

DUPO: Dual-Mode Unified Policy Optimization for Tool-Using Agents

Zexin Hu^{1,2}, Mu Hu¹, Lijuan Yang¹, Yisen Wang¹, Xiaogang Zhu³,
Kangjian Jiao¹, Yu Fu¹, Lihong Gu¹, Zhiyong Wang², Kun Hu^{4,*}

¹Ant Group, ²The University of Sydney, ³Adelaide University, ⁴Edith Cowan University

{huzexin.hu,daiwan.hm,qianru.ylj,wangyisen.wys,jiaokangjian.jkj,fuyu.fy,lihong.glh}@antgroup.com
xiaogang.zhu@adelaide.edu.au, zhiyong.wang@sydney.edu.au, k.hu@ecu.edu.au

Abstract

Tool-using large language models (LLMs) have improved at producing executable API calls, but reliable tool use requires coordinating *semantic intent* with *executable invocation*, especially in multi-step settings. Existing tool-learning objectives and traces collapse these two roles into a single homogeneous generation stream, so the learning signal cannot distinguish a flawed plan from a flawed call. We propose *Dual-Mode Unified Policy Optimization* (DUPO), a typed trajectory optimization framework that represents each step with a PLAN span for the current subgoal and a TOOL span for the API invocation. DUPO follows a PLAN-to-TOOL rollout factorization, optimizes the two modes with mode-separated policy ratios, and introduces the Chain-Aware Upstream Advantage (CAUA), a position prior that gives stronger weight to upstream decisions because their errors can invalidate downstream actions during autoregressive inference. To support this optimization, we introduce *PlanArena*, a human-annotated dataset of 20,646 trajectories across 7,941 scenarios with explicit PLAN-TOOL supervision. Experiments on BFCL, API-Bank, and Tau2 show that DUPO improves tool-use performance, with gains concentrated in multi-step and multi-turn settings. Our project page is available at <https://dupo-project.github.io/>

1 Introduction

Tool-using large language models (LLMs) extend language generation into interactive problem solving by invoking external APIs and conditioning on returned observations (Qin et al., 2024; Schick et al., 2023). Recent progress in function calling has improved the ability of LLMs to produce executable API invocations (Liu et al., 2024; Wang et al., 2024). Yet reliable tool use requires more than formatting a valid call: before acting, an agent

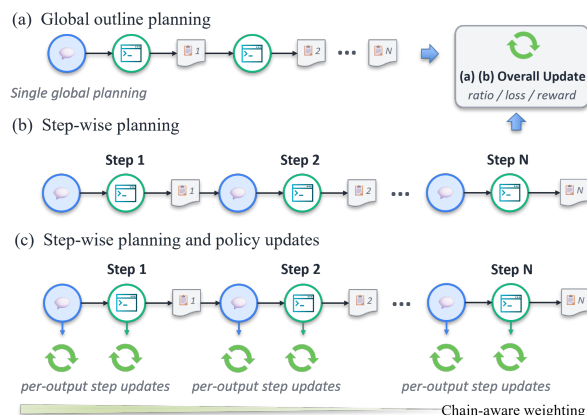


Figure 1: Comparison of trajectory optimization. Global tool planning (a) applies one update to the full trajectory, while step-wise planning (b) interleaves planning and tool invocation without step-wise updates. DUPO and PlanArena (c) enable step-wise PLAN-TOOL generation with per-output policy updates.

must plan the intended subgoal and ground that intent into an appropriate API invocation.

This distinction becomes especially important in multi-step tool orchestration. At each step, the agent must coordinate two heterogeneous components: semantic intent about the current subgoal and a formal API invocation that realizes this intent. We represent them as PLAN and TOOL spans, respectively: PLAN tokens specify what the agent intends to do, while TOOL tokens must satisfy schema-level constraints such as function selection, argument structure, and validity. As shown in Figure 1, existing learning objectives and tool-use traces often collapse these spans into a single homogeneous generation stream (Yu et al., 2025b; Qian et al., 2025), making the learning signal less targeted for semantic planning and tool execution.

We address this issue with *Dual-Mode Unified Policy Optimization* (DUPO), a typed trajectory optimization framework for tool-using agents. During rollout, DUPO follows a PLAN-to-TOOL generation process: the policy first generates a PLAN span

*Corresponding author.

and then conditions on it to produce the TOOL span. PLAN and TOOL tokens are optimized with separate policy ratios within a unified objective. For multi-step learning, the Dual-Mode Reward scores PLAN and TOOL fidelity, and the Chain-Aware Upstream Advantage (CAUA) applies a position-dependent anchor that prioritizes upstream decisions.

To support this optimization, we construct *PlanArena*, a human-annotated dataset for multi-step tool orchestration, containing 20,646 trajectories across 7,941 scenarios with up to 9 tool-use steps per trajectory. Unlike execution-only traces that primarily record API invocations, PlanArena pairs each TOOL span with a natural-language PLAN span and records the corresponding execution/status output. This provides explicit supervision for aligning intended subgoals with executable tool invocations across multi-step trajectories.

We evaluate DUPO on common tool-use benchmarks: BFCL (Patil et al., 2025), API-Bank (Li et al., 2023), and Tau2 (Barres et al., 2025). Experiments show that DUPO improves tool-use performance across backbones, with gains concentrated in multi-step and multi-turn settings.

Our core contributions are as follows:

- We identify PLAN–TOOL entanglement as a key challenge in tool-learning optimization.
- We propose DUPO, a typed trajectory optimization framework with mode-separated PLAN and TOOL updates.
- We propose PlanArena, a human-annotated multi-step PLAN–TOOL dataset.
- We show that DUPO improves tool-use performance, especially in multi-step scenarios.

2 Related Work

Plan–tool structure in tool trajectories. As tool-use horizons grow (Qian et al., 2025), recent work introduces explicit planning into tool trajectories. *Plan-and-Act* (Erdogan et al., 2025) and *Pre-Act* (Rawat et al., 2025) generate high-level plans before execution, reducing redundant calls and dead-end branches. However, plan and tool tokens are treated in these methods with the same role-agnostic next-token objective, providing limited direct optimization pressure for the plan to shape subsequent tool selection. Another line moves plan–tool separation into training. *CoDA* (Liu et al., 2026) keeps a single backbone but routes planner

and executor through segregated context windows under a trajectory-level reward, while *RLTR* (Li et al., 2025) trains the planner in isolation by masking tool outputs from the loss, so the planner is not optimized against the execution stream it induces. DUPO instead keeps PLAN and TOOL generation in the same rollout and uses their span boundaries to enable mode-separated policy updates and reward signals within a unified policy.

Rewards and credit assignment for multi-step tool use. Reward design is another key challenge in multi-step tool learning (Liu et al., 2025; Qin et al., 2025; Qian et al., 2023). Outcome-only signals (Qin et al., 2024) are too sparse to guide intermediate decisions (Qian et al., 2023), motivating dense step-level rewards in methods such as *Step-Tool* (Yu et al., 2025b) and *ToolRL* (Qian et al., 2025), often combined with trajectory-level GRPO-style optimization (Shao et al., 2024). However, these rewards are typically aggregated in a position-agnostic manner. In multi-step orchestration, earlier decisions often condition the usefulness of later tool calls: a downstream invocation may be locally valid but still irrelevant if the preceding subgoal is wrong. DUPO addresses this issue by combining typed step fidelity for PLAN and TOOL spans with a chain-aware upstream advantage.

Stabilizing RL for tool-use training. RL-based optimization has become a common approach for improving tool-use agents beyond supervised imitation (Yu et al., 2025b; Qian et al., 2025). However, on-policy methods such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024) can be unstable in tool-use settings, where invalid calls or poorly grounded intermediate steps may prevent useful feedback from accumulating. Recent work addresses this instability through more conservative or hybrid updates: *DAPO* (Yu et al., 2025a) modifies clipping to control high-variance updates, while *CHORD* (Zhang et al., 2026) combines supervised and reinforcement signals. These techniques improve optimization stability, but they are largely agnostic to the PLAN–TOOL structure of tool-use trajectories. DUPO uses stabilization as part of typed trajectory optimization: its continuation schedule anchors early updates to supervised PLAN–TOOL trajectories, while mode-separated policy updates and a typed dual-mode reward, together with CAUA for multi-step credit assignment, provide structure-specific learning signals.

Table 1: Comparison of tool-learning datasets: ToolBench (Guo et al., 2024), API-Bank (Li et al., 2023), ToolAlpaca (Tang et al., 2023), MetaTool (Huang et al., 2024), APIBench (Patil et al., 2024), and PlanArena.

