BREAKING THE ORCHESTRATION MONOCULTURE: TRAINING GENERALIZABLE LLM AGENTS VIA MULTI-ORCHESTRATION REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning (RL) has become the dominant paradigm for training LLM-based agents on tool-use tasks, yet virtually all existing work trains under a single, fixed orchestration such as REACT throughout the entire RL process. This orchestration monoculture limits both generalization and data utilization efficiency: agents overfit to one reasoning pattern and become brittle on out-of-distribution queries that demand different problem-decomposition strategies. We argue that the root cause runs deeper than a static choice: the training value of an orchestration drifts as the policy's capability grows, so early leaders may cap the long-run ceiling or even collapse while complex scaffolds only pay off once the policy can exploit them. We therefore treat orchestration as a **self-evolving curriculum**: a fixed pool of structurally diverse orchestrations whose training-time allocation is routed online by the policy's own reward stream. Instantiated as **Multi-Orchestration Reinforcement Learning (MORL)** with Discounted-UCB routing, the curriculum reweights rather than rewrites, adapting to capability drift while keeping the rollout distribution stable. On seven Search QA benchmarks, MORL outperforms a rollout-matched single-orchestration baseline by 5.3 exact-match points on average (43.3% vs. 38.0% with Qwen2.5-3B-Ins), with notable gains on OOD multi-hop sets (+7.8 on Bamboogle, +5.7 on MuSiQue), demonstrating strong task generalizability and robust orchestration adaptability across model sizes and domains.

1 INTRODUCTION

Training LLM-based agents to use external tools via reinforcement learning (RL) has gained significant momentum since the success of outcome-supervised RL for mathematical reasoning (Guo et al., 2025). A growing body of work demonstrates that RL can train agents to invoke search engines (Jin et al., 2025; Feng et al., 2025; Sun et al., 2025), execute SQL queries (Li et al., 2023), navigate web interfaces (Yao et al., 2022; Li et al., 2026), and complete complex multi-step tasks through extended interactive trajectories.

Most of these pipelines commit to a single orchestration, such as REACT (Yao et al., 2023), Reflexion (Shinn et al., 2023), or Plan-and-Solve (Wang et al., 2023a), and keep it fixed throughout training, an *orchestration monoculture*. The choice of reasoning scaffold is thus a hidden hyperparameter of agentic RL, one that is almost never exposed to optimization. What is less noticed is that even when the community does optimize this choice, it does so statically. Practitioners pick an orchestration by hand; automated approaches search over workflow structures offline against a frozen policy (Zhang et al., 2024; Hu et al., 2024; Shang et al., 2024); and recent self-evolving agent systems rewrite their own scaffolding in an offline outer loop driven by an external evaluator (Zhang et al., 2025b; Hebbar et al., 2026). All of these paradigms evolve the orchestration only at test time, on a static policy that is about to change.

During RL, the policy's capability improves by large factors, and the training value of an orchestration is contingent on that capability rather than being an intrinsic property of the scaffold. An orchestration is not a neutral wrapper: each one encodes a distinct inductive bias about how to decompose a problem and when to invoke tools, so different orchestrations yield their largest gains on different task types. A decomposition-heavy scaffold helps compositional multi-hop questions but adds little to single-hop

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

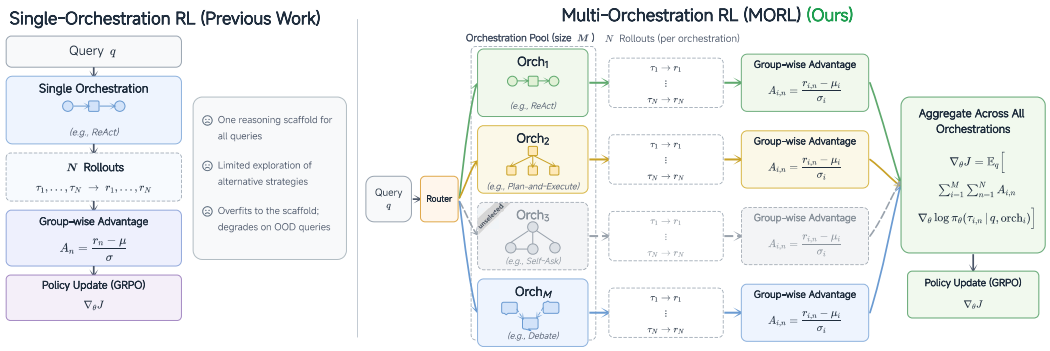


Figure 1: Existing pipelines treat the orchestration as a static, one-shot choice fixed before training (left), while MORL maintains a pool of diverse orchestrations and lets the training-time curriculum over them evolve with the policy's capability (right).

recall, while a verification-heavy one pays off mainly when evidence conflicts. The same asymmetry plays out over training time. Early in RL, when the policy can barely operate a complex scaffold, a simple linear loop yields the most usable signal; as capability grows, the complex scaffolds that initially produced little reward become the most instructive training environments. Static selection therefore falls into an *early-winner trap*: it commits to the scaffold that looks best at the starting line, which is precisely the one most likely to cap the policy's long-run ceiling. Orchestration choice is thus not a one-shot design decision but a dynamic decision process that must track the policy's evolving capability.

If the training-time value of an orchestration drifts with capability, the curriculum over orchestrations should evolve with it. We therefore introduce the **self-evolving orchestration curriculum**: a fixed pool of structurally diverse orchestrations whose training-time allocation is routed online by the policy's own reward stream, with no external evaluator in the loop. Together with the policy's weight updates, this forms a two-timescale loop: capability reshapes the curriculum, and the reshaped curriculum reshapes future capability.

We instantiate the curriculum as **Multi-Orchestration Reinforcement Learning (MORL)** with Discounted-UCB (D-UCB) routing (Garivier & Moulines, 2011). An LLM prepares a pool of M diverse orchestrations offline as executable code; for each training query, every orchestration defines an independent GRPO rollout group with per-group advantage normalization, so all scaffolds contribute complementary gradient signal; and D-UCB maintains discounted reward statistics per task type and reallocates the rollout budget toward orchestrations that are currently most valuable, while its exploration bonus keeps degraded scaffolds recoverable.

We evaluate on three task domains: open-domain QA with search (seven benchmarks), embodied planning (ALFWorld), and web navigation (WebShop). MORL improves over the compute-matched single-orchestration baseline by 5.3 exact-match points on average on Search QA (43.3% vs. 38.0% with Qwen2.5-3B-Ins), with the strongest gains on out-of-distribution multi-hop sets, and held-out-orchestration tests show substantially smaller degradation under unseen test-time formats. D-UCB routing matches this performance at a fraction of the rollout budget and leads uniform MORL under a matched rollout budget (Table 1, §5.4), with the largest margins exactly in the high-drift regimes our analysis predicts.

Our contributions are:

- **Phenomenon.** We characterize the non-stationarity of orchestration value during RL: reward rankings invert with capability, early winners can cap the ceiling or collapse, and scaffold value is capability-contingent. Static orchestration choice, by hand or by offline search, is blind to all three.
- **Framework.** We propose the self-evolving orchestration curriculum, instantiated as MORL with D-UCB routing: a fixed pool whose training-time allocation evolves online from the policy's own reward stream, reconfiguring rather than rewriting.

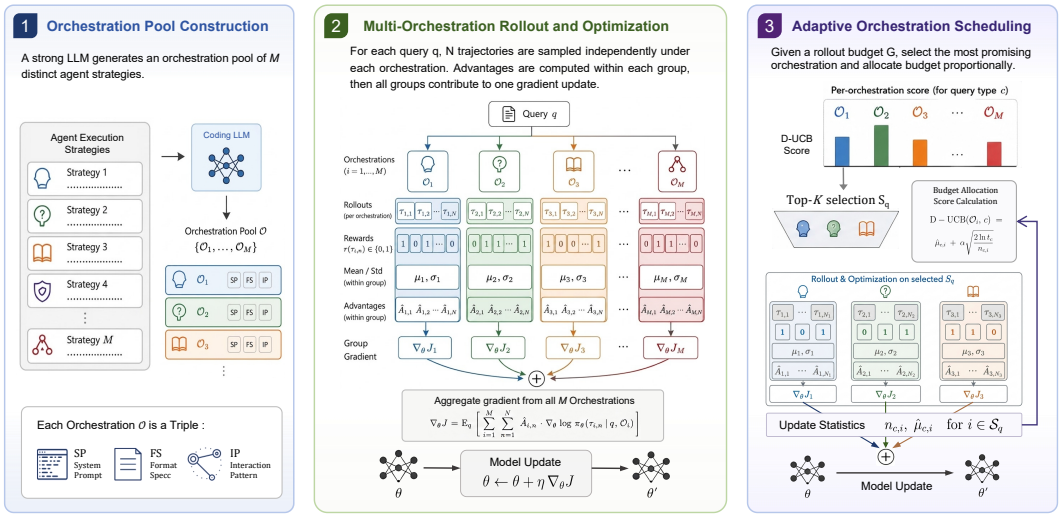


Figure 2: Overview of MORL: (1) a closed-source LLM generates the orchestration pool offline; (2) each training query launches independent rollout groups per orchestration with per-group GRPO normalization; (3) D-MORL evolves the curriculum by adaptively allocating the rollout budget across orchestrations as the policy's capability drifts.

- **Evidence.** Across three task domains and two model scales, MORL delivers two-level generalization (across tasks and across unseen orchestrations), and joint training yields net-positive cross-orchestration transfer that pure test-time scaffold selection cannot explain.

