RoleFlow: Self-Evolving Multi-Role Agentic Collaboration via Structural Adaptation

Anonymous ACL submission

Abstract

Training LLM agents to adapt their collaboration structure online has converged on group-relative policy gradients, whose group-mean baseline is well calibrated when actions are content-level but breaks once the action set mutates the chain itself: free sampling conflates the effect of a specific edit with the prior over edit types, and per-edit credit cannot be recovered. To this end, we propose SECPO, Single-Edit Counterfactual Policy Optimization, which reserves one rollout per group for a forced no-edit anchor and computes the advantage of each candidate edit against this anchor rather than against the freely sampled mean. SECPO strictly extends GRPO: a single forced rollout per group recovers per-edit credit when the chain mutates, and the estimator reduces to GRPO when it does not. We instantiate SECPO inside RoleFlow, a per-query structural-adaptation framework whose single-edit-per-cycle constraint is dictated by the counterfactual estimator's well-definedness, and whose Role Library is curated across queries under counterfactual evidence. Across four task domains (multi-hop QA, mathematical reasoning, software engineering, interactive web environments), 200 in-domain examples per benchmark suffice for RoleFlow to lift out-of-distribution F1 on 2WikiMultiHopQA by 6.7 points over the strongest cohort baseline and to match the leading online-RL agent on WebShop at one-sixth of its per-query token cost. Single-edit attribution and counterfactual anchoring both contribute to these gains.

1 Introduction

Multi-agent LLM systems are increasingly trained, not designed: a recent line of work (Li et al., 2025; Song et al., 2025; Wang et al., 2025a) uses group-relative policy optimization (GRPO (Shao et al., 2024)) to learn how agents should adapt their collaboration structure online. The default recipe is straightforward. The policy samples K rollouts and scores each rollout's reward against the group mean $\bar{R}$. This recipe is well calibrated when the action is content-level (token choice, retrieval target, tool call), and the group mean approximates a stationary prior over rollout types against which individual actions can be ranked.

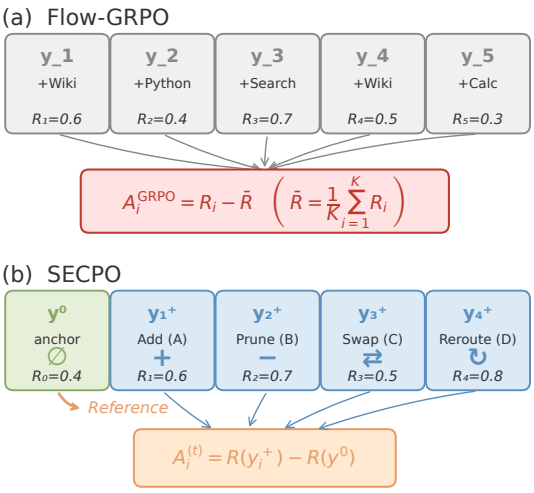


Figure 1: **Single-edit counterfactual advantage.** (a) GRPO and Flow-GRPO normalize reward against $\bar{R}$ over K freely sampled rollouts; when the action set mutates the chain itself, $\bar{R}$ conflates edit effect with edit-type prior. (b) SECPO fixes one rollout as a no-edit anchor y^0 and differences the remaining $K-1$ against it: $A_i^{(t)} = R(y_i^+) - R(y^0)$.

When the action set itself mutates the chain that produces the rollout, however, this assumption breaks. Adding, removing, or rerouting a role yields a heterogeneous group in which $\bar{R}$ conflates the effect of a specific edit with the prior over edit types (Figure 1). The policy learns that edits help on average, but per-edit credit cannot be cleanly recovered, and the router that emerges overfits to the edit distribution of its training domain.

We address this conflation with **SECPO**, Single-Edit Counterfactual Policy Optimization. In every

K-rollout group, SECPO reserves one rollout for the null edit $\varnothing$ and computes the advantage of each remaining rollout against this anchor rather than against the freely sampled mean. The change to GRPO is minimal: a single forced rollout per group. The estimator reduces to GRPO whenever no structural change occurs, so SECPO is a strict extension rather than a replacement.