Dataset	#Traj.	#Scen./Cat.	Plan	H-Annot.	Multi-Turn
ToolBench	126,486	49	✗	✗	✓
API-Bank	1,888	1,000	✗	✓	✓
ToolAlpaca	3,938	50	✗	✗	✓
MetaTool	21,127	10	✗	✗	✗
APIBench	16,450	40	✗	✗	✗
<i>PlanArena</i>	20,646	7,941	✓	✓	✓

3 PlanArena: A Dataset for Complex Tool Orchestration

Standard tool-use benchmarks predominantly record API invocations while leaving the planning rationale behind each call implicit (Qin et al., 2024). Although such traces supervise executable calls, they provide limited step-level guidance for why a call is needed or how it depends on previous observations. We introduce *PlanArena*, a human-annotated dataset for multi-step tool orchestration that pairs each TOOL interaction with a natural-language PLAN span for the intended subgoal. Table 1 compares PlanArena with existing resources. Among the listed datasets, *PlanArena* is the only one that combines human annotation, explicit per-step PLAN supervision, and multi-turn coverage, supporting the study of planning and acting in complex agentic settings.

Task definition. *PlanArena* studies *multi-step tool orchestration*. Given a scenario description x with a user goal, constraints, and tool schemas, an agent completes the task through a sequence of tool-use steps. Each step contains a natural-language PLAN that states the intended subgoal and a TOOL call that executes the corresponding API invocation; the returned output or status may condition subsequent steps. Thus, PlanArena provides step-level supervision for both planning rationale and executable tool-call construction.

Dataset overview. Figure 2 summarizes *PlanArena*’s *breadth* (diverse scenario coverage) and *depth* (extended sequential horizons). The dataset contains 20,646 trajectories over 7,941 scenarios, with up to 9 tool-call steps per trajectory. Each step includes an explicit natural-language PLAN span paired with the corresponding TOOL span and execution/status record, yielding direct step-level supervision for

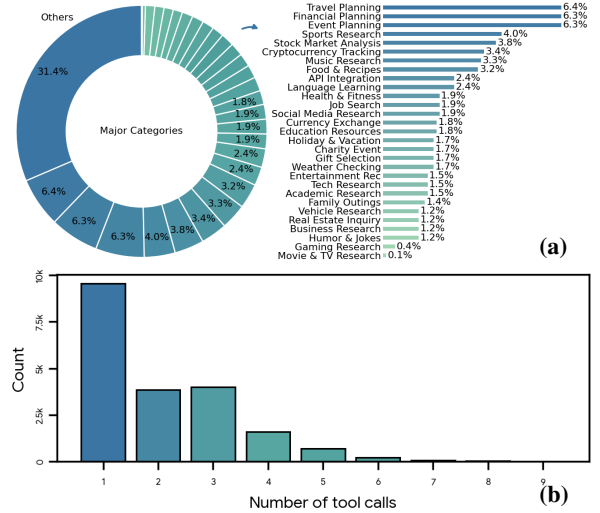


Figure 2: (a) Distribution of the 30 most frequent scenario categories in PlanArena. (b) Distribution of the number of tool-call steps.

planning and tool orchestration. PlanArena covers nearly all RapidAPI categories, averaging 265 unique scenarios per category. In our structural analysis, 82.5% of non-initial steps were marked as dependent on a preceding decision or tool result. This is a corpus-structure diagnostic rather than a full-corpus human ground-truth label, and motivates the upstream credit assignment used by DUPO.

Annotation pipeline. Annotators review each candidate trajectory by (i) writing or revising step-level PLAN rationales for tool selection, (ii) executing the API call specified by the TOOL span, and (iii) submitting the trajectory to a cross-checker who re-validates PLAN-TOOL coherence, tool-execution correctness, and consistency across steps. Before full-scale annotation, we ran a pilot on 100 trajectories with independent annotations. Cohen’s $\kappa = 0.84$ is therefore a pre-annotation calibration and qualification measure, rather than an estimate based on the entire corpus. Every trajectory then passes a final acceptance check: failures are manually corrected and reviewed by a different annotator, and only samples that pass the second review are retained. We additionally audit a random 10% of accepted trajectories at the final stage.

Splits and contamination control. PlanArena is used exclusively as a training corpus, so we do not treat it as an evaluation benchmark or report a held-out PlanArena test split. Evaluation uses the official splits of BFCL, API-Bank, and Tau2. To reduce leakage, scenarios related to these three

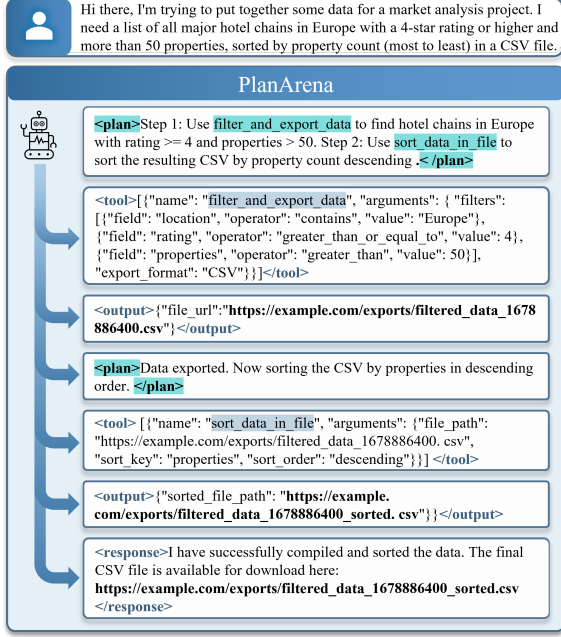


Figure 3: An example PlanArena trajectory.

benchmarks were excluded during initial scenario generation, before annotation. This separation also means that all comparisons use the same PlanArena training trajectories and the same serialized PLAN–TOOL format.

Data format. PlanArena represents a trajectory as agent-side action steps plus execution/status records. For N tool-use steps, the agent-side sequence is $\tau = \langle a_1, \dots, a_N \rangle$, with

$$a_n = (\mathcal{A}_n^p, \mathcal{A}_n^t). \quad (1)$$

Here, $\mathcal{A}_n^p$ is the PLAN span, which states the subgoal in natural language, and $\mathcal{A}_n^t$ is the TOOL span, which contains the structured API invocation.

These spans are marked by structural tags:

$$\underbrace{\langle \text{plan} \rangle \dots \langle \text{/plan} \rangle}_{\text{PLAN span } (\mathcal{A}_n^p)} \oplus \underbrace{\langle \text{tool} \rangle \dots \langle \text{/tool} \rangle}_{\text{TOOL span } (\mathcal{A}_n^t)}. \quad (2)$$

Executing the TOOL span produces an execution/status record o_n , represented by fields such as `<output>` and, when applicable, terminal `<response>`. The record o_n is appended to the trajectory history but is not part of the agent-side action a_n . We use $\mathcal{A}_n^p$, $\mathcal{A}_n^t$, and o_n throughout the paper to distinguish PLAN spans, TOOL spans, and execution/status records.

4 Methodology

Current RL training for tool-using agents often uses a homogeneous, role-agnostic objective that

updates all tokens identically. This can blur the distinction between PLAN tokens, which express the intended subgoal, and TOOL tokens, which must realize that intent as an executable API call. We propose Dual-Mode Unified Policy Optimization (DUPO), a typed trajectory optimization framework that separates PLAN and TOOL updates within a unified objective, uses typed rewards for semantic and syntactic feedback, assigns position-aware credit across steps via a Chain-Aware Upstream Advantage, and applies a continuation schedule for stable optimization.

4.1 Typed Trajectory Factorization

DUPO follows the typed trajectory format in Section 3: each agent-side step contains a PLAN span $\mathcal{A}_n^p$ and a TOOL span $\mathcal{A}_n^t$, while the environment output or status o_n is appended to the history after execution. The policy generates PLAN and TOOL spans, with outputs as conditioning context.

During each rollout, hatted symbols denote generated or observed quantities. DUPO factorizes each rollout step into a PLAN-to-TOOL generation process: given history $\hat{h}_n$, the policy first generates $\hat{\mathcal{A}}_n^p$ and then conditions on it to generate $\hat{\mathcal{A}}_n^t$. This yields the following policy factorization:

$$\pi_\theta(\hat{\tau} \mid x) = \prod_{n=1}^N \pi_\theta(\hat{\mathcal{A}}_n^p \mid \hat{h}_n) \pi_\theta(\hat{\mathcal{A}}_n^t \mid \hat{h}_n, \hat{\mathcal{A}}_n^p), \quad (3)$$

where $\hat{\tau} = \{(\hat{\mathcal{A}}_n^p, \hat{\mathcal{A}}_n^t)\}_{n=1}^N$ denotes the policy-generated spans, and

$$\hat{h}_n = \left(x, \left\langle (\hat{\mathcal{A}}_i^p, \hat{\mathcal{A}}_i^t, \hat{o}_i) \right\rangle_{i=1}^{n-1} \right), \quad (4)$$

with $\hat{o}_i$ the output or status observed after executing the i -th generated TOOL span.