2 Related Work

Agentic RL for LLMs. Applying reinforcement learning with verifiable rewards (RLVR) (Guo et al., 2025) to tool-augmented LLM agents has been explored across interactive settings: search agents that interleave retrieval with reasoning (Jin et al., 2025; Chen et al., 2025; Sun et al., 2025; Wang et al., 2025b), navigation and embodied control with stabilized long-horizon training (Feng et al., 2025; Yao et al., 2022; Shridhar et al., 2021; Li et al., 2026), and general-purpose tool use (Wang et al., 2025a). These works share one characteristic: a single, fixed orchestration format throughout training. MORL is orthogonal and complementary: it enriches the rollout distribution with multiple orchestrations and lets their allocation evolve with the policy.

Agent orchestration and its optimization. Orchestrations structure how agents decompose long-horizon tasks, from interleaving reasoning with action (Yao et al., 2023), iterative self-correction (Shinn et al., 2023), and pre-execution planning (Wang et al., 2023a) to role-based multi-agent pipelines (Wu et al., 2023; Hong et al., 2024; Qian et al., 2024) and RL-trained multi-agent systems (Feng et al., 2026; Hong et al., 2025). A complementary line optimizes the orchestration structure itself: workflow search over code or modular components (Zhang et al., 2024; Hu et al., 2024; Shang et al., 2024; Dang et al., 2025), and self-evolving systems that rewrite their scaffolding in an offline outer loop or update harness and weights jointly (Zhang et al., 2025b; Hebbar et al., 2026; Gao et al., 2025). These paradigms evolve offline or at test time against an effectively frozen policy, with an exogenous signal from an external evaluator or meta-agent. The closest training-time counterpart, PATS (Shi et al., 2026), adapts textual skill entries in the prompt during RL and compresses them away as they are internalized. Our self-evolving curriculum differs on three axes: the evolving object is the allocation over structurally diverse orchestrations rather than prompt content or scaffold code; the goal is maintained diversity for generalization rather than internalization and compression; and the signal is endogenous, namely the policy's own reward stream.

Data augmentation and curriculum learning. Broader training distributions yield more robust generalization, from bootstrapped instructions and curated agent trajectories in SFT (Wang et al.,

2023b; Chen et al., 2023; Zeng et al., 2023; Longpre et al., 2023) to environment randomization in robot RL (Tobin et al., 2017) and mixed off-policy/on-policy rollouts in agent RL (Yan et al., 2025). MORL diversifies at the orchestration level instead: the same query is presented under multiple reasoning templates, injecting diversity into the online rollout distribution at no additional data cost. The curriculum view further connects to curriculum learning (Bengio et al., 2009) and self-paced learning (Kumar et al., 2010), where the learner's own signal schedules the training data; we schedule reasoning structures rather than instances, driven by verifiable rewards rather than instance loss.

3 METHOD: MULTI-ORCHESTRATION RL

3.1 PROBLEM FORMULATION

We consider an LLM agent π_θ interacting with an environment $\mathcal{E}$ over multiple turns. A trajectory is $\tau = (s_0, a_0, o_0, s_1, a_1, o_1, \ldots, s_T)$, where s_t is the state, a_t is the agent's action at the current turn, and o_t is the environment's observation. The agent receives a terminal reward $R(\tau) \in \mathbb{R}$ upon completing the task.

An *orchestration* $\mathcal{O}$ is a triple (*SystemPrompt*, *FormatSpec*, *InteractionPattern*) that encodes the high-level decision-making strategy governing how the agent reasons and acts. Different orchestrations encode different inductive biases about how to solve a task. For instance, REACT (Yao et al., 2023) assumes a linear `Thought`→`Action`→`Observation` loop, while SELFASK (Press et al., 2023) decomposes the query into explicit sub-questions before issuing search calls.

3.2 ORCHESTRATION POOL CONSTRUCTION

We define an *orchestration pool* $\mathcal{P} = \{\mathcal{O}_1, \ldots, \mathcal{O}_M\}$ of M heterogeneous strategies. The orchestration pool is constructed offline by a closed-source LLM. It receives a context $\mathcal{C} = (\mathcal{E}_{\text{task}}, \mathcal{I}_{\text{env}})$, where $\mathcal{E}_{\text{task}}$ is a description of the task domain and $\mathcal{I}_{\text{env}}$ is the agent execution interface (available tool endpoints, observation schema, and the code template the agent must conform to). Given this context, the model performs chain-of-thought reasoning to analyze the task's structural properties and then generates M diverse strategies in a single pass, directly expressing each as executable agent code that invokes the specified tool endpoints:

$$\{\mathcal{O}_1, \ldots, \mathcal{O}_M\} = \text{LLM}(\mathcal{E}_{\text{task}}, \mathcal{I}_{\text{env}}). \tag{1}$$

Each $\mathcal{O}_i$ is therefore a runnable code module that can be directly embedded into the rollout loop. This generation is performed once offline. Representative orchestration examples and the system-prompt templates for our experimental setup are provided in Appendix B.

3.3 MULTI-ORCHESTRATION ROLLOUT

For each training query q, MORL generates N rollout trajectories independently for each orchestration $\mathcal{O}_i \in \mathcal{P}$:

$$\tau_{i,1}, \ldots, \tau_{i,N} \sim \pi_\theta(\cdot \mid q, \mathcal{O}_i), \quad i = 1, \ldots, M. \tag{2}$$

Each orchestration thus defines its own rollout group of size N.

The reward for each trajectory combines an outcome reward with an optional format reward:

$$R(\tau_{i,n}) = R_{\text{correct}}(\tau_{i,n}) + \lambda \cdot R_{\text{format}}(\tau_{i,n}), \tag{3}$$

where R_{correct} is the correctness signal (depends on the task type), $R_{\text{format}} \in \{0, 1\}$ checks adherence to the orchestration's format specification, and $\lambda \geq 0$ controls the format weight. Because every orchestration attempts the same query under a shared correctness metric, each trajectory provides an independent training signal on the same data point, effectively multiplying the useful gradient updates extracted per query.

3.4 GRPO WITH MULTI-ORCHESTRATION

MORL computes group-relative advantages within each orchestration group, following GRPO (Shao et al., 2024):

$$\hat{A}_{i,n} = \frac{R(\tau_{i,n}) - \mu_{q,i}}{\sigma_{q,i}}, \tag{4}$$

where $\mu_{q,i}$ and $\sigma_{q,i}$ are the mean and standard deviation of rewards within group $(q, \mathcal{O}_i)$.

The policy gradient objective aggregates over all M groups:

$$\nabla_\theta J = \mathbb{E}_q \left[\sum_{i=1}^{M} \sum_{n=1}^{N} \hat{A}_{i,n} \cdot \nabla_\theta \log \pi_\theta(\tau_{i,n} \mid q, \mathcal{O}_i) \right] - \beta \nabla_\theta D_{\mathrm{KL}}\big(\pi_\theta \,\|\, \pi_{\mathrm{ref}}\big). \tag{5}$$

Eq. (5) is the standard GRPO objective (Shao et al., 2024) in our fully on-policy setting: the complete objective weights each advantage by the PPO-clipped importance ratio $\pi_\theta / \pi_{\theta_{\mathrm{old}}}$, but every batch is sampled from the current policy and used for a single update, so the ratio is identically one and the clip is inactive; the KL term ties the policy to the initial checkpoint π_{ref} (β in Appendix D). Different orchestrations expose distinct facets of the same query, so the per-group advantage signals are complementary rather than redundant: joint optimization over all M groups broadens the solution trajectories the model is rewarded for rather than overfitting to a single scaffold.

3.5 Adaptive Orchestration Scheduling

The orchestration curriculum. We call the allocation of the rollout budget across the pool the *orchestration curriculum*. Formally, at training step t and for a query of task type c, the curriculum is a distribution $\boldsymbol{\omega}_{c,t} = (\omega_1, \ldots, \omega_M)$ over $\mathcal{P}$, where ω_i is the share of the budget assigned to orchestration $\mathcal{O}_i$. Standard MORL uses a constant curriculum, $\omega_i = 1/M$ for all t. Our central claim is that the curriculum should *self-evolve*: because the policy π_θ changes throughout training, the value of each orchestration as a training environment drifts with it (§5.2), so a fixed allocation is suboptimal by construction. We deliberately evolve the curriculum by *reconfiguring rather than rewriting*: the pool stays fixed, and only the allocation $\boldsymbol{\omega}_{c,t}$ moves. Mutating the orchestrations themselves mid-training would make the rollout distribution discontinuously non-stationary and destabilize the RL loop; Appendix E.3 illustrates this failure mode.

Standard MORL allocates the rollout budget uniformly across all M orchestrations. When the total budget is limited, it is natural to ask: *which orchestrations are most valuable for a given query, and can we focus the budget on them?* We address this with **D-MORL** (MORL with D-UCB routing), which frames orchestration selection as a *non-stationary bandit*: each orchestration is an arm whose expected reward drifts as the policy π_θ evolves during training. We adopt the Discounted UCB (D-UCB) criterion of Garivier & Moulines (2011) to track this drift. The routing signal is fully endogenous: the discounted statistics are computed from the policy's own rollout rewards, with no external evaluator, meta-agent, or human-designed schedule in the loop.

Setup. Let $\mathcal{C}$ be a set of task types, where each query q belongs to exactly one type $c \in \mathcal{C}$ determined by the dataset it originates from. For each orchestration $\mathcal{O}_i$ and task type c, we maintain running statistics with discount factor $\gamma \in (0, 1)$:

$$\begin{aligned} \tilde{n}_{c,i}(t) &= \textstyle\sum_{s \le t} \gamma^{t-s}\, \mathbf{1}[\mathcal{O}_i \text{ selected for type-}c\text{, step } s], \\ \tilde{\mu}_{c,i}(t) &= \frac{\sum_{s \le t} \gamma^{t-s}\, r_{c,i}(s)\, \mathbf{1}[\cdots]}{\tilde{n}_{c,i}(t)}. \end{aligned} \tag{6}$$

Exponential discounting down-weights stale observations, allowing statistics to track the current reward level as the policy drifts.