To put SECPO to work we instantiate it inside **RoleFlow**, a per-query structural-adaptation framework (Figure 2). RoleFlow maintains an executable RoleFlowChain RFC_t, a Global Router GR that emits at most one structural edit per cycle drawn from $\{\textsc{Add}, \textsc{Prune}, \textsc{Swap}, \textsc{Reroute}, \varnothing\}$, and a shared Role Library $\mathcal{R}$ whose contents are curated across queries under the counterfactual gain measured against each SECPO anchor. The single-edit-per-cycle constraint is not arbitrary; it is precisely what makes the SECPO counterfactual estimator well defined, since consecutive chains then differ by exactly one structural unit and the marginal effect of that unit is what the anchor isolates.

We evaluate the SECPO-trained RoleFlow on six benchmarks spanning multi-hop question answering on HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022); mathematical reasoning on MATH-500 (Lightman et al., 2023); software engineering on SWE-bench-Lite (Jimenez et al., 2024); and web shopping on WebShop (Yao et al., 2022). With a 200-sample HotpotQA training budget, RoleFlow leads the strongest cohort baseline on three multi-hop QA benchmarks (up to $+6.7$ F1 on 2WikiMultiHopQA out of distribution) and exceeds AgentFlow on WebShop at one-sixth of its per-query token budget. Two ablations isolate the contributions of single-edit attribution and counterfactual anchoring (Table 3).

Contributions.

- We propose SECPO, a minimal extension of GRPO that replaces the freely sampled group baseline with a forced no-edit anchor, restoring per-edit credit when the action set mutates the agent's chain. The estimator reduces to GRPO when no structural change occurs.
- We instantiate SECPO inside RoleFlow, a per-query structural-adaptation framework whose single-edit-per-cycle constraint makes the counterfactual estimator well defined, and whose Role Library is curated across queries under counterfactual evidence.
- We evaluate the SECPO-trained RoleFlow on six benchmarks across four task domains and show that a 200-sample HotpotQA-trained checkpoint leads the strongest cohort baseline on three multi-hop QA benchmarks (up to $+6.7$ F1 out of distribution) and on WebShop at one-sixth of AgentFlow's per-query token budget.

2 Related Work

Static Multi-Agent Pipelines and Content-Level Adaptation. LLM-based multi-agent systems traditionally commit to fixed collaboration topologies at design time. ChatDev (Qian et al., 2024) hard-codes a software development pipeline, Chain-of-Agents (Zhang et al., 2024b) fixes a worker-manager chain, and ReAct (Yao et al., 2023) fixes an observe-think-act loop; broader orchestrators such as AgentOrchestra (Zhang et al., 2025c) and Maestro (Wang et al., 2025b) delegate roles hierarchically but leave the topology static once instantiated. A complementary line adapts what content a model sees rather than how it is composed: retrieval-augmented generation (Lewis et al., 2021; Guu et al., 2020; Borgeaud et al., 2022), tool use (Schick et al., 2023), decomposed prompting (Khot et al., 2023), and mixture-of-agents ensembling (Wang et al., 2024) expand the input space on a fixed graph. Learned routers (Rosenbaum et al., 2017; Cases et al., 2019; Pan et al., 2025; Madaan et al., 2023; Zhang et al., 2025d) and graph-based test-time composition such as Graph-of-Agents (Yun et al., 2026) select among predefined modules, yet the act of adding, removing, or rewiring an agent role is not itself a learning target.

Self-Evolving Agents and Offline Workflow Search. A recent line treats workflow structure itself as a learnable object but searches it offline. AFlow (Zhang et al., 2024a) performs MCTS over workflow graphs; EvoFlow (Zhang et al., 2025a) evolves diverse workflows via genetic operators; MaAS (Zhang et al., 2025b) searches an agentic supernet; and AgentEvolver (Zhai et al., 2025), EvolveR (Wu et al., 2025), Multi-Agent Evolve (Team, 2025), and CoMAS (Xue et al., 2025) train self-evolving lifecycles through interaction-shaped rewards. AgentNet (Yang et al., 2025) adds decentralized evolutionary coordination across agent populations. These methods learn structural choices, but the search consumes thousands of trajectories and the discovered configuration is frozen at inference; cross-domain transfer

degrades when the test distribution diverges from the search distribution.

Online Structural Adaptation and Group-Relative Credit Assignment. A recent line of work learns structural adaptation policies online at the per-query level. AgentFlow with Flow-GRPO (Li et al., 2025), R1-Searcher (Song et al., 2025), and AgentConductor (Wang et al., 2025a) optimize structural policies via group-relative advantages (Shao et al., 2024) that normalize each rollout's reward against the mean of K siblings. This recipe is well calibrated when the