Operationally, the structural tags defined in the data format, such as `<plan>` and `<tool>`, identify PLAN/TOOL token boundaries.

These boundaries enable mode-separated loss masks and reward-component assignment: PLAN spans are used for semantic fidelity evaluation, while TOOL spans are used for syntactic-validity evaluation.

Teacher-forced step sampling. The factorization above describes a free-running trajectory, which is the history used at autoregressive inference time. During training, we instead sample at aligned decision points. For an annotated step n , let

$$h_n = \left(x, \left\langle (\mathcal{A}_i^p, \mathcal{A}_i^t, o_i) \right\rangle_{i=1}^{n-1} \right) \quad (5)$$

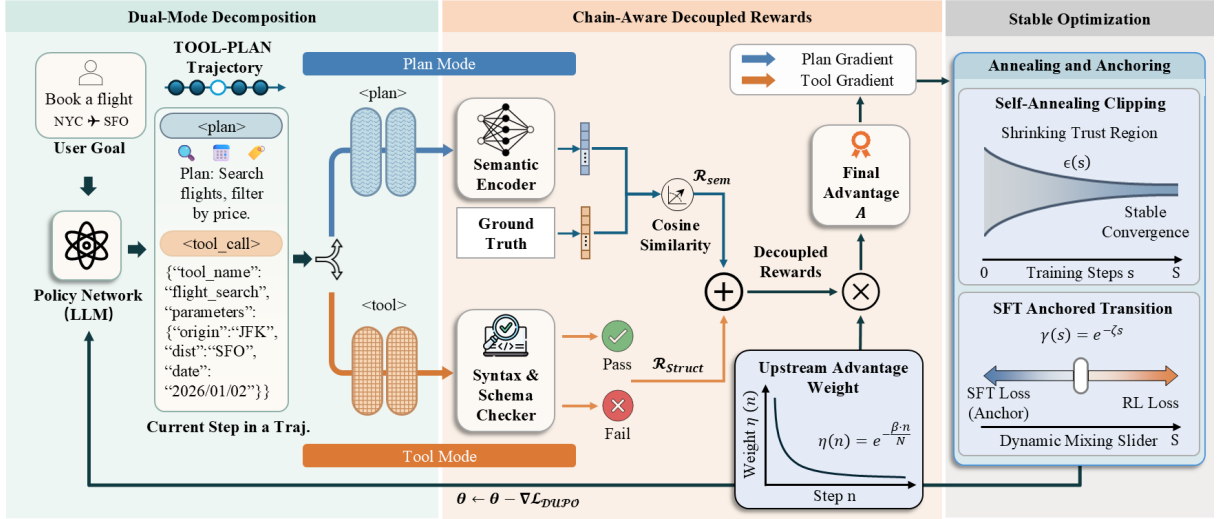


Figure 4: Illustration of the proposed Dual-Mode Unified Policy Optimization (DUPO) framework. The figure’s “Chain-Aware Decoupled Rewards” block denotes the Dual-Mode Reward together with the Chain-Aware Upstream Advantage (CAUA).

be the ground-truth prefix, including the preceding tool outputs. For every candidate g , we restore the same h_n , sample $\hat{\mathcal{A}}_{g,n}^p$, and then sample $\hat{\mathcal{A}}_{g,n}^t$ conditioned on that candidate PLAN. Consequently, candidates compared at step n share the same task state and are never aligned by matching actions from divergent histories. The semantic PLAN reward compares each candidate PLAN with the reference $\mathcal{A}_n^p$ at this same annotated decision point; it is not matched to a plan from a generated history. The comparison baselines use the same teacher-forced convention and the same serialized PLAN–TOOL inputs; DUPO’s distinction is how the typed spans are optimized.

4.2 Mode-Separated Policy Objective

DUPO performs a GRPO-style update (Shao et al., 2024) with separate PLAN and TOOL contributions. For each input x and annotated decision point n , we sample G candidate pairs $\{(\hat{\mathcal{A}}_{g,n}^p, \hat{\mathcal{A}}_{g,n}^t)\}_{g=1}^G$ from the rollout policy $\pi_{\theta'}$ conditioned on the shared teacher-forced prefix h_n , where θ' denotes pre-update policy parameters. Each candidate is scored at the step level: Section 4.4 defines the reward $\mathcal{R}_{g,n}$.

Following GRPO, we normalize rewards within the rollout group at each annotated decision point:

$$\tilde{\mathcal{A}}_{g,n} = \frac{\mathcal{R}_{g,n} - \text{mean}(\mathcal{R}_{1,n}, \dots, \mathcal{R}_{G,n})}{\text{std}(\mathcal{R}_{1,n}, \dots, \mathcal{R}_{G,n})}. \quad (6)$$

The mean and standard deviation are computed over the G candidates at the same annotated step

n . Thus, group normalization compares candidates under one restored task state rather than across positions from different free-running trajectories.

Chain-Aware Upstream Advantage. During autoregressive inference, an incorrect upstream sub-goal can make otherwise valid downstream actions irrelevant. The upstream weighting does not assume that early steps are intrinsically harder; it is a position prior encoding their greater potential to affect later decisions. DUPO applies this prior *after* group normalization:

$$\bar{\mathcal{A}}_{g,n} = \eta(n) \cdot \tilde{\mathcal{A}}_{g,n}, \quad (7)$$

with

$$\eta(n) = e^{-\frac{\beta n}{N}}, \quad \beta = \ln(N + 1). \quad (8)$$

This yields $\eta(1) = (N + 1)^{-1/N}$ and $\eta(N) = 1/(N + 1)$: early steps receive larger normalized-advantage magnitudes, while later steps are attenuated but retain a non-zero signal. Crucially, because $\eta(n)$ is applied *after* the per-step standardization in Eq. (6), it cannot be absorbed by the normalization and provides a strictly monotone emphasis on prefix quality. The shaped advantage $\bar{\mathcal{A}}_{g,n}$ is used in the policy loss below.

The flat, linear, and discrete schedules used in the ablation are defined in Appendix A.3; their values and breakpoints are fixed across datasets and backbones.

Mode-Separated Objective. DUPO then computes separate importance ratios for the generated

PLAN and TOOL spans. For the PLAN span, the ratio is

$$\rho_{g,n}^p \equiv \rho_{g,n}^p(\theta) = \frac{\pi_\theta(\hat{\mathcal{A}}_{g,n}^p | h_n)}{\pi_{\theta'}(\hat{\mathcal{A}}_{g,n}^p | h_n)}, \quad (9)$$

and for the TOOL span, the ratio is

$$\rho_{g,n}^t \equiv \rho_{g,n}^t(\theta) = \frac{\pi_\theta(\hat{\mathcal{A}}_{g,n}^t | h_n, \hat{\mathcal{A}}_{g,n}^p)}{\pi_{\theta'}(\hat{\mathcal{A}}_{g,n}^t | h_n, \hat{\mathcal{A}}_{g,n}^p)}. \quad (10)$$

Each span probability is computed only over tokens selected by the corresponding PLAN or TOOL mask. We use the clipped surrogate

$$\ell(\rho, A) = \min(\rho A, \text{clip}(\rho, 1 - \epsilon, 1 + \epsilon)A), \quad (11)$$

where ϵ is later instantiated as an adaptive clipping bound $\epsilon(s)$ in Section 4.3.

The resulting mode-separated RL loss is

$$\mathcal{L}_{\text{RL}}(\theta) = -\frac{1}{G} \sum_{g,n} \sum_{m \in \{p,t\}} \omega_m \ell(\rho_{g,n}^m, \bar{A}_{g,n}), \quad (12)$$

The coefficients ω_p and ω_t calibrate the two spans, while the mode separation itself comes from computing distinct policy ratios and losses for PLAN and TOOL tokens.

4.3 Continuation Schedule for Optimization

On-policy rollouts in RL can be unstable before the policy has internalized the PLAN–TOOL format, while pure supervised imitation cannot improve beyond reference trajectories.

DUPO therefore uses a continuation schedule that gradually shifts optimization from supervised typed imitation to RL-driven trajectory improvement, coupling two mechanisms: an adaptive clipping bound $\epsilon(s)$ that controls policy updates, and an SFT anchor $\gamma(s)$ for early format stability.

Self-Annealing Clipping. Let s be the current training step and S the total training budget. We define the remaining training horizon as $\Delta_s = S - s + 1$. Instead of using a fixed clipping bound, DUPO adapts the trust region over training:

$$\epsilon(s) = \frac{\bar{\epsilon}(s) \cdot \Delta_s}{\bar{\epsilon}(s) + \Delta_s}, \quad \epsilon_0 = 0.5, \quad (13)$$

where $\bar{\epsilon}(s)$ is the moving average of previous clipping bounds, initialized with $\bar{\epsilon}(0) = \epsilon_0$. Larger remaining horizon permits larger updates, while the bound tightens near convergence to reduce late-stage variance.