D-UCB score. Given a query q of type c at training step t_c, the D-UCB score for orchestration $\mathcal{O}_i$ is:

$$\text{D-UCB}(\mathcal{O}_i, c) = \tilde{\mu}_{c,i} + \alpha \sqrt{\frac{2 \ln t_c}{\tilde{n}_{c,i}}}, \tag{7}$$

where $\alpha > 0$ is an exploration coefficient.

Per-query scheduling. For each query q of type c:

1. Compute D-UCB$(\mathcal{O}_i, c)$ for all $i \in [M]$.
2. Select the top-K orchestrations: $\mathcal{S}_q = \text{Top-}K\big(\{\text{D-UCB}(\mathcal{O}_i, c)\}_{i=1}^{M}\big)$.

Table 1: Exact match (%) on Search QA benchmarks under a matched rollout budget. *: in-distribution (ID); †: out-of-distribution (OOD). Best result per column in **bold**; second-best underlined. All methods are trained and evaluated under unified settings and dataset splits; for ExpandSearch, we use a same-scale model as the Squeezer component to avoid knowledge leakage. D-MORL reinvests the rollouts that uniform allocation spends on degraded arms into additional training queries. The data-budget-matched counterpart is reported in Appendix E.1.

Method	NQ*	TriviaQA†	PopQA†	HotpotQA*	2Wiki†	MuSiQue†	Bamboogle†	Avg
Qwen2.5-3B-Ins								
General baselines								
ZeroSearch	41.4	57.4	44.8	27.4	30.0	9.8	11.1	31.7
Search-R1	34.1	54.5	37.8	32.4	31.9	10.3	26.4	32.5
GiGPO	41.6	59.3	43.6	36.9	38.0	13.3	30.7	37.6
Router-R1	40.2	63.6	40.4	38.1	36.3	14.4	31.2	37.7
ExpandSearch	43.5	60.8	43.6	38.2	37.8	15.4	31.2	38.6
Single-Orchestration RL								
REACT	14.9	30.3	14.5	26.9	22.2	7.8	14.9	18.8
+ RL	40.0	50.3	40.2	42.9	38.1	18.8	35.6	38.0
Reflexion	0.0	3.2	0.0	3.2	1.6	0.8	5.7	2.1
+ RL	44.1	51.9	39.8	37.7	36.0	16.7	32.1	36.9
Plan-and-Exec	0.0	0.0	0.0	0.8	0.0	0.8	1.6	0.5
+ RL	42.3	55.1	46.0	37.7	36.5	15.0	35.1	38.2
MORL (Ours)	48.3	62.6	**51.4**	44.1	39.3	19.0	38.5	43.3
D-MORL (Ours)	**49.3**	**63.8**	51.2	**45.6**	**40.4**	**23.5**	**39.0**	**44.7**
Qwen2.5-7B-Ins								
General baselines								
ZeroSearch	43.6	65.2	48.8	34.6	35.2	18.4	27.8	39.1
Search-R1	39.3	61.0	39.7	37.0	41.4	14.6	36.8	38.5
GiGPO	46.1	65.2	46.3	42.2	44.6	19.0	36.3	42.8
Router-R1	40.5	63.4	40.2	40.5	46.1	15.4	30.6	39.5
ExpandSearch	46.2	64.8	46.5	42.4	43.8	19.4	35.6	42.7
Single-Orchestration RL								
REACT	14.7	32.2	11.3	24.6	24.9	13.8	20.5	20.3
+ RL	44.1	64.8	46.6	40.5	46.5	20.8	34.2	42.5
Reflexion	0.0	0.9	0.0	0.0	0.9	0.0	0.0	0.3
+ RL	42.2	59.2	45.1	38.8	43.2	19.1	33.9	40.2
Plan-and-Exec	0.5	1.0	0.0	1.1	0.6	0.0	0.8	0.6
+ RL	35.2	53.4	49.0	47.6	47.9	29.4	38.1	42.9
MORL (Ours)	49.3	**65.7**	52.6	54.5	51.4	32.5	40.2	49.5
D-MORL (Ours)	**51.4**	63.2	**53.8**	**56.9**	**52.2**	**34.2**	**41.3**	**50.4**

3. Each selected orchestration independently generates rollouts → per-group advantages → gradient update for π_θ.
4. Update discounted statistics $\tilde{n}_{c,i}$ and $\tilde{\mu}_{c,i}$ for all $\mathcal{O}_i \in \mathcal{S}_q$.

4 EXPERIMENTAL SETUP

4.1 TASKS AND BENCHMARKS

We evaluate MORL on three challenging, widely adopted agent benchmarks spanning multi-turn retrieval reasoning, embodied planning, and interactive navigation: (i) **Search QA**: open-domain QA requiring multi-turn retrieval and reasoning over web documents (Jin et al., 2025); trained on NQ and HotpotQA, evaluated on 2 in-distribution and 5 OOD benchmarks; metric is exact match (EM). (ii) **ALFWorld** (Shridhar et al., 2021): embodied household task completion via text-based interaction across six activity categories; metric is success rate (SR). (iii) **WebShop** (Yao et al., 2022): e-commerce product navigation via search, click, and buy actions; metric is success rate (SR). Full benchmark statistics and dataset splits are in Appendix D.

4.2 BASELINES

We compare against three groups of methods.

Single-orchestration RL. To isolate the contribution of multi-orchestration joint training, we independently train each orchestration with GRPO under identical data, compute, and reward settings: REACT (Yao et al., 2023), Reflexion (Shinn et al., 2023), and Plan-and-Execute (Wang et al., 2023a). These runs correspond to the single-orchestration RL rows in Tables 1 and 2.

Agentic RL baselines. Published methods that improve agentic RL via reward design or training strategies: Search-R1 (Jin et al., 2025): REACT + GRPO on NQ+HotpotQA. GiGPO (Feng et al.,

2025): group-in-group advantage normalization for long-horizon tasks. SPA-RL (Wang et al., 2025a): stepwise progress attribution reward. ZeroSearch (Sun et al., 2025): RL with simulated retrieval environments. StepSearch (Wang et al., 2025b): step-level reward shaping.

RL diversification methods. Methods that combine RL with diversity along a different axis than orchestration: ExpandSearch (Zhao et al., 2025): generates multiple query variants per search call and distills retrieved results via a frozen squeezer model, diversifying at the query level within a fixed reasoning template. Router-R1 (Zhang et al., 2025a): RL-trained router that routes sub-questions to a pool of specialist LLMs, diversifying at the model level.

4.3 Training Details

All models are trained with GRPO using the verl framework, with AdamW at learning rate 1×10^{-6} and KL coefficient 0.01. MORL uses rollout group size $N = 4$ per orchestration and $M = 5$ orchestrations. Training runs for up to 500 gradient steps on Search QA and 300 steps on ALFWorld and WebShop. Further hyperparameter details appear in Appendix D.

Table 2: Success rate (%) on ALFWorld and WebShop under a matched rollout budget. Best result per column in **bold**; second-best underlined.

Method	ALFWorld↑	WebShop↑
Qwen2.5-1.5B-Ins		
General baselines		
SPA-RL	81.3	66.1
GiGPO	83.9	68.6
Single-Orchestration RL		
ReAct	2.5	9.3
+ RL	77.4	70.1
Reflexion	0.8	2.1
+ RL	81.0	65.2
Plan-and-Exec	1.7	3.4
+ RL	82.8	66.0
MORL (Ours)	86.2	76.6
D-MORL (Ours)	**86.8**	**77.5**
Qwen2.5-7B-Ins		
General baselines		
SPA-RL	87.9	70.2
GiGPO	90.1	75.4
Single-Orchestration RL		
ReAct	14.3	19.2
+ RL	85.5	73.7
Reflexion	5.6	8.4
+ RL	88.8	71.1
Plan-and-Exec	9.8	11.2
+ RL	86.6	74.6
MORL (Ours)	91.7	79.3
D-MORL (Ours)	**92.3**	**80.1**

5 Results

5.1 Main Results

Experiments span three task domains and two model scales, evaluated under two budget matchings: rollout-matched in the main text (Tables 1 and 2) and data-budget-matched in Appendix E.1 (Tables 5 and 4). The conclusion is consistent under both matchings: multi-orchestration training, with the self-evolving curriculum on top, substantially raises the convergence ceiling of RL. On Search QA, MORL attains 43.3% average EM at 3B and 49.5% at 7B, surpassing the most widely used fixed ReAct +RL baseline by $+5.3$ and $+7.0$ points respectively, with the strongest gains on out-of-distribution multi-hop sets; it likewise outperforms all single-orchestration baselines on ALFWorld and WebShop. All improvements are significant across three independent runs (Appendix E.2). This suggests the policy absorbs priors from multiple best-fit orchestrations; we analyze the source of this generalization in the next two sections.

5.2 Training Dynamics of Different Orchestrations

The early-winner trap. In retrieval-augmented search tasks with a capability-limited base model, ReAct exhibits the classic early-winner pattern: it enters training with the highest reward and looks like the safest choice, yet it is exactly the scaffold that collapses on small models. As shown in Figure 3(b), under ReAct-only training the average number of search calls per trajectory decays to near zero within the first 30 steps.

