocation edges. The task-agnostic pheromone values on newly added edges are initialized to τ_0, and their memory banks are initialized as empty. If the new tool is not yet covered by updated reference trajectories or sampled rollouts, its associated edges remain at τ_0 with empty memory banks. Statistics start to accumulate only after subsequent trajectories traverse these edges. In many practical settings, newly introduced tools are accompanied by argument schemas and a small number of updated demonstrations. When such data are available, the same supervised

warm-up can warm-start the corresponding tool and invocation edges without re-initializing the whole policy.

When a tool is deprecated, the corresponding node and edges can be immediately masked from the action space. Existing pheromone values and memory entries for the deprecated tool are no longer queried during rollout generation. Evaporation further reduces the influence of stale statistics over time. If the tool identity remains but its API schema changes, the update mainly affects the argument-invocation patterns and the corresponding tool-to-invocation edges, while unaffected tool-transition edges can be preserved.

We validate this behavior with a tool-evolution test on Toolathlon using Qwen2.5-7B. We merge several frequent atomic tool chains into macro tools that replace the original chains, simulating a realistic API upgrade. Under this change, local graph and memory update recovers to about 87% of the pre-change performance using about 16% of the original RL training steps. By comparison, global re-initialization requires about 38% of the original RL training steps to reach the same recovery level. Performance on unaffected tasks remains stable under local update. These results indicate that the explicit graph-indexed transition prior helps PhGPO adapt to evolving tool environments with limited retraining.

M Scalability and Computational Overhead

PhGPO adds three main components on top of GRPO: graph maintenance, pheromone-guided sampling, and task-dependent edge-memory retrieval. These components are lightweight because pheromone values are indexed by graph edges and queried only for the current candidate set during rollout generation. PhGPO does not require inference-time tree search, additional full-trajectory rollouts, or LLM-based judging during training.

We benchmark the end-to-end overhead of these components on Toolathlon with Qwen2.5-7B. Table 7 reports training-step time, peak GPU memory, and inference-step latency. The full PhGPO pipeline uses 11.89s per training step compared with 11.86s for GRPO, and 59.3ms per inference step compared with 58.5ms for GRPO. The peak GPU memory increases from 17.3GB to 17.4GB. The incremental rows show that graph maintenance, pheromone-guided sampling, and edge-indexed retrieval each add only marginal overhead. In practice, runtime remains dominated by rollout generation and the forward/backward computation of the base model.

Table 7: **Runtime and memory overhead on Toolathlon.** We report end-to-end overhead using Qwen2.5-7B.

Method	Train step (s)	Train overhead	Peak GPU (GB)	Infer/step (ms)	Infer overhead
GRPO	11.86	baseline	17.3	58.5	baseline
+ graph maintenance	11.87	+0.01s	17.3	58.8	+0.3ms
+ graph + pheromone sampling	11.88	+0.02s	17.4	59.1	+0.6ms
Full PhGPO	11.89	+0.03s	17.4	59.3	+0.8ms

The small additional cost comes from local operations on a sparse tool-transition graph. Graph maintenance updates only traversed edges, pheromone-guided sampling reweights the Top-K candidate set, and task-dependent retrieval is performed over edge-indexed memory banks. These results suggest that PhGPO scales favorably in large tool spaces because it avoids global search over all possible tool sequences.

N Random Exploration and Pheromone Guidance

Both random exploration and pheromone guidance affect the exploration-exploitation trade-off, but they play different roles. The random exploration probability p_{rand} introduces a small memoryless perturbation at the current decision step. In contrast, pheromone is a history-informed explicit transition prior accumulated across episodes. It biases rollout generation toward tool-transition and argument-invocation patterns that repeatedly appear in verified successful trajectories. Therefore, p_{rand} provides undirected local exploration, while pheromone provides directed cross-episode exploration.

Table 8: **Random exploration and pheromone guidance on Toolathlon.** We compare the effects of removing pheromone guidance and changing the random exploration probability p_{rand} using Qwen2.5-7B.

Setting	β **schedule**	p_{rand}	**Match R.** $\uparrow$	**TAcc** $\uparrow$	**Final Diversity**
No pheromone	0	0.05	21.23	19.86	0.76
Full model w/o random exploration	dynamic β	0	24.63	26.34	0.52
Full model	dynamic β	0.05	25.25	27.16	0.58
Full model, larger random exploration	dynamic β	0.10	24.38	26.47	0.66

To quantify their relative effects, we compare four settings on Toolathlon with Qwen2.5-7B. The results are shown in Table 8. Removing pheromone causes the largest drop in Match Ratio and Next-tool Accuracy while keeping diversity high, which is consistent with unfocused exploration. Removing random exploration causes only a small decrease, indicating that pheromone is the primary source of effective exploration guidance. Increasing p_{rand} from 0.05 to 0.10 increases diversity but slightly hurts performance, suggesting that stronger random perturbation can dilute the explicit transition prior.

Overall, pheromone is the primary mechanism governing directed exploration in PhGPO, while p_{rand} serves as a lightweight auxiliary safeguard against early over-concentration. This distinction also explains why the two mechanisms are complementary: pheromone reuses accumulated transition evidence, whereas random exploration preserves a small chance of escaping over-concentrated paths.