SFT-Anchored Transition. DUPO also anchors early RL updates to reference PLAN–TOOL trajectories through a supervised loss. The total loss combines the mode-separated RL loss from Section 4.2 with a supervised fine-tuning (SFT) loss:

$$\mathcal{L}_{\text{DUPO}}(\theta) = \gamma(s) \mathcal{L}_{\text{SFT}}(\theta) + (1 - \gamma(s)) \mathcal{L}_{\text{RL}}(\theta). \quad (14)$$

The mixing coefficient $\gamma(s)$ decays over training, where s denotes the current global optimizer step:

$$\gamma(s) = \exp(-\zeta s), \quad (15)$$

where $\zeta \in [0, 1)$ controls the transition rate from supervised anchoring to RL optimization.

The SFT term $\mathcal{L}_{\text{SFT}}$ is the per-step-averaged negative log-likelihood of the ground-truth step a_n under the current policy:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{N} \sum_{n=1}^N \log \pi_\theta(a_n | h_n). \quad (16)$$

h_n is the reference history before step n (defined as $\hat{h}_n$ without hats). As $\gamma(s)$ decays, the supervised term stabilizes high-variance early updates while the RL term receives increasing weight. In parallel, $\epsilon(s)$ tightens the update bound.

4.4 Dual-Mode Step Rewards

A multi-step tool reward must give informative feedback within each step: terminal rewards are too coarse, and an undifferentiated score obscures whether an error arose in planning or execution. DUPO therefore decomposes the step score into typed semantic and syntactic components through the Dual-Mode Reward (DMR):

$$\mathcal{R}_{g,n} = \mathcal{R}_{\text{syn}}(g, n) + \lambda_{\text{sem}} \mathcal{R}_{\text{sem}}(g, n) \quad (17)$$

Here, $\mathcal{R}_{g,n}$ is the aggregate reward for candidate g at annotated step n , used in Section 4.2. Its two terms are evaluated against different spans: $\mathcal{R}_{\text{syn}}$ checks the TOOL span and $\mathcal{R}_{\text{sem}}$ checks the PLAN span before they are combined into the scalar step score. Thus, the default retains typed reward components and span-specific policy ratios, while group normalization produces the scalar step advantage used by the objective. The upstream anchor is handled separately by the Chain-Aware Upstream Advantage (CAUA).

Table 2: Performance comparison across BFCL, API-Bank, and Tau2. Scores are percentages. NL denotes Non-Live and MT denotes Multi-Turn; for API-Bank, L1–L3 denote Level 1–3 success. Bold and underline indicate the best and second-best result, respectively, within each backbone and metric; ties are shown with the same emphasis.

Method	BFCL						API-Bank				Tau2		
	Ovr. ↑	NL ↑	Live ↑	MT ↑	Rel. ↑	Irrel. ↑	Ovr. ↑	L1 ↑	L2 ↑	L3 ↑	Retail ↑	Airline ↑	Telecom ↑
Backbone: Qwen2.5-3B													
Raw	33.04	42.52	53.96	1.00	64.71	56.01	35.68	43.36	22.39	19.08	36.8	30.7	26.3
SFT	47.82	66.67	67.75	2.38	77.78	100.00	57.12	61.15	49.25	40.52	52.2	40.0	35.3
StepTool	52.39	80.95	72.30	3.15	<u>85.66</u>	83.75	<u>64.68</u>	<u>71.13</u>	<u>62.68</u>	44.72	56.0	42.4	35.0
ToolRL	<u>52.98</u>	81.58	<u>73.78</u>	<u>3.75</u>	88.24	84.85	63.32	68.92	61.19	<u>47.33</u>	<u>59.1</u>	<u>45.6</u>	<u>38.9</u>
DUPO	53.10	81.29	75.57	10.12	83.33	<u>87.92</u>	66.33	71.18	65.67	48.85	60.3	46.9	39.2
Backbone: Qwen2.5-7B													
Raw	41.97	66.02	53.51	4.25	76.47	62.66	58.29	65.66	41.79	<u>44.27</u>	46.9	38.4	24.1
SFT	43.05	51.35	65.53	4.62	50.00	86.86	59.63	66.67	53.73	41.22	44.5	33.7	37.3
StepTool	56.63	85.31	73.40	16.22	<u>81.11</u>	75.83	62.73	71.00	<u>65.28</u>	33.17	65.0	<u>50.3</u>	<u>43.0</u>
ToolRL	<u>58.38</u>	86.17	74.90	<u>18.12</u>	83.33	<u>76.68</u>	<u>64.66</u>	<u>73.93</u>	61.19	38.16	<u>66.7</u>	45.3	38.6
DUPO	59.91	86.62	75.55	24.63	83.33	71.99	67.84	74.23	65.67	53.44	67.2	52.4	44.0
Backbone: Qwen2.5-14B													
Raw	56.32	83.38	74.01	14.75	83.33	70.09	45.56	51.88	31.34	33.59	62.0	48.3	36.6
SFT	56.89	81.65	<u>75.92</u>	11.50	<u>72.22</u>	86.70	64.82	72.43	59.85	42.75	59.6	46.7	38.3
StepTool	61.53	87.12	74.29	21.08	71.88	77.50	63.14	67.43	60.76	49.89	<u>70.0</u>	49.2	<u>47.1</u>
ToolRL	<u>62.40</u>	88.48	75.97	<u>23.87</u>	<u>72.22</u>	<u>79.16</u>	<u>66.16</u>	<u>73.93</u>	<u>62.69</u>	<u>50.38</u>	66.7	<u>51.8</u>	46.0
DUPO	65.79	89.29	75.68	34.00	66.67	78.00	68.01	74.19	67.33	53.91	72.2	54.9	48.7
Backbone: Qwen3-30B-A3B													
Raw	59.64	88.79	71.92	19.62	<u>94.44</u>	70.88	40.20	47.87	35.82	19.08	64.2	48.4	38.3
SFT	59.48	85.25	74.99	17.75	83.33	81.16	48.46	63.41	38.81	33.44	65.7	49.1	36.7
StepTool	60.56	88.49	70.62	22.76	100.00	70.83	58.38	<u>67.39</u>	49.33	38.04	64.0	<u>51.6</u>	<u>46.1</u>
ToolRL	<u>61.99</u>	<u>89.46</u>	72.19	<u>25.88</u>	100.00	70.99	<u>62.41</u>	66.52	<u>59.67</u>	<u>38.26</u>	<u>70.3</u>	47.0	45.9
DUPO	64.37	90.88	<u>74.08</u>	31.25	100.00	<u>72.89</u>	65.18	69.83	60.14	41.73	71.9	56.9	46.8
Backbone: Llama3.1-8B													
Raw	39.17	74.48	50.64	8.62	77.78	<u>37.70</u>	50.30	57.92	<u>48.21</u>	28.17	44.3	36.0	26.7
SFT	44.83	79.42	<u>52.78</u>	9.00	<u>88.89</u>	40.89	55.61	63.41	38.81	<u>40.46</u>	55.2	<u>45.7</u>	31.5
StepTool	43.92	85.11	49.85	9.25	87.68	19.17	58.01	67.81	46.56	37.69	49.6	42.4	26.8
ToolRL	<u>44.96</u>	86.04	50.96	11.00	<u>88.89</u>	20.19	<u>59.46</u>	<u>69.92</u>	62.69	25.95	56.4	44.0	35.6
DUPO	45.63	86.31	53.84	15.25	94.44	32.94	64.82	72.18	62.69	43.51	56.8	46.0	36.7

Dual-Mode Step Fidelity. The step-fidelity term contains two complementary signals:

(1) TOOL-side syntactic validity. $\mathcal{R}_{\text{syn}}(g, n)$ is a binary reward for whether the generated TOOL span satisfies the required tool-call schema, including valid function names, arguments, and JSON formatting.

(2) PLAN-side semantic alignment. $\mathcal{R}_{\text{sem}}(g, n)$ measures whether the generated PLAN span matches the reference planning intent:

$$\mathcal{R}_{\text{sem}}(g, n) = \text{sim}\left(\phi(\hat{\mathcal{A}}_{g,n}^p), \phi(\mathcal{A}_n^p)\right), \quad (18)$$

where $\phi(\cdot)$ is a pre-trained sentence encoder and $\text{sim}(\cdot, \cdot)$ denotes cosine similarity. This dense signal provides PLAN-side feedback beyond binary execution success/failure outcomes. It is intended primarily to tolerate linguistic paraphrases of the annotated intent; it does not establish equivalence

between structurally different decompositions, tool orders, or numbers of steps. A paraphrase sanity check is reported in Appendix A.3.