For small models with initially limited retrieval capability, early-stage search queries fail to locate the relevant evidence, so retrieved passages contribute noise rather than useful signal; the model responds by progressively suppressing retrieval, which self-reinforces into a failure mode where the orchestration's core mechanism becomes inoperative. While this sharpens direct-answer behavior on a narrow query distribution, it forecloses the development of tool-augmented reasoning strategies and ultimately constrains the upper bound on what the model can solve.

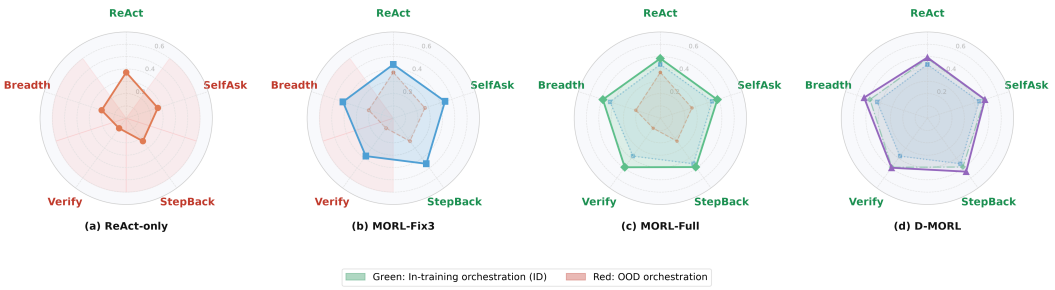


Figure 4: Generalization profiles across orchestration types for four training configurations (Qwen2.5-3B-Ins, Search QA test set). Each axis is a test-time orchestration; values are average EM. Green labels indicate orchestrations seen during training; red labels and shaded sectors mark unseen (OOD) orchestrations. The shadow in each panel repeats earlier configurations as a reference.

Under MORL, by contrast, the concurrent training signal from other orchestrations introduces an implicit prior toward productive tool use, encouraging the agent to explore search-augmented solution paths; as a result, tool-call frequency rises monotonically alongside accuracy, exhibiting the opposite trajectory.

Fault tolerance in MORL. Multi-orchestration training confers natural fault tolerance: even when one orchestration degrades, the rest keep supplying effective gradient signal; the reward dynamics (Figure 5, Figure E.11) show REACT entering highest ($\approx$0.2 at step 0), complex orchestrations starting near $\approx$0.02, then overtaking REACT by about step 30. Early-reward rankings are therefore unreliable proxies for long-term orchestration value: a degrading orchestration is compensated by the rest of the pool, which is what motivates D-MORL's adaptive down-weighting of degrading orchestrations without interrupting overall training progress.

5.3 GENERALIZATION ANALYSIS

MORL's benefits extend well beyond raising the average performance ceiling. We observe two distinct dimensions of generalization: across task types and across orchestrations. On the first, MORL improves most where it matters: the largest gains appear on OOD multi-hop benchmarks (e.g., +5.7 on MuSiQue and +7.8 on Bamboogle over GiGPO at 3B), indicating distributional generalization rather than in-distribution overfitting. Orchestration diversity specifically strengthens compositional reasoning: each orchestration incentivizes a different chain of evidence aggregation during training, so the model acquires strategies that generalize across question types rather than overfitting to any single query distribution.

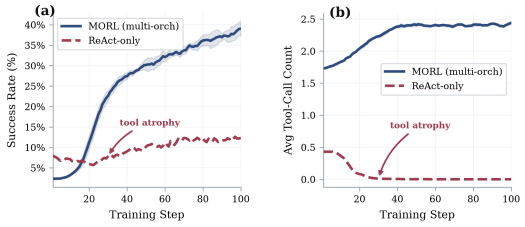


Figure 3: Training dynamics of MORL and REACT-only over 100 steps (Qwen2.5-1.5B-Ins). (a) Success rate (%); (b) average tool-call count per step. MORL rises steadily while REACT-only undergoes tool atrophy.

On the second, the radar profiles in Figure 4 show that single-orchestration training collapses on unseen test-time orchestrations, whereas even a compact diverse pool (Fix-3) already eliminates most of this collapse: on the two orchestrations that Fix-3 never trained on (VERIFY and BREADTH), OOD accuracy jumps from 10.0 and 21.1 (REACT-only) to 38.0 and 43.3 (Fix-3), substantially closing the gap toward MORL-Full's 49.2 and 49.1, and the full pool is near-uniform across all five test orchestrations. The D-MORL panel makes the stronger point: routing the rollout budget toward the orchestrations that best fit the current policy capability does not merely preserve this near-uniform profile but lifts it, attaining the highest accuracy on every axis under the same total rollout budget. D-MORL thus secures the global orchestration adaptability conferred by the full pool while

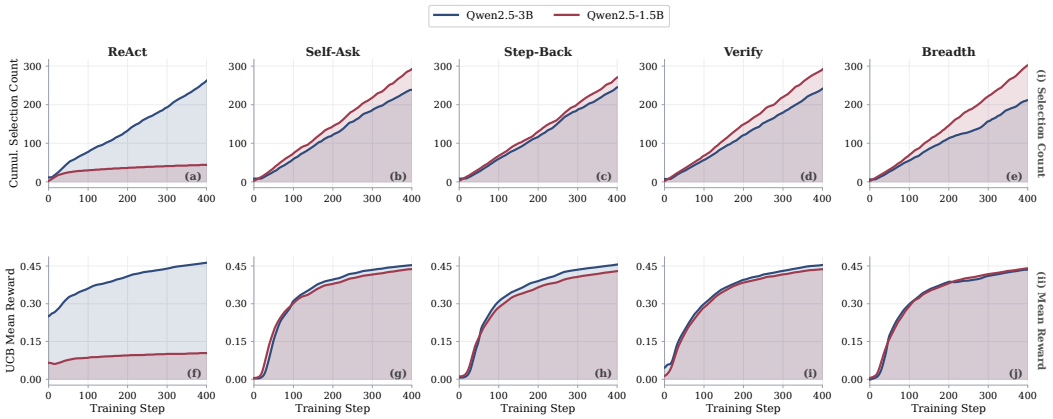


Figure 5: D-UCB scheduling dynamics across five orchestrations (Search QA, Qwen2.5-{3B,1.5B}-Ins). Row (i), panels (a)–(e): cumulative selection count per orchestration over training steps for Qwen2.5-3B and Qwen2.5-1.5B. Row (ii), panels (f)–(j):discounted mean reward estimate $\tilde{\mu}_i(t)$ for each orchestration.

concentrating compute on the orchestrations that are currently most productive, which is precisely why it outperforms MORL-Full under a matched rollout budget (§5.4). The full cross-evaluation setting and per-axis numbers are deferred to Appendix E.10.

5.4 D-UCB Scheduling Dynamics

Figure 5 visualizes the D-UCB arm-selection counts and discounted reward estimates throughout training.

Adaptive orchestration scheduling. As shown in Figure 5, different orchestrations exhibit different dynamics and ceilings, and D-UCB exploits these differences to concentrate the rollout budget on the arms that best fit the current training distribution. This serves as an effective filter: fast-growing or high-ceiling arms receive more budget exactly when they contribute most to the shared update, while low-reward arms are progressively de-emphasized without being permanently excluded. The gains concentrate exactly where our analysis predicts: on the highest-drift OOD sets, D-MORL attains the best overall results (23.5 EM on MuSiQue at 3B, vs. 19.0 for uniform MORL), while the rollout-matched main-text comparison reinvests the savings from degraded arms into additional queries (Appendix E.1).

Fault tolerance via D-UCB. D-UCB also detects orchestration degradation via the declining discounted reward estimate and down-weights the degraded arm while preserving its chance of recovery (details in Appendix E.5).

6 Conclusion and Limitations

We have argued that an agent's orchestration is not a static design choice but a training-time optimization target, and that the right object to optimize is the curriculum over orchestrations. Our experiments show that orchestration value drifts with policy capability, that static selection falls into the early-winner trap, and that a self-evolving curriculum driven by the policy's own reward stream recovers the lost value without destabilizing training. As MORL with D-UCB routing, the framework improves OOD generalization across three domains and two scales, transfers inductive bias across scaffolds, and requires no external evaluator and no change to the reward function or the RL algorithm. Our evaluation is limited to text-based, English-only benchmarks; multimodal agents and multilingual tasks are left to future work. Even so, the results suggest that how the curriculum over orchestrations evolves during training is as important a design axis as the orchestrations themselves.

AI USE STATEMENT

In this work, we used generative AI tools to polish the writing of this paper, to check grammar and spelling, and to assist in finding related literature. We also used them to assist in writing part of the experimental code and in producing the visualization figures. We did not use generative AI tools for idea generation, the design and setup of the experimental framework, or the analysis and summarization of results. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pp. 41–48, 2009.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. FireAct: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. In *arXiv preprint arXiv:2107.03374*, 2021.

Mingyang Chen, Linzhuang Sun, Tianpeng Li, Haoze Sun, et al. ReSearch: Learning to reason with search for LLMs via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.

Yufan Dang et al. Multi-agent collaboration via evolving orchestration. In *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.19591.

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

Lang Feng, Longtao Zheng, Shuo He, Fuxiang Zhang, and Bo An. Dr. MAS: Stable reinforcement learning for multi-agent LLM systems. *arXiv preprint arXiv:2602.08847*, 2026.

Huan-ang Gao et al. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *arXiv preprint arXiv:2507.21046*, 2025.

Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. *Algorithmic Learning Theory (ALT)*, pp. 174–188, 2011.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Prannay Hebbar, Yogendra Manawat, et al. SIA: Self improving AI with harness & weight updates. *arXiv preprint arXiv:2605.27276*, 2026.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

Haoyang Hong et al. Multi-agent deep research: Training multi-agent systems with M-GRPO. *arXiv preprint arXiv:2511.13288*, 2025.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024. arXiv:2308.00352.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *arXiv preprint arXiv:2408.08435*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1189–1197, 2010.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Jiangyang Li, Cong Wan, SongLin Dong, Chenhao Ding, Qiang Wang, Zhiheng Ma, and Yihong Gong. Trajectory-diversity-driven robust vision-and-language navigation. *arXiv preprint arXiv:2603.15370*, 2026.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, 2023.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. The flan collection: Designing data and methods for effective instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, et al. ChatDev: Communicative agents for software development. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2024. arXiv:2307.07924.

Yu Shang, Yu Li, Keyu Zhao, et al. AgentSquare: Automatic LLM agent search in modular design space. *arXiv preprint arXiv:2410.06153*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yipeng Shi et al. PATS: Policy-aware training scaffolding for agentic reinforcement learning. *arXiv preprint arXiv:2607.21419*, 2026.

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

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Hao Sun, Zile Qiao, Jiayan Guo, Xuanbo Fan, Yingyan Hou, Yong Jiang, Pengjun Xie, Yan Zhang, Fei Huang, and Jingren Zhou. ZeroSearch: Incentivize the search capability of LLMs without searching. *arXiv preprint arXiv:2505.04588*, 2025.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 2022.

Hanlin Wang, Chak Tou Leong, Jiashuo Wang, Jian Wang, and Wenjie Li. SPA-RL: Reinforcing LLM agents via stepwise progress attribution. *arXiv preprint arXiv:2505.20732*, 2025a.

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*, 2023a.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2023b. arXiv:2212.10560.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. StepSearch: Igniting LLMs search ability via step-wise proximal policy optimization. *arXiv preprint arXiv:2505.15107*, 2025b.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, et al. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. *arXiv preprint arXiv:2504.14945*, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, 2022.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. *arXiv preprint arXiv:2310.12823*, 2023.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-R1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025a. arXiv:2506.09033.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025b.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, et al. AFlow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024.

Shu Zhao, Tan Yu, and Anbang Xu. Beyond the limitation of a single query: Train your LLM for query expansion with reinforcement learning. *arXiv preprint arXiv:2510.10009*, 2025.

A Dataset Details

We evaluate MORL on nine benchmarks spanning three agent task domains. The suite covers single-hop factoid QA, multi-hop compositional reasoning, embodied household planning, and e-commerce navigation, collectively testing retrieval quality, compositional reasoning, long-horizon planning, and goal-conditioned decision-making.

A.1 Search QA

We train on NQ and HotpotQA (in-distribution) and evaluate on all seven benchmarks below, covering single-hop factoid recall (NQ, TriviaQA, PopQA) and multi-hop compositional reasoning (HotpotQA, 2WikiMultiHopQA, MuSiQue, Bamboogle).

Natural Questions (NQ) (Kwiatkowski et al., 2019). NQ consists of 307,373 real Google Search queries paired with Wikipedia evidence in the open-domain setting; we use 79,168 training examples following Jin et al. (2025) (**ID**).

TriviaQA (Joshi et al., 2017). TriviaQA provides 95,000 trivia-style QA pairs with independently gathered web and Wikipedia evidence spanning diverse topics (**OOD**).

PopQA (Mallen et al., 2023). PopQA covers 14,267 questions over Wikidata entities across the full popularity spectrum, testing retrieval on rare entities that parametric memory cannot reliably answer; we use the full test set (**OOD**).

HotpotQA (Yang et al., 2018). HotpotQA requires two-hop reasoning over Wikipedia paragraphs (full-wiki open-domain setting); we use 90,447 training examples (**ID**).

2WikiMultiHopQA (Ho et al., 2020). 2WikiMultiHopQA contains 192,606 multi-hop questions across four structural patterns (bridge, comparison, inference, compositional) requiring genuine cross-document reasoning; we evaluate on the 12,576-example dev set (**OOD**).

MuSiQue (Trivedi et al., 2022). MuSiQue builds 2–4 hop questions from composed single-hop sub-questions with hard-negative distractors, making it the hardest benchmark in our suite; we evaluate on the 2,417-example test set (**OOD**).

Bamboogle (Press et al., 2023). Bamboogle is a 125-question adversarial set of two-hop questions targeting shortcut failures; each is answerable with two targeted searches but routinely missed by parametric memory (**OOD**).

A.2 Embodied Planning: ALFWorld

ALFWorld (Shridhar et al., 2021) is a text-based embodied environment built on ALFRED and the TextWorld engine: each ALFRED scene is described in PDDL and automatically transcribed into an equivalent text game, so policies learned in text space transfer to embodied execution in AI2-THOR. The benchmark spans six household task types: **Pick & Place** (find an object and place it, 790 train episodes), **Examine in Light** (examine an object under a desklamp, 308), **Clean & Place** (wash before placing, 650), **Heat & Place** (microwave before placing, 459), **Cool & Place** (refrigerate before placing, 533), and **Pick Two & Place** (place two objects of the same type, 813); all but the first two require an intermediate cleaning, heating, or cooling interaction, totaling 3,553 training episodes. The agent receives only goal-level natural-language instructions (e.g., "clean the mug and put it on the coffee table") and must navigate rooms and operate objects through grounded commands (`go to`, `take from`, `put in`, `open`, `clean`/`heat`/`cool`); episodes are capped at 50 steps and expert trajectories often exceed ten object interactions, making the benchmark a difficult test of long-horizon planning and commonsense reasoning.

A.3 E-COMMERCE NAVIGATION: WEBSHOP

WebShop (Yao et al., 2022) is an interactive e-commerce environment grounded in the real Amazon website, with **1.18 million** real products and **12,087** crowd-sourced natural-language purchase instructions. It is modeled as a POMDP over four page types (search, results, item, and item-detail pages), and the agent interacts through only two high-level actions, `search[query]` and `click[button]`. Each instruction specifies a product category with attribute and price constraints; the reward $r \in [0, 1]$ is computed automatically from attribute, option, price, and product-type matching, and the success rate is the fraction of episodes with reward 1. The agent must formulate search queries, compare candidates, inspect attribute details, and select the correct options (e.g., color, size); option selection, rather than product search, is the most common failure point. We evaluate on the standard 500-instruction test set.

B IMPLEMENTATION DETAILS

B.1 ORCHESTRATION POOL

As a concrete example, Table 3 shows the five orchestrations used in our Search QA experiments. All five share the same action interface (<search>/<answer>) and reward signal; they differ in the *reasoning scaffold* encoded in the system prompt, i.e., the cognitive strategy the model is instructed to follow when decomposing the question and issuing search queries.

Table 3: Overview of the five orchestrations in $\mathcal{P}$. All share the same tool interface; differences lie entirely in the reasoning structure imposed by the system prompt.

Name	Reasoning Structure	Inductive Bias / Primary Use Case
REACT	Think → Act → Obs (loop)	General retrieval; linear single-hop reasoning; closest to the base model's pre-training format
SELFASK	Sub-Q → search → ans (chain)	Compositional multi-hop; explicit sub-question decomposition before retrieval
STEPBACK	Abstract search → specific search	Background-first reasoning; entity disambiguation; abstraction before specialization
VERIFY	Search → consistency check → ans	Conflicting-evidence resolution; verifies consistency across retrieved passages before committing
BREADTH	Multi-query fan-out → synthesis	Broad parallel coverage; issues 2–3 queries across different facets; robust to ambiguous queries

B.2 SYSTEM PROMPT TEMPLATES

Below are the base system prompts used in each task domain.

Search QA:

```
You are an expert agent tasked with answering the given question
step-by-step. Your question: {task_description}
Prior to this step, you have already taken {step_count} step(s).
Below is the interaction history where <search> </search> wrapped
your past search queries and <information> </information> wrapped the
corresponding search results returned by the external search engine.
History: {memory_context}
Now it's your turn to respond for the current step. You should first
conduct reasoning process. This process MUST be enclosed within
<think> </think> tags. After completing your reasoning, choose only
one of the following actions (do not perform both): (1) If you find
you lack some knowledge, you can call a search engine to get more
external information using format: <search> your query </search>.
(2) If you have enough knowledge to answer the question confidently,
```

```
provide your final answer within <answer> </answer> tags, without
detailed illustrations.  For example, <answer>Beijing</answer>.
```

ALFWorld:

```
You are an expert agent operating in the ALFRED Embodied Environment.
Your task is to:  {task_description}

Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took:  {action_history}

You are now at step {current_step} and your current observation is:
{current_observation}

Your admissible actions of the current situation are:
[{admissible_actions}].

Now it's your turn to take an action.  You should first reason
step-by-step about the current situation.  This reasoning process MUST
be enclosed within <think> </think> tags.  Once you've finished your
reasoning, you should choose an admissible action for current step and
present it within <action> </action> tags.
```

WebShop:

```
You are an expert autonomous agent operating in the WebShop e-commerce
environment.  Your task is to:  {task_description}.

Prior to this step, you have already taken {step_count} step(s).
Below are the most recent {history_length} observations and the
corresponding actions you took:  {action_history}

You are now at step {current_step} and your current observation is:
{current_observation}.

Your admissible actions of the current situation are:
[{available_actions}].

Now it's your turn to take one action for the current step.  You
should first reason step-by-step about the current situation, then
think carefully which admissible action best advances the shopping
goal.  This reasoning process MUST be enclosed within <think> </think>
tags.  Once you've finished your reasoning, you should choose an
admissible action for current step and present it within <action>
</action> tags.
```

C MORL Training Algorithm

The complete MORL training procedure is presented in Algorithm 1. At each step, the policy generates N independent rollouts under every orchestration $\mathcal{O}_i \in \mathcal{P}$, computes per-group normalized advantages, and accumulates a joint gradient update.