O Broader Impact

This work studies long-horizon tool planning for LLM agents in benchmarked tool-use environments. Its potential positive impact is to improve the reliability and training efficiency of tool-use agents by making successful transition patterns explicit and reusable. More reliable tool planning may benefit applications that require multi-step interaction with software tools, such as information retrieval, code execution, data processing, and workflow automation.

At the same time, stronger tool-use agents should be deployed with care. If connected to real external APIs, an agent may execute incorrect, unintended, or unsafe tool calls, especially when task instructions are ambiguous or tool outputs are unreliable. Such risks can be more serious in sensitive domains involving personal data, security-relevant operations, financial actions, or medical and legal decisions. PhGPO does not remove the need for tool-level safeguards, access control, or human oversight.

We therefore view PhGPO as a method for improving tool-use training in controlled environments, not as a complete deployment framework. Practical deployments should use sandboxed execution, permission-gated tools, audit logs, tool allowlists, validation checks, and human confirmation for high-impact actions. The pheromone memory should also be monitored when tool interfaces or task distributions change, since accumulated transition statistics may otherwise reflect outdated tool-use patterns. Our experiments are conducted in benchmarked tool environments rather than real-world high-stakes deployments.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 3 state that PhGPO learns a pheromone-based explicit transition prior from verified successful tool-use trajectories and uses it to guide policy optimization. The scope is limited to long-horizon tool planning, and the empirical claims are supported by the results in Sec. 4.2 on Toolathlon, TRAJECT-Bench, and TOUCAN.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses future directions in Sec. 5 and provides additional analysis in Appendix L on evolving tool environments and Appendix M on scalability and computational overhead. These sections clarify the scope of the method and its behavior under tool updates and large tool spaces.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The paper does not present formal theoretical theorems or proofs. The contribution is algorithmic and empirical. The method formulation is provided in Sec. 3, including the tool-transition graph, pheromone update rule, task-dependent pheromone estimation, and GRPO-based training objective.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. 4.1 describes the datasets, baselines, evaluation metrics, and implementation details. Appendix B provides algorithm implementation details, training mechanisms, reward signals, pheromone bookkeeping, and simulator setup. The anonymized code repository is provided at `https://github.com/ElevenLiy/PhGPO-Pheromone-Guided-Policy-Optimization-for-Long-Horizon-Tool-Planning`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides an anonymized code repository at `https://anonymous.4open.science/r/PhGPO-Pheromone-Guided-Policy-Optimization-for-Long-Horizon-Tool-Planning-CB1C/`. The repository contains the implementation of PhGPO, training scripts, evaluation scripts, and supporting code for reproducing the reported experiments. Appendix B describes the training pipeline and implementation details.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 4.1 specifies the datasets, benchmark splits, backbone models, baselines, metrics, LoRA setup, decoding configuration, maximum trajectory length, rollout group size, and pheromone scheduling. Appendix B provides the full training mechanism, and Appendix G reports the selected hyperparameters and sensitivity analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Appendix G reports hyperparameter sensitivity with mean performance and ± 1 standard deviation across three random seeds. The figure caption explicitly states that shaded regions indicate ± 1 standard deviation. The main comparison uses fixed test splits and reports deterministic benchmark-level metrics.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix M reports training-step time, inference-step latency, and peak GPU memory for GRPO and PhGPO components on Toolathlon with Qwen2.5-7B. The code repository provides scripts and configuration files for reproducing the experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The work uses public or benchmarked tool-use datasets and does not involve human-subject data collection, private personal data, or deployment in real-world high-stakes settings. Appendix O discusses responsible use considerations for stronger tool-use agents.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Appendix O discusses potential positive impacts of improving reliable long-horizon tool planning, as well as possible risks from unsafe or unintended tool execution when agents are connected to real APIs. It also discusses mitigation strategies such as sandboxed execution, permission-gated tools, audit logs, tool allowlists, validation checks, and human oversight.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper releases code for a training and evaluation framework, but does not release a new pretrained foundation model, scraped dataset, or high-risk generative model. Appendix O nevertheless discusses safeguards that should be used when tool-use agents are connected to real external APIs.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original sources for all benchmarks, backbone models, and relevant methods in Sec. 4.1 and the references. The experiments use existing benchmark datasets and open model backbones according to their intended research use. The released code repository includes accompanying documentation for reproducing the experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The main new asset is the anonymized implementation of PhGPO. The code is released at `https://anonymous.4open.science/r/PhGPO-Pheromone-Guided-Policy-Optimization-for-Long-Horizon-Tool-Planning-CB1C/`, and Appendix B documents the algorithm implementation, training pipeline, reward signals, pheromone memory, and simulator setup.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or research with human subjects. The experiments are conducted on benchmarked tool-use datasets and simulated or benchmarked tool execution environments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, private user data, or crowdsourced human annotations. Therefore, IRB approval or equivalent human-subjects review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The paper explicitly studies LLM-based tool-use agents and uses Qwen2.5-7B and Llama3.1-8B as policy backbones in Sec. 4.1. It also evaluates Task Success Rate using GPT-4o as an outcome judge under a fixed protocol, described in Sec. 4.1 and Appendix D.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.