5 Experiments & Discussion

5.1 Experiment Setting

Benchmarks. Following Qian et al. (2025), we evaluate on three suites: BFCL (Patil et al., 2025), API-Bank (Li et al., 2023), and Tau2 (Barres et al., 2025), the last spanning the Retail, Airline, and Telecom domains. Further details are provided in Appendix A.1.

Implementation details. Backbone and hyperparameter details are in Appendix A.2.

Fair comparison. StepTool, ToolRL, and DUPO are trained on exactly the same PlanArena trajectories, serialized PLAN–TOOL format, preceding tool outputs, and benchmark protocols. Step-

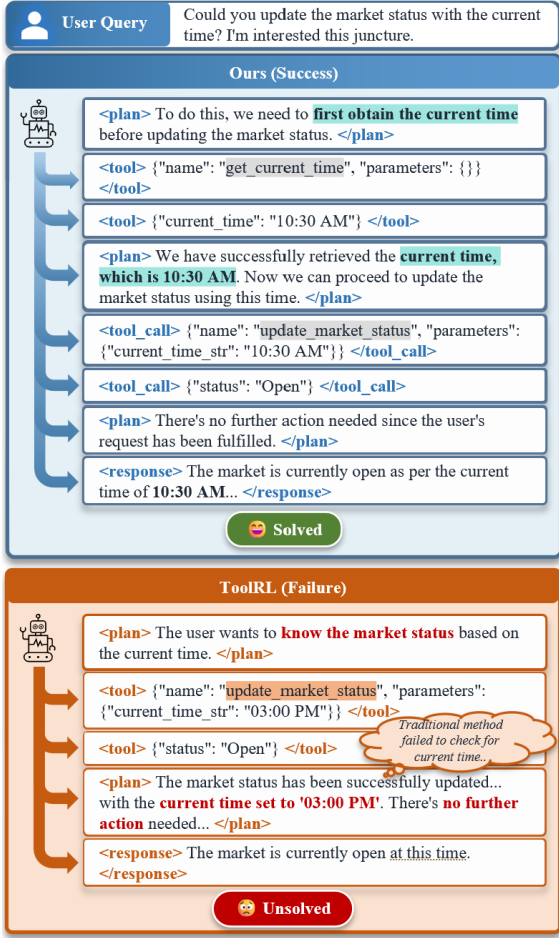


Figure 5: Qualitative comparison on a time-dependent scheduling request, Qwen2.5-14B. **Top:** DUPO articulates the dependency in `<plan>`, retrieves the current time, then issues a semantically grounded call. **Bottom:** the baseline emits a single `<tool_call>` with a hallucinated literal (“03:00 PM”), violating the logical chain.

Tool and ToolRL optimize the serialized action sequence with a single undifferentiated objective, whereas DUPO uses the same information with typed masks, mode-separated ratios, and separate reward components. The *w/o PLAN* ablation removes the PLAN channel; the objective and reward ablations retain the PLAN–TOOL data and isolate how it is optimized. Additional cross-dataset and reward-design checks are reported in Appendix A.3.

5.2 Overall Performance

Quantitative performance. Table 2 reports results on BFCL, API-Bank, and Tau2 across Qwen and Llama backbones. Although DUPO achieves the best Overall Accuracy, this metric is dominated by single-step execution; we therefore emphasize *Multi-Turn* accuracy, which better reflects coordi-

nated multi-step tool use. We compare against two recent RL-based tool-learning methods, StepTool (Yu et al., 2025b) and ToolRL (Qian et al., 2025). On BFCL (Patil et al., 2025), DUPO improves Multi-Turn accuracy over the strongest RL baseline, ToolRL, across backbones, with the largest absolute gain on Qwen2.5-14B (+10.13 percentage points); the gap is already +6.37 points at 3B. The improvement also holds on the MoE backbone Qwen3-30B-A3B (+5.37 points), suggesting that the benefit of typed PLAN–TOOL optimization is not tied to a single model family. Following prior work (Qian et al., 2025; Liu et al., 2025), we also report Relevance and Irrelevance, but treat them as ancillary to trajectory success and execution fidelity. On API-Bank, DUPO matches or outperforms both ToolRL and StepTool on every backbone–difficulty cell. At Level 3, the gain over StepTool is +20.27 points on Qwen2.5-7B (53.44 vs. 33.17; 61.1% relative) and +5.82 points on Llama3.1-8B (43.51 vs. 37.69; 15.4% relative), indicating that the advantage of mode-separated optimization grows with task complexity. We further evaluate on the Tau2 benchmark across three domains (Retail, Airline, Telecom). DUPO achieves the best score on all 15 model–domain pairs, with the largest gain in the Airline domain for Qwen3-30B-A3B (+5.3 points over the strongest baseline).

Qualitative performance. Figure 5 illustrates the behavior of DUPO on a multi-step request using Qwen2.5-14B. The baseline trajectory attempts to satisfy the final goal directly, hallucinating a parameter (“03:00 PM”) and skipping the necessary information-retrieval step `get_current_time`. This produces a syntactically plausible tool invocation but fails to ground the action in the required intermediate observation. In contrast, DUPO first uses the PLAN span to identify the missing prerequisite—the current time—and invokes `get_current_time` before updating the market status. The returned observation (“10:30 AM”) is then carried into the next PLAN–TOOL step and used as the argument for `update_market_status`. This example illustrates how PLAN-to-TOOL factorization helps ground an executable invocation in the intended subgoal and the relevant intermediate observation.

5.3 Ablation Study

We ablate DUPO along three axes: PLAN–TOOL supervision, dual-mode reward and upstream ad-

Table 3: Ablation study on BFCL. Results are reported as Overall / Multi-Turn accuracy (%).

Component	Variant	Qwen2.5-3B	Qwen2.5-7B
<i>PLAN–TOOL supervision</i>			
PLAN span	w/o PLAN	48.70 / 3.25	59.26 / 10.88
	w/ PLAN	53.10 / 10.12	59.91 / 24.63
<i>Reward & Advantage design</i>			
Semantic reward	$\lambda_{\text{sem}} = 0$	48.74 / 3.88	58.80 / 22.12
	$\lambda_{\text{sem}} = 0.2$	48.26 / 3.88	59.01 / 23.25
	$\lambda_{\text{sem}} = 0.5$	49.27 / 6.62	59.54 / 23.50
	$\lambda_{\text{sem}} = 0.8$	53.10 / 10.12	59.91 / 24.63
Upstream Adv.	Flat (no decay)	48.26 / 5.25	59.37 / 24.12
	Linear	48.77 / 4.25	59.82 / 23.88
	Discrete	48.27 / 3.62	60.17 / 23.75
	Exp.	53.10 / 10.12	59.91 / 24.63
<i>Optimization stabilization</i>			
Self-annealing	w/o	49.27 / 6.62	57.31 / 20.57
	w/	53.10 / 10.12	59.91 / 24.63
SFT anchor	RL only	50.21 / 5.25	59.54 / 21.62
	Anchored	53.10 / 10.12	59.91 / 24.63

vantage, and optimization stabilization (Table 3).

PLAN supervision. To test whether explicit planning supervision is necessary, we remove the PLAN span and train only on TOOL spans. PLAN supervision substantially improves Multi-Turn accuracy, especially for the 3B model (3.25% $\rightarrow$ 10.12%). The gain indicates that exposing the intended subgoal provides a useful intent channel for multi-step orchestration, rather than forcing the model to infer dependencies solely from TOOL tokens.

Semantic reward and weight. We study the PLAN-side semantic reward weight λ_{sem} . Both 3B and 7B models perform best at $\lambda_{\text{sem}} = 0.8$. Compared with using only the binary TOOL-side syntactic signal ($\lambda_{\text{sem}} = 0$), adding $\mathcal{R}_{\text{sem}}$ consistently improves Multi-Turn accuracy, showing that dense feedback on planning intent is useful for typed trajectory optimization.

Upstream advantage. We ablate the chain-aware upstream advantage by comparing its exponential anchor $\eta(n)$ in Eq. (8) with flat, linear, and discrete schedule alternatives while keeping the rest of DUPO fixed. The exponential anchor design yields the best Multi-Turn accuracy on both backbones. The gain over the closest alternative is +4.87pp on 3B and +0.51pp on 7B. Linear and discrete schedules underperform the flat baseline at 7B, suggesting that overly aggressive or non-smooth advantage shaping can suppress useful mid-trajectory feedback.

Self-annealing clipping. We compare DUPO’s adaptive clipping bound with a static clipping baseline–DAPO (Yu et al., 2025a). Self-annealing improves Multi-Turn accuracy from 6.62% to 10.12% on 3B and from 20.57% to 24.63% on 7B. This suggests that allowing larger early updates while tightening the bound later better balances exploration and stability in multi-step tool learning.

SFT anchoring. We ablate the SFT anchor with a pure-RL cold start, which drops Multi-Turn accuracy from 10.12% to 5.25% (3B) and from 24.63% to 21.62% (7B), showing that supervised anchoring stabilizes early PLAN–TOOL learning while the RL contribution grows (Table 3).