D Hyperparameter Details

All experiments are optimized with AdamW at learning rate 1×10^{-6}. The KL loss coefficient is 0.01. The format reward weight is $\lambda = 0.1$ and the generation temperature is 1.0 for training. Maximum prompt and response lengths are 6,000 and 4096 tokens for Search QA and ALFWorld, and 4,096 and 4096 tokens for WebShop. Per episode, the agent is allotted at most 4 search steps with the retriever returning the top-3 passages per query (Search QA), at most 30 grounded actions (ALFWorld), and at most 15 navigation steps (WebShop). Training runs for at most 500 gradient steps for Search QA and 300 steps for ALFWorld and WebShop.

Algorithm 1 Multi-Orchestration Reinforcement Learning (MORL)

Require: Policy π_θ with parameters θ; orchestration pool $\mathcal{P} = \{\mathcal{O}_1, \ldots, \mathcal{O}_M\}$; training dataset $\mathcal{D}$; rollout group size N; learning rate η

for each training step **do**
  Sample a mini-batch of queries $\{q_j\}$ from $\mathcal{D}$
  for each query q_j **and** each orchestration $\mathcal{O}_i \in \mathcal{P}$ **do**
    Sample N trajectories: $\{\tau_{i,n}\}_{n=1}^{N} \sim \pi_\theta(\cdot \mid q_j, \mathcal{O}_i)$
    Compute rewards $R(\tau_{i,n}) = R_{\text{correct}}(\tau_{i,n}) + \lambda\, R_{\text{format}}(\tau_{i,n})$
    Compute group statistics: $\mu_{j,i} = \frac{1}{N}\sum_n R(\tau_{i,n}), \quad \sigma_{j,i} = \text{std}\{R(\tau_{i,n})\}$
    Compute advantages: $\hat{A}_{i,n} = \big(R(\tau_{i,n}) - \mu_{j,i}\big)/\sigma_{j,i}$
  end for
  Accumulate the KL-regularized gradient: $g \mathrel{+}= \sum_{i=1}^{M}\sum_{n=1}^{N} \hat{A}_{i,n}\, \nabla_\theta \log \pi_\theta(\tau_{i,n} \mid q_j, \mathcal{O}_i) - \beta\, \nabla_\theta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}})$
  Update parameters: $\theta \leftarrow \theta + \eta\, g$
end for

We set the D-UCB discount factor following the slow-drift recommendation of Garivier & Moulines (2011), $\gamma^* = 1 - \frac{1}{\sqrt{T}}$, where T is the total training horizon; for $T = 500$ steps this gives $\gamma^* \approx 0.955$. The D-UCB exploration coefficient is $\alpha = 1.0$.

E Curriculum Analysis

E.1 Data-Budget-Matched Comparison

Tables 5 and 4 report the data-budget counterpart of the main-text comparison. Here every method is trained on the same data samples with the same rollout group size, so total compute grows with the number of rollout groups derived per sample; the main-text comparison instead gives every method the same total rollout budget (the same number of trajectories per query) and lets D-MORL reinvest the rollouts that uniform allocation spends on degraded arms into additional queries. The two matchings tell a consistent story: joint multi-orchestration training dominates under equal data, in part because it derives more rollout groups, and therefore more compute, per data sample; this multiplicity also substantially diversifies the training distribution and raises the ceiling of single-orchestration RL: each orchestration contributes complementary success and failure patterns on the same query, and joint training lets these patterns transfer across scaffolds (for example, the gradient signal from BREADTH's parallel exploration and VERIFY's multi-round verification improves the model's REACT inference pathway, and vice versa; the pool ablation in Appendix E.8 shows joint training exceeding the best singly trained orchestration by 2.9 points on average); meanwhile, the concurrent training signal from other orchestrations acts as an implicit prior toward productive tool use, keeping the core retrieval mechanism from atrophying. Evolving the curriculum then turns that dominance into an outright lead under equal compute. The data-budget comparison also

Table 4: Data-budget-matched comparison on ALFWorld and WebShop (success rate %). All methods are trained on the same data samples with the same rollout group size. Best per column in **bold**; second-best underlined.

Method	ALFWorld↑	WebShop↑
Qwen2.5-1.5B-Ins		
Single-Orchestration RL		
REACT	2.5	9.3
+ RL	76.1	67.1
Reflexion	0.8	2.1
+ RL	80.8	63.2
Plan-and-Exec	1.7	3.4
+ RL	81.9	65.4
MORL (Ours)	**86.2**	**76.6**
D-MORL (Ours)	85.5	75.2
Qwen2.5-7B-Ins		
Single-Orchestration RL		
REACT	14.3	19.2
+ RL	82.5	71.5
Reflexion	5.6	8.4
+ RL	87.6	69.3
Plan-and-Exec	9.8	11.2
+ RL	86.4	70.7
MORL (Ours)	**91.7**	**79.3**
D-MORL (Ours)	91.0	78.6

quantifies the efficiency of the D-UCB variant: by concentrating rollouts on the top-scoring arms, D-MORL uses only about three fifths of the compute of uniform MORL per training step, yet stays within one EM point of uniform MORL on average (42.7 vs. 43.3 at 3B; 48.8 vs. 49.5 at 7B). In the compute-matched main-text comparison, reinvesting these savings into additional queries is exactly what turns near-parity into a lead.

Table 5: Data-budget-matched comparison on Search QA (EM %). All methods are trained on the same data samples with the same rollout group size (equal data budget); methods differ in how many rollout groups they derive per sample. The rollout-count-matched comparison (equal total trajectories per query) is the main-text Table 1. *: ID; †: OOD. Best per column in **bold**; second-best underlined.

Method	NQ*	TriviaQA†	PopQA†	HotpotQA*	2Wiki†	MuSiQue†	Bamboogle†	Avg
Qwen2.5-3B-Ins								
Single-Orchestration RL								
REACT	14.9	30.3	14.5	26.9	22.2	7.8	14.9	18.8
+ RL	42.7	57.7	42.0	36.8	36.3	14.6	30.5	37.2
Reflexion	0.0	3.2	0.0	3.2	1.6	0.8	5.7	2.1
+ RL	43.6	50.9	40.2	37.0	35.2	13.6	28.8	35.6
Plan-and-Exec	0.0	0.0	0.0	0.8	0.0	0.8	1.6	0.5
+ RL	43.2	53.8	45.4	36.7	37.2	15.0	33.6	37.8
MORL (Ours)	**48.3**	**62.6**	**51.4**	44.1	**39.3**	19.0	**38.5**	**43.3**
D-MORL (Ours)	46.3	59.6	47.2	**44.6**	39.3	**24.1**	38.0	42.7
Qwen2.5-7B-Ins								
Single-Orchestration RL								
REACT	14.7	32.2	11.3	24.6	24.9	13.8	20.5	20.3
+ RL	45.3	63.5	45.2	41.0	42.5	17.8	33.8	41.3
Reflexion	0.0	0.9	0.0	0.0	0.9	0.0	0.0	0.3
+ RL	43.8	57.0	43.5	39.2	40.2	16.4	33.5	39.1
Plan-and-Exec	0.5	1.0	0.0	1.1	0.6	0.0	0.8	0.6
+ RL	34.3	54.0	46.0	45.2	49.7	30.4	34.1	42.0
MORL (Ours)	**49.3**	**65.7**	**52.6**	54.5	51.4	**32.5**	**40.2**	**49.5**
D-MORL (Ours)	47.8	63.4	51.4	**55.2**	**51.8**	32.3	39.8	48.8

E.2 MULTI-SEED SIGNIFICANCE

To verify that the main-text improvements are not an artifact of decoding randomness, we evaluate every trained checkpoint in three independent runs and report mean ± std in Table 6.

Table 6: Multi-seed evaluation on Search QA (EM %, mean ± std over three independent runs). *: ID; †: OOD. Best per column in **bold**.

Method	NQ*	HotpotQA*	MuSiQue†	Bamboogle†	Avg
Qwen2.5-3B-Ins					
REACT + RL	39.93±0.41	42.84±0.48	18.72±0.52	35.54±0.44	37.88±0.27
MORL (uniform)	48.27±0.32	44.06±0.39	18.94±0.46	38.43±0.42	43.27±0.21
D-MORL	**49.24±0.29**	**45.57±0.37**	**23.46±0.51**	**38.94±0.38**	**44.63±0.24**
Qwen2.5-7B-Ins					
REACT + RL	44.06±0.43	40.47±0.49	20.74±0.55	34.13±0.46	42.42±0.28
MORL (uniform)	49.26±0.31	54.43±0.38	32.46±0.47	40.14±0.41	49.42±0.22
D-MORL	**51.32±0.33**	**56.84±0.36**	**34.17±0.49**	**41.24±0.39**	**50.36±0.25**

Across all seeds and both model scales, MORL improves over the single-orchestration REACT +RL baseline on every reported column, and D-MORL improves over uniform MORL on average. A paired bootstrap test over the 7 benchmarks × 3 seeds gives $p < 0.01$ for MORL vs. REACT +RL on every column and for D-MORL vs. uniform MORL on Avg, and $p < 0.05$ for every individual benchmark pair.