The additional API-Bank component ablations (Table 5), reward-implementation comparisons (Table 6), reward-granularity test (Table 7), and clipping-initialization sweep (Table 8) are reported in Appendix A.3. Together they test whether the gains persist across datasets and under alternative ways of assigning feedback.

5.4 Sensitivity Analysis

We further ablate the SFT–RL anchor decay rate ζ and the PLAN/TOOL calibration weights ω_p, ω_t (Appendix A.5). The default ζ performs best, and the mode-weight sweep peaks at the symmetric setting—indicating that DUPO’s gains stem from typed optimization and advantage shaping rather than manual asymmetric reweighting.

5.5 Diagnostic Analysis

Additional diagnostics are provided in Appendix A.4. We further stratify BFCL by interaction complexity, analyze training-time reward curves, and break down residual failure modes. These diagnostics show that DUPO’s gains concentrate on long-horizon regimes without degrading simple calls, while improving training stability and reducing the share of planning-related failures.

6 Conclusion

We presented DUPO, a typed trajectory optimization framework that separates PLAN-side intent from TOOL-side invocation within a unified policy, using mode-separated updates, a dual-mode reward, and an upstream advantage to improve multi-step tool use across standard benchmarks. We also introduced PlanArena, a human-annotated PLAN–TOOL dataset supporting this paradigm.

Limitations

Five caveats bound the scope of our claims. (i) *Dependence on structured plan annotations*. The semantic reward $\mathcal{R}_{\text{sem}}$ requires a reference plan trace per step, which is more expensive to elicit than outcome-only labels. PlanArena is a high-quality seed rather than a hard prerequisite, but the amount of pseudo-plan supervision needed to recover the reported gains remains open. (ii) *Single-reference supervision*. The embedding reward is designed mainly to tolerate linguistic paraphrases; it does not guarantee equivalence for structurally different but valid decompositions, tool orders, or planning granularities, which may receive lower scores. (iii) *Training–inference gap*. Training samples each decision point from the annotated prefix, whereas inference conditions autoregressively on previously generated actions. We do not yet train on off-reference histories, so robustness to compounding model-generated deviations remains to be studied. (iv) *Position prior*. CAUA emphasizes upstream steps because errors there can invalidate dependent decisions; later steps that are unusually critical may consequently be underweighted. (v) *Evaluation surface and capacity*. Although Tau2 extends coverage beyond BFCL/API-Bank, the suites remain primarily textual function-call benchmarks, and the gains attenuate for smaller backbones. Multimodal tools, stateful web environments, and adversarial schema changes are not yet stress-tested.

References

- Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. 2025. [\$\tau^2\$ -bench: Evaluating conversational agents in a dual-control environment](#). *arXiv preprint arXiv:2506.07982v1*.
- Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. [Plan-and-act: Improving planning of agents for long-horizon tasks](#). In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 15419–15462. PMLR.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The Llama 3 herd of models](#). *arXiv preprint arXiv:2407.21783v3*.
- Zhicheng Guo, Sijie Cheng, Hao Wang, Shihao Liang, Yujia Qin, Peng Li, Zhiyuan Liu, Maosong Sun, and Yang Liu. 2024. [StableToolBench: Towards stable large-scale benchmarking on tool learning of large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024 (Findings of ACL)*, pages 11143–11156. Association for Computational Linguistics.
- Yue Huang, Jiawen Shi, Yuan Li, Chenrui Fan, Siyuan Wu, Qihui Zhang, Yixin Liu, Pan Zhou, Yao Wan, Neil Gong, and Lichao Sun. 2024. [MetaTool benchmark for large language models: Deciding whether to use tools and which to use](#). In *International Conference on Learning Representations (ICLR)*, volume 2024, pages 42978–43007.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. [API-bank: A comprehensive benchmark for tool-augmented LLMs](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3102–3116. Association for Computational Linguistics.
- Zhiwei Li, Yong Hu, and Wenqing Wang. 2025. [Encouraging good processes without the need for good answers: Reinforcement learning for LLM agent planning](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track (EMNLP Industry Track)*, pages 1654–1666. Association for Computational Linguistics.
- Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Ning Wu, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, and 8 others. 2025. [ToolACE: Winning the points of LLM function calling](#). In *International Conference on Learning Representations (ICLR)*, volume 2025, pages 41359–41381.
- Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. [AgentBench: Evaluating large language models as agents](#). In *International Conference on Learning Representations (ICLR)*, volume 2024, pages 52989–53046.
- Xuanzhang Liu, Jianglun Feng, Zhuoran Zhuang, Junzhe Zhao, Maofei Que, Jieting Li, Dianlei Wang, Hao Tong, Ye Chen, and Pan Li. 2026. [CoDA: A context-decoupled hierarchical agent with reinforcement learning](#). In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 436–446. Association for Computing Machinery.
- Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. [The berkeley function](#)

- calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pages 48371–48392. PMLR.
- Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. **Gorilla: Large language model connected with massive APIs**. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 126544–126565. Curran Associates, Inc.
- Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiushi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. 2025. **ToolRL: Reward is all tool learning needs**. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pages 105523–105553. Curran Associates, Inc.
- Cheng Qian, Chi Han, Yi Ren Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. 2023. **CREATOR: Tool creation for disentangling abstract and concrete reasoning of large language models**. In *Findings of the Association for Computational Linguistics: EMNLP 2023 (Findings of EMNLP)*, pages 6922–6939. Association for Computational Linguistics.
- Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, and 24 others. 2025. **Tool learning with foundation models**. *ACM Computing Surveys (CSUR)*, 57(4):1–40.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. **Tool-LLM: Facilitating large language models to master 16000+ real-world APIs**. In *International Conference on Learning Representations (ICLR)*, volume 2024, pages 9695–9717.
- Mrinal Rawat, Ambuje Gupta, Rushil Goomer, Alessandro Di Bari, Neha Gupta, and Roberto Pieraccini. 2025. **Pre-act: Multi-step planning and reasoning improves acting in LLM agents**. *arXiv preprint arXiv:2505.09970v2*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. **Toolformer: Language models can teach themselves to use tools**. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pages 68539–68551. Curran Associates, Inc.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. **Proximal policy optimization algorithms**. *arXiv preprint arXiv:1707.06347v2*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. **DeepSeekMath: Pushing the limits of mathematical reasoning in open language models**. *arXiv preprint arXiv:2402.03300v3*.
- Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. 2023. **ToolAlpaca: Generalized tool learning for language models with 3000 simulated cases**. *arXiv preprint arXiv:2306.05301v2*.
- Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. 2024. **MINT: Evaluating LLMs in multi-turn interaction with tools and language feedback**. In *International Conference on Learning Representations (ICLR)*, volume 2024, pages 32593–32627.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chuji Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. **Qwen3 technical report**. *arXiv preprint arXiv:2505.09388v1*.
- 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, Jiayi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024. **Qwen2.5 technical report**. *arXiv preprint arXiv:2412.15115v2*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gao-hong Liu, Juncai Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, and 17 others. 2025a. **DAPO: An open-source LLM reinforcement learning system at scale**. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, pages 113222–113244. Curran Associates, Inc.
- Yuanqing Yu, Zhefan Wang, Weizhi Ma, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. 2025b. **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 (CIKM)*, pages 3952–3962. Association for Computing Machinery.
- Wenhao Zhang, Yuexiang Xie, Yuchang Sun, Yanxi Chen, Guoyin Wang, Yaliang Li, Bolin Ding, and Jingren Zhou. 2026. **On-policy RL meets off-policy experts: Harmonizing supervised fine-tuning and reinforcement learning via dynamic weighting**. In *International Conference on Learning Representations (ICLR)*.
- Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang,

Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. [Qwen3 Embedding: Advancing text embedding and reranking through foundation models](#). *arXiv preprint arXiv:2506.05176v3*.

A Additional Details

A.1 Benchmarks

We evaluate DUPO on three tool-use suites that cover complementary aspects of multi-step tool interaction: BFCL (Patil et al., 2025), API-Bank (Li et al., 2023), and Tau2 (Barres et al., 2025). BFCL measures function-calling accuracy across live, non-live, and multi-turn settings, and also reports relevance/irrelevance detection for tool-use necessity. API-Bank evaluates generalization across three difficulty levels, ranging from single-tool calls to tasks requiring integrated planning. Tau2 tests sustained tool use in dual-control conversations where both the agent and a simulated user can modify shared state; we report results on Retail, Airline, and Telecom, with Telecom representing the deepest dialogue-state setting.

A.2 Implementation Details

We fine-tune Qwen-2.5-Instruct (Qwen2.5) (Yang et al., 2024), Qwen3 (Yang et al., 2025), and Llama3.1 (Grattafiori et al., 2024) on PlanArena. DUPO uses $G = 16$ candidates per aligned decision point with sampling temperature 1.0. We incorporate the dual-mode step reward of §4.4 using Qwen3-Embedding-0.6B (Zhang et al., 2025) as the encoder $\phi(\cdot)$, an anchored schedule with $\zeta = 0.5$, and the dual-mode weights $\omega_p = \omega_t = 0.5$ with $\lambda_{\text{sem}} = 0.8$ (both stress-tested in §A.5). The upstream advantage anchor $\eta(n)$ (Eq. 8) requires no additional hyperparameters beyond the trajectory length N . We analyze these choices in Sections 5.3 and A.5. All BFCL results use **BFCL v3** with the official evaluation harness. Training is performed on $8 \times \text{NVIDIA A100 (80GB)}$ GPUs with global batch size 256. Wall-clock cost is closely matched to ToolRL: DUPO trains in 7.5 h (3B), 10.1 h (7B), and 14.2 h (14B), compared with 7.3 h / 10.3 h / 14.1 h for ToolRL, corresponding to an overhead of at most 3%.

Table 4 summarizes the reported settings that are needed to reproduce the experiments.

A.3 Additional Ablations and Robustness

We report additional cross-dataset and robustness experiments. Unless noted otherwise, the ablations use the same PlanArena training data, teacher-forced step sampling, and evaluation protocols as the main experiments. Benchmark scores are percentages; embedding similarities are reported on a 0–1 scale.

Table 4: Reproducibility summary of settings reported in the manuscript.

Item	Reported setting
Backbones	Qwen2.5-Instruct (3B, 7B, 14B), Qwen3-30B-A3B, and Llama3.1-8B.
Training corpus	PlanArena (20,646 trajectories and 7,941 scenarios); benchmark-related scenarios were excluded before annotation.
Decision-point sampling	Teacher-forced prefix through step $n - 1$, including preceding tool outputs; sample PLAN and then TOOL at the aligned decision point.
Rollout group	$G = 16$ candidates per decision point, sampling temperature 1.0.
Reward	Qwen3-Embedding-0.6B encoder; $\lambda_{\text{sem}} = 0.8$ and $\omega_p = \omega_t = 0.5$.
Upstream credit	$\eta(n) = \exp(-\beta n/N)$ with $\beta = \ln(N + 1)$.
Stabilization	Self-annealing initialization $\epsilon_0 = 0.5$ and SFT-anchor decay $\zeta = 0.5$.
Compute	8× NVIDIA A100 (80GB) GPUs; global batch size 256.
Evaluation	BFCL v3 official harness; official benchmark splits for API-Bank and Tau2.
Runtime	DUPO: 7.5 h / 10.1 h / 14.2 h for 3B / 7B / 14B, respectively; at most 3% overhead over ToolRL.

Alternative upstream schedules. The main method uses one-based step indices $n \in \{1, \dots, N\}$. For the ablation schedules, let $k = n - 1$ be the zero-based step index. The flat (fixed/no-decay) baseline is $\eta_{\text{flat}}(k) = 1$, and the linear baseline is

$$\eta_{\text{lin}}(k) = \frac{2}{k + 1}. \quad (19)$$

The discrete schedule is

$$\eta_{\text{disc}}(k) = \begin{cases} 2.0, & k = 0, \\ 1.5, & 0 < k \leq 1, \\ 1.0, & 1 < k \leq 2, \\ 0.5, & 2 < k \leq 3, \\ 0.4, & 3 < k \leq 4, \\ 0.3, & 4 < k \leq 5, \\ 0.2, & k > 5. \end{cases} \quad (20)$$

Cross-dataset ablation. Table 5 repeats the component ablations on API-Bank for Qwen2.5-3B and Qwen2.5-7B. The Overall/L3 trends remain consistent with the BFCL results, supporting the robustness of the factorization across datasets and model sizes.

Reward implementation. Table 6 isolates PLAN–TOOL feedback and objective design on Qwen2.5-3B. The unified variant collapses the two spans into one undifferentiated action treatment. Replacing the semantic PLAN reward with a conventional token-level PLAN objective also substantially reduces performance.

Reward granularity. We compare aligned decision-point training with a free-running trajectory-level alternative in Table 7. The alternative assigns one shared reward to the full generated trajectory, whereas DUPO assigns localized

rewards at annotated decision points. Because the alternative jointly changes the sampled history and reward granularity, this comparison evaluates the two operational training regimes rather than isolating granularity while holding histories fixed.

Clipping initialization. Table 8 varies the initial self-annealing clipping value on Qwen2.5-3B. The default $\epsilon_0 = 0.5$ is strongest among the tested values while preserving the same cross-dataset trend.

Paraphrase robustness. We sampled 100 annotated decision points and asked four LLMs to paraphrase each reference PLAN while preserving the request, context, intended decision, entities, arguments, and dependencies. We then encoded the original and paraphrased plans with Qwen3-Embedding-0.6B and measured cosine similarity. Table 9 shows that the scorer assigns high similarity to these positive paraphrases. This is a wording-invariance sanity check, not an evaluation of correctness or discrimination: it includes neither incorrect-plan controls nor structurally different but valid plans.

A.4 Diagnostic Analysis

Long-horizon gains without simple-task regression. We stratify BFCL by interaction complexity and compare Qwen2.5-7B DUPO with Qwen2.5-7B ToolRL (Table 10). DUPO yields the largest gains on the regimes that require multi-step coordination: +120% relative improvement on MT-Long (4+ steps) and +35.9% on MT-Base (2+ steps with state). In contrast, the change on Parallel and Simple tasks is negligible (+2% and ≈ 0), indicating that DUPO improves long-horizon tool use without sacrificing shallow-call performance.

Training-time stability. Figure 6 plots PLAN, TOOL, and total reward during training for ToolRL

Table 5: Additional component ablations on API-Bank. Each cell reports Overall / L3 accuracy (%).

Variant	Qwen2.5-3B	Qwen2.5-7B
w/o PLAN	59.30 / 37.40	65.83 / 47.33
$\lambda_{\text{sem}} = 0$	59.80 / 38.17	67.00 / 50.38
Upstream: flat (no decay)	59.13 / 38.93	65.33 / 46.56
Upstream: linear	60.64 / 40.46	66.67 / 48.85
Upstream: discrete	61.31 / 39.69	66.83 / 51.15
w/o self-annealing	61.14 / 42.75	65.49 / 48.09
RL-only cold start	61.64 / 41.22	67.17 / 51.91
DUPO	66.33 / 48.85	67.84 / 53.44

Table 6: Alternative PLAN–TOOL reward and objective implementations on Qwen2.5-3B.

Reward implementation	BFCL Overall	BFCL Multi-Turn	API-Bank Overall	API-Bank L3
Default design	53.10	10.12	66.33	48.85
Unified PLAN–TOOL objective	39.38	5.25	56.45	41.22
Token-level PLAN objective	41.10	7.75	61.98	39.69

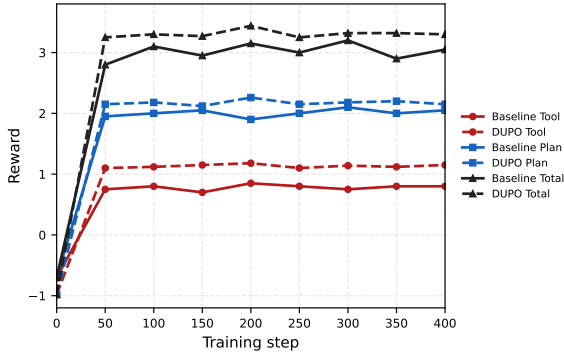


Figure 6: PLAN, TOOL, and total reward during training on Qwen2.5-3B. DUPO maintains higher rewards and lower step-to-step variance than ToolRL.

and DUPO on Qwen2.5-3B. DUPO maintains higher PLAN reward (≈ 2.18 vs. ≈ 2.0) and TOOL reward (≈ 1.1 vs. ≈ 0.8), with lower step-to-step variance. The total-reward gap appears within the first 50 steps and persists through convergence. This supports the role of DUPO’s continuation schedule in stabilizing early PLAN–TOOL learning.

Failure-mode diagnostic. We categorize failed multi-turn BFCL trajectories from Qwen2.5-3B into planning, parameter-binding, and tool-selection errors. Table 11 reports conditional shares within failed trajectories, so rows sum to approximately 100% (rounding). DUPO both improves overall Multi-Turn accuracy and reduces the share of planning failures among the remaining errors from 63.3% to 42.0% (-21.3pp). Because the categories are normalized within failures, the increases in binding and selection shares reflect re-

distribution after reducing the dominant planning category rather than a direct regression in those modes. This supports the intended effect of the typed PLAN–TOOL objective: improving trajectory success while reducing planning-related residual failures.