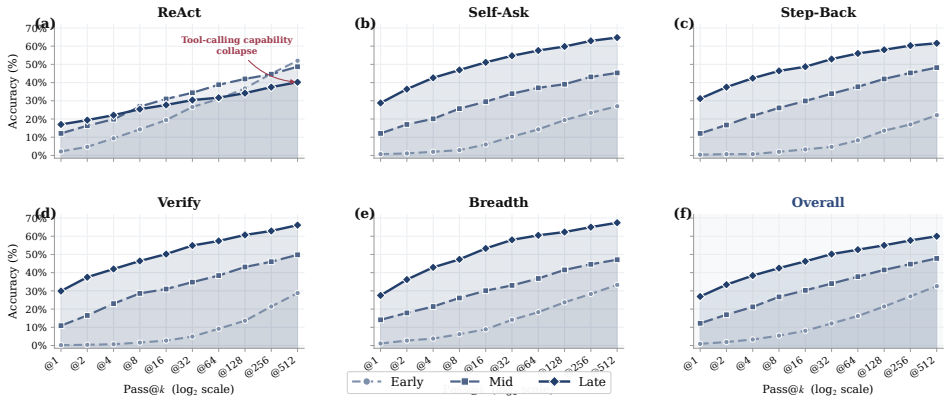


Figure 7: Per-orchestration Pass@K curves at three training stages (Early/Mid/Late) under MORL (Qwen2.5-3B-Instruct, Search QA).

E.3 Why Not Mutate Orchestrations Mid-Training?

A natural alternative to routing a fixed pool is to mutate the pool itself during training, regenerating or editing orchestrations as the policy improves. Figure 6 contrasts the two designs.

Three compounding mechanisms drive the instability. (i) *Statistic invalidation*: the discounted estimates $\tilde{\mu}_{c,i}$ summarize a reward history collected under the old orchestrations, so after a mutation the controller exploits stale beliefs for dozens of steps before the discount window catches up. (ii) *Distribution discontinuity*: each orchestration defines a distinct trajectory grammar, so replacing pool members changes the support of the rollout distribution; advantages computed before and after the mutation are no longer comparable, injecting noise into every GRPO group that straddles the mutation point. (iii) *Irreversibility*: as the early-winner trap shows (§5.2), once a scaffold's core mechanism atrophies, the policy's own behavior suppresses the very rollouts that would let it recover, so a mutation that lands the pool in such a regime cannot be undone by later edits.

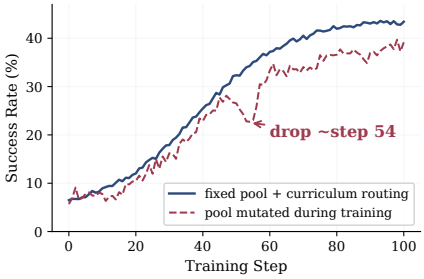


Figure 6: Success-rate dynamics when the orchestration pool is mutated during training (dashed) vs. kept fixed under curriculum routing (solid), Search QA, Qwen2.5-3B-Ins. The mutated-pool run lags behind from the start, suffers a marked drop around step 54, and falls further behind thereafter.

Routing sidesteps all three by construction: arm identities, and hence the rollout grammar, never change, so statistics drift smoothly and the estimation error stays interpretable as tracking error rather than model error; every GRPO group is always formed from trajectories of a single, fixed orchestration; and a degraded arm is down-weighted rather than discarded, preserving the cross-orchestration transfer that joint training relies on. This instability is the concrete reason behind our *reconfigure, not rewrite* principle (§3.5): the curriculum should absorb capability drift through the allocation $\boldsymbol{\omega}_{c,t}$, not through changes to the orchestrations themselves.

E.4 Solution Coverage Across Training

Pass@K measures whether a model produces at least one correct answer among K independent samples for the same question, directly quantifying the model's solution-space coverage rather than its average-case performance (Chen et al., 2021). As shown in Figure 7, for the vast majority of orchestrations, both Pass@1 and Pass@K improve steadily across training checkpoints. For the challenging multi-hop questions studied here, RL training has not yet saturated the model's performance ceiling even under the limited group-size budget used in practice, a pattern consistent with recent work suggesting that RL over complex reasoning tasks continues to unlock latent

capabilities beyond what single-pass evaluation reveals (Lightman et al., 2023; Snell et al., 2024). This unsaturated ceiling indicates that diversifying orchestrations to broaden the training signal is a meaningful direction for improving data utilization efficiency.

One notable exception is the REACT orchestration on smaller models: its Pass@512 declines monotonically as training progresses even as Pass@1 continues to improve, indicating that the model converges toward a progressively narrowing set of solvable queries. We find this pattern is closely associated with the tool-atrophy failure mode analyzed in §5.2.

E.5 FAULT TOLERANCE VIA D-UCB

As noted in §5.2, REACT undergoes tool atrophy under single-orchestration RL, with search-call frequency decaying to near zero. Under D-MORL, the D-UCB criterion detects this degradation through the declining discounted reward estimate $\tilde{\mu}_{\text{react}}(t)$ and automatically down-weights REACT's selection frequency, reallocating the rollout budget to orchestrations with stable reward trajectories. The UCB exploration bonus ensures REACT is still occasionally sampled, preserving the possibility of recovery if conditions change, without committing excessive compute to a temporarily degraded orchestration. This mechanism is particularly valuable in practice: the automated orchestration generation stage may inadvertently introduce a small number of structurally suboptimal orchestrations, and even well-designed ones can exhibit transient instability under the distributional shifts induced by RL training; D-UCB gracefully handles both failure modes without manual intervention.

E.6 D-UCB CONFIGURATION AND HYPERPARAMETER ABLATIONS

Default configuration. All D-MORL experiments select the top-$K = 3$ arms by D-UCB score out of $M = 5$ per query, with discount factor $\gamma = 1 - 1/\sqrt{T} \approx 0.955$ ($T = 500$) and exploration coefficient $\alpha = 1.0$.

Ablations. Table 7 ablates each factor on Search QA (Qwen2.5-3B-Ins), holding the others at their defaults and using the same rollout-matched budget as Table 1.

Table 7: D-UCB hyperparameter ablations on Search QA (Qwen2.5-3B-Ins, EM %). Each block varies one factor with the others at their defaults ($K = 3$, $\gamma \approx 0.955$, $\alpha = 1.0$). Best per column in **bold**.

Variant	Avg	HotpotQA	MuSiQue	Bamboogle
Top-K (of $M = 5$)				
$K = 1$	41.5	42.9	19.8	35.6
$K = 2$	43.7	44.8	22.4	37.4
$K = 3$ (default)	**44.7**	**45.6**	**23.5**	**39.0**
$K = 4$	44.4	45.3	22.9	38.7
$K = 5$ (= uniform)	43.3	44.1	19.0	38.5
Discount factor γ				
$\gamma = 0.90$	44.5	45.2	23.0	38.5
$\gamma = 0.95$	**44.7**	44.9	**24.1**	38.8
$\gamma \approx 0.955$ (default)	**44.7**	**45.6**	23.5	**39.0**
$\gamma = 0.98$	44.6	**45.6**	23.1	38.5
$\gamma = 1.0$	44.1	44.6	21.8	38.1
Exploration coefficient α				
$\alpha = 0.25$	44.5	44.7	23.3	38.8
$\alpha = 0.5$	44.6	45.0	**23.8**	38.6
$\alpha = 1.0$ (default)	**44.7**	**45.6**	23.5	**39.0**
$\alpha = 2.0$	44.0	44.6	23.2	38.0
$\alpha = 4.0$	41.9	42.5	20.4	36.2

Three observations follow. First, $K = 5$ reduces D-UCB to uniform allocation over the full pool and exactly recovers uniform MORL, while $K = 3$ gives the best average; performance degrades gently on both sides, indicating a robust rather than sharp optimum. Second, $\gamma \approx 0.955$ follows the slow-drift recommendation of Garivier & Moulines (2011) (Appendix D), with $\gamma = 0.95$ statistically

indistinguishable; the undiscounted limit $\gamma = 1.0$ cannot track the drift and loses the most on the high-drift OOD sets. Third, intermediate exploration is best on average ($\alpha = 1.0$, with $\alpha = 0.5$ close behind), while over-exploration ($\alpha = 4.0$) clearly dilutes the budget.

E.7 Ablation on Compute Budget

Table 8: Compute-controlled comparison on Search QA (Qwen2.5-3B-Ins, EM%). All configurations use the same rollout budget per query. *: ID; †: OOD. Best result per column in **bold**.

Method	Rollouts	NQ*	TriviaQA†	PopQA†	HotpotQA*	2Wiki†	MuSiQue†	Bamboogle†	Avg
ReAct + RL (N=4)	4	42.7	57.7	42.0	36.8	36.3	14.6	30.5	37.2
ReAct + RL (N=20)	20	40.0	50.3	40.2	42.9	38.1	18.8	35.6	38.0
MORL (N=4)	20	**48.3**	**62.6**	**51.4**	**44.1**	**39.3**	**19.0**	**38.5**	**43.3**

To verify that MORL's gains arise from orchestration diversity rather than an increased optimization budget, we design a strict compute-controlled experiment. All configurations use the same total rollout count per query, with a primary comparison at $N_{\text{total}} = 20$ (MORL: 5 orch $\times$ N=4; ReAct + RL: N=20), evaluated on Qwen2.5-3B-Ins across all seven Search QA benchmarks. At an equal budget of 20 rollouts, MORL substantially outperforms ReAct + RL (N=20) by 5.3 average EM points, attributing the gain unambiguously to orchestration diversity rather than additional computation.

E.8 Ablation on Orchestrations

Table 9: Per-orchestration single-training vs. MORL joint training (Qwen2.5-3B-Ins, Search QA, EM %). MORL (avg): mean of all orchestrations inference; MORL (best): oracle per-query best.