A.5 Sensitivity Analysis

Sensitivity to the anchor decay rate ζ . We compare the default SFT–RL anchor schedule with alternative decay rates on Qwen2.5-3B (Table 12). The default setting achieves the best Overall, Live, Multi-Turn, Relevance, and Irrelevance scores among the tested configurations, including the best Multi-Turn accuracy (10.12%) and Live accuracy (75.57%). The intermediate decay rate strikes the best balance: the slower transition ($\zeta = 0.3$) retains SFT anchoring too long, whereas the faster transition ($\zeta = 0.8$) exposes the policy to RL too aggressively. This indicates that the anchor decay rate is important for balancing supervised PLAN–TOOL anchoring with RL-driven improvement.

Impact of PLAN/TOOL calibration weights ω_p and ω_t . We sweep ω_p and ω_t over $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ on Qwen2.5-3B (Figure 7). The symmetric setting $\omega_p = \omega_t = 0.5$ attains the best Multi-Turn score (10.12%) and the highest Overall score in the grid. Strong asymmetry hurts both metrics: corner settings such as 0.1/0.9 or 0.9/0.1 reduce Multi-Turn accuracy to 3.62%–4.88% and Overall below 35%. The result suggests that DUPO’s gains come primarily from span-level routing and structured rewards rather than manual asymmetric scalar weighting; the weights mainly

Table 7: Aligned step-level DUPO versus a free-running trajectory-level alternative on Qwen2.5-3B.

Reward granularity	BFCL Overall	BFCL Multi-Turn	API-Bank Overall	API-Bank L3
Trajectory-level	33.28	2.00	36.18	14.50
Step-level (DUPO)	53.10	10.12	66.33	48.85

Table 8: Robustness to the initial clipping value ϵ_0 on Qwen2.5-3B.

Metric	$\epsilon_0 = 0.1$	$\epsilon_0 = 0.3$	$\epsilon_0 = 0.5$	$\epsilon_0 = 0.8$
BFCL Overall / Multi-Turn	31.54 / 4.88	34.77 / 7.75	53.10 / 10.12	19.73 / 3.50
API-Bank Overall	31.99	51.09	66.33	38.53

serve as calibration terms.

A.6 Annotation Guidelines

Annotators and compensation. All annotators were college-educated with experience in computer science or a related technical field, and were compensated at \$15 USD per hour (or the local equivalent).

Task presentation. Each annotation instance is a single trajectory presented with three components: (i) a natural-language user goal together with any relevant constraints, such as expected output format, restricted tool usage, or ordering requirements; (ii) the complete tool schema for every available tool, including the function name, natural-language description, parameter table with types and optionality, and return type; and (iii) the generated trajectory, structured as a sequence of steps. Each step contains two typed fields: `<plan>`, a natural-language reasoning trace describing the agent’s intent for the step, and `<tool>`, the tool call issued by the agent with its function name and argument values.

Annotation procedure. For each step, annotators wrote a step-level strategic rationale justifying the tool selection, grounding the decision in the user goal and prior context. They then verified each `<tool>` call by executing it against the live API, confirming that the function name, arguments, and behavior aligned with the tool schema and user intent. When a field was judged incorrect, the annotator supplied a correction: a revised `<plan>` when the reasoning was incorrect or incomplete; the correct invocation when the `<tool>` call used the wrong function, invalid arguments, missing required parameters, or was misaligned with the plan; and a removal marker when a step was entirely unnecessary.

Quality control. A separate annotator independently cross-checked plan coherence across steps, ensuring that the sequence of `<plan>` fields formed a logically consistent strategy toward the user goal. Before full-scale annotation, a pilot agreement test on 100 trajectories yielded Cohen’s $\kappa = 0.84$; this is a calibration measure rather than a full-corpus estimate. Every trajectory then passed a final acceptance check. Samples failing PLAN–TOOL coherence, tool-execution correctness, or cross-step consistency were manually corrected and reviewed by a different annotator before acceptance. We additionally audited a random 10% of accepted trajectories at the final stage.

Licensing and terms of use. We use all artifacts in accordance with their licenses. Backbones: Qwen2.5-7B/14B (Yang et al., 2024), Qwen3 (incl. Qwen3-30B-A3B) (Yang et al., 2025), and Qwen3-Embedding-0.6B (Zhang et al., 2025) are released under Apache-2.0; Qwen2.5-3B is used under the Qwen Research License for non-commercial research; Llama-3.1 (Grattafiori et al., 2024) is used under the Llama 3.1 Community License with its attribution requirements. Benchmarks BFCL (Patil et al., 2025) (Apache-2.0), API-Bank (Li et al., 2023) (MIT), and Tau2 (Barres et al., 2025) (MIT) are used for research evaluation under their respective licenses. PlanArena scenarios are derived from publicly listed RapidAPI tool schemas; consistent with prior RapidAPI-based resources, we use them for research only and release annotation-layer artifacts (PLAN–TOOL trajectories and metadata) rather than proprietary API responses. PlanArena and our code will be released under CC BY-NC 4.0 for research use.

Table 9: PLAN embedding similarity under paraphrasing (100 reference plans, four paraphraser, 400 pairs).

Paraphraser	Mean	Median	≥ 0.90
DeepSeek-V4-Flash	0.9518	0.9622	93%
Qwen3.5-397B-A17B	0.9385	0.9554	84%
GLM-5.1	0.9348	0.9484	82%
Kimi-K2.5	0.9323	0.9521	81%
All 400 pairs	0.9394	0.9543	85.0%

Table 10: DUPO vs. ToolRL on Qwen2.5-7B across BFCL complexity regimes. Gains concentrate on multi-step tasks without measurable regression on simple or parallel calls.

Task regime	ToolRL	DUPO	Δ (rel.)
MT-Long (4+ steps)	7.50	16.50	+120%
MT-Base (2+ steps)	18.12	24.63	+35.9%
Parallel (concurrent tools)	83.25	84.75	+2%
Simple (1-step)	75.69	75.72	≈ 0

Table 11: Failure-mode breakdown on multi-turn BFCL (Qwen2.5-3B). Values are conditional shares within failed trajectories and sum to approximately 100% after rounding.

Error type	ToolRL	DUPO	Change (pp)
Planning	63.3	42.0	-21.3
Parameter binding	23.3	33.3	+10.0
Tool selection	13.3	24.7	+11.4

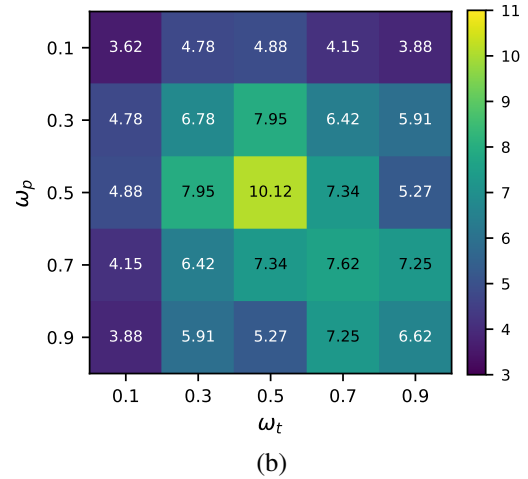
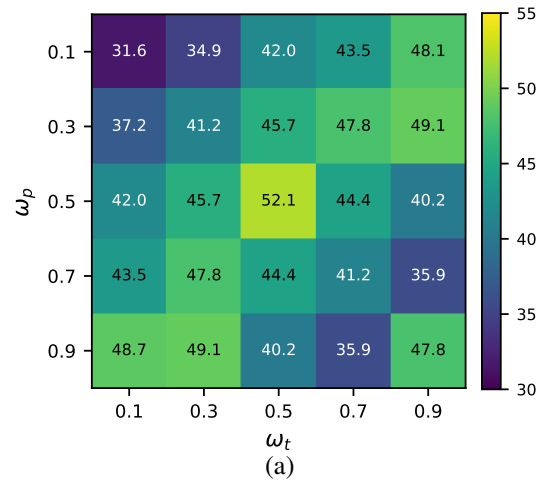


Figure 7: Sensitivity to PLAN/TOL calibration weights ω_p and ω_t . The symmetric center (0.5, 0.5) performs best on both Overall and Multi-Turn accuracy, while strong asymmetry degrades both metrics.

Table 12: Sensitivity to the anchor decay rate ζ on Qwen2.5-3B (BFCL, all metrics in %). The default configuration performs best among the tested settings.

Schedule	Overall $\uparrow$	Non-Live AST $\uparrow$	Live $\uparrow$	Multi-Turn $\uparrow$	Relevance $\uparrow$	Irrelevance $\uparrow$
$\zeta = 0.3$	50.18	78.65	64.50	7.62	77.78	70.38
$\zeta = 0.8$	50.96	74.38	66.59	9.62	77.78	76.14
$\zeta = 0.5$ (ours)	53.10	81.29	75.57	10.12	83.33	87.92