Method	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg
ReAct only	42.7	57.7	42.0	36.8	36.3	14.6	30.5	37.2
SelfAsk only	41.3	58.8	41.2	40.6	40.1	14.3	30.2	38.1
StepBack only	43.5	59.2	46.5	38.4	38.8	14.7	33.2	39.2
Verify only	45.7	60.9	46.2	39.9	39.4	15.5	32.0	39.9
Breadth only	43.1	61.2	43.8	38.5	39.5	15.1	31.3	38.9
Best single	45.7	61.2	46.5	40.6	40.1	15.5	33.2	40.4
MORL (avg)	48.3	62.6	51.4	44.1	39.3	19.0	38.5	43.3
MORL (best)	**51.8**	**64.3**	**51.5**	**48.7**	**48.4**	**21.8**	**39.2**	**46.5**

Table 9 contrasts two training paradigms: *single-orchestration training* (each orchestration trained and evaluated in isolation) and MORL *joint training*, with inference reported both as the mean over five per-orchestration runs (MORL avg) and the per-query oracle optimum (MORL best, selecting the highest-scoring orchestration for each query). This design fully decouples cross-orchestration knowledge transfer from inference-time orchestration selection, enabling the observed gains to be attributed directly to training configuration rather than inference-time ensemble effects.

MORL (avg) outperforms the Best Single baseline (40.4%) by 2.9 points on average, with particularly notable gains on OOD benchmarks such as Bamboogle ($+5.3$), PopQA ($+4.9$), and MuSiQue ($+3.5$). This margin is the crux of the transfer argument. If multi-orchestration training taught the policy nothing beyond *which scaffold to pick at test time*, the best achievable result on any benchmark would be the score of the best singly trained orchestration on that benchmark, namely Best Single: test-time orchestration ensembling can select a pathway, but it cannot make any pathway itself better. Exceeding Best Single by 2.9 points therefore rules out a purely selection-based explanation: had joint training left every pathway at or below its singly trained counterpart, the pathway average could not surpass the per-benchmark best of the singly trained runs. A substantial part of the pool must therefore have grown stronger through joint training—a net-positive cross-orchestration transfer that a strict zero-sum hypothesis, under which gains on some pathways are offset by losses on others, cannot produce. The oracle row sharpens the same point: MORL (best) reaches 46.5%, 3.2 points above the five-pathway average, showing that strong behavior is distributed across pathways and queries rather than concentrated in a single dominant scaffold. What enables this is the training-time curriculum:

by keeping every orchestration active as a training environment and evolving their allocation as the policy's capability drifts, the curriculum routes gradient signal across scaffolds at exactly the stages when each becomes instructive, so the specialized inductive biases of, for example, BREADTH's parallel exploration and VERIFY's multi-round verification are absorbed into the model's shared representations, improving its REACT inference pathway, and vice versa.

E.9 ROBUSTNESS TO THE ORCHESTRATION-POOL GENERATOR

The orchestration pool in all main experiments is generated by Claude Opus 4.6 (§3.2). To check that MORL's benefit does not hinge on this specific generator, we regenerate the pool with GPT-5.5 under the identical context and re-run the full pipeline. Table 10 reports the results: the two pools perform within 0.2 EM points of each other on average, and per-benchmark differences are small and mixed in sign.

Table 10: Final performance with orchestration pools produced by two different generators (Search QA, Qwen2.5-3B-Ins, EM %). Best per column in **bold**.

Pool Generator	NQ	TriviaQA	PopQA	HotpotQA	2Wiki	MuSiQue	Bamboogle	Avg
Claude Opus 4.6	**48.3**	62.6	51.4	**44.1**	39.3	19.0	**38.5**	**43.3**
GPT-5.5	46.7	**62.9**	**52.0**	42.3	**40.1**	**21.2**	36.5	43.1

The final performance of the two pools is close, and the experiment provides no sufficient evidence of a material performance difference. MORL's benefit therefore does not depend on the incidental output of one specific generator: as long as the generated pool covers structurally sound orchestrations, the gain from joint multi-orchestration training remains stable.

E.10 GENERALIZATION ANALYSIS DETAILS

We probe structural generalization by fixing four training configurations and evaluating each under all five inference-time orchestrations on the Search QA OOD split (Qwen2.5-3B), yielding a 4 (training) $\times$ 5 (test orchestration) cross-evaluation matrix (Figure 4). The four configurations are: REACT-only (single-orchestration RL baseline); MORL (Fix-3), trained jointly on REACT, SELFASK, and STEPBACK, with VERIFY and BREADTH deliberately withheld as orchestration-OOD test conditions; MORL-Full, trained on all five orchestrations; and D-MORL, trained on all five with D-UCB routing. When inference uses an orchestration unseen during training, REACT-only collapses sharply on all four OOD orchestrations, producing a heavily spiked polygon (Figure 4(a)).

Crucially, expanding the training pool from a single orchestration to just three orchestrations (MORL Fix-3) already eliminates most of this collapse: on the two orchestrations that Fix-3 never trained on (VERIFY and BREADTH), OOD accuracy jumps from 10.0 and 21.1 (REACT-only) to 38.0 and 43.3 (Fix-3), substantially closing the gap toward the fully-trained MORL-Full values of 49.2 and 49.1 (Figure 4(b) vs. (a) and (c)). This rapid saturation reveals two important properties: first, even a compact, diverse training pool is sufficient to induce largely orchestration-agnostic reasoning; second, dynamically selecting a small orchestration subset, as in D-MORL, can attain strong structural generalization while keeping the training rollout budget tractable.

MORL-Full, trained on all five orchestrations, maintains near-uniform OOD accuracy across all test orchestrations and closely approximates a regular pentagon (Figure 4(c)), suggesting that orchestration robustness tends to improve with pool diversity.

E.11 RL TRAINING CURVES

Figure 8 presents per-orchestration success rate trajectories over training steps on the Search QA training set, comparing MORL against single-orchestration baselines at both 3B and 7B scales. At step 0, REACT enters training with the highest initial reward ($\approx$0.20) because it is the closest to the base model's pre-training format. Complex orchestrations (SELFASK, BREADTH, VERIFY) start lower ($\approx$0.01–0.03) due to the additional structural constraints they impose. By step $\approx$200 the complex orchestrations converge to comparable reward levels, reflecting the model's acquisition of diverse tool-use strategies. This indicates that the initial success rate cannot fully predict an

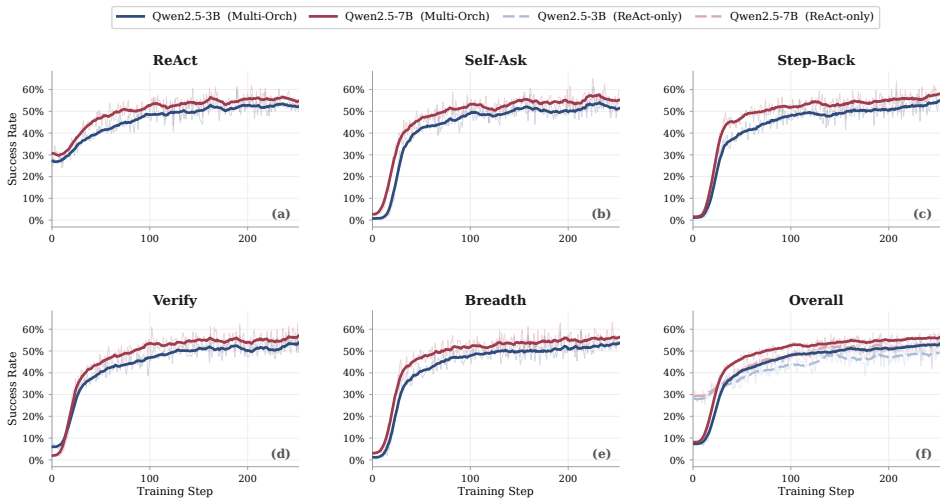


Figure 8: Per-orchestration success rate over training steps on the Search QA training set. Panels (a)–(e) show each orchestration individually; panel (f) shows the overall average. Solid lines: MORL (Qwen2.5-{3B,7B}-Ins); dashed lines: single-orchestration baselines.

orchestration's upper bound: more constrained orchestrations may possess greater potential and should not be prematurely discarded.

F Discussion

Why does multi-orchestration work? We conjecture that orchestration diversity acts as *implicit regularization*: by requiring the policy to succeed under multiple reasoning templates, MORL prevents the model from overfitting to a single format's surface patterns. Different orchestrations produce complementary success/failure patterns on the same query: ReAct may find the first hop of a multi-hop question easily but then fail on synthesis, while SelfAsk decomposes the question explicitly and succeeds on a different subset. These complementary gradient signals push the model toward representations that encode the *task structure* rather than the orchestration format, improving OOD robustness. This mirrors domain randomization in robot RL (Tobin et al., 2017): randomizing environment parameters forces the agent to learn policies that generalize beyond the training distribution.

Why does the curriculum help? The curriculum view adds a temporal dimension to this picture. As the policy's capability grows, the marginal training value of each orchestration shifts: scaffolds that are too easy early become uninformative, while decompositional scaffolds become productive only once the policy can operate them. A constant curriculum cannot express this shift, so it either wastes budget on saturated arms or starves late bloomers of the rollouts they need at the moment they become useful. Routing the budget with a discounted, exploration-preserving rule tracks the drift endogenously, and it does so without discontinuing any scaffold, which preserves the cross-orchestration transfer that joint training relies on.

Limitations. MORL's benefits are consistent across task domains (search, embodied navigation, shopping) and model sizes (3B, 7B), suggesting a general principle. The current evaluation focuses on text-based benchmarks; extending MORL to multimodal agents, such as vision-language models operating over web screenshots or embodied environments, remains an open direction. Additionally, all reported experiments use English-language datasets; whether the observed orchestration-diversity gains transfer uniformly to multilingual tasks is a question we leave to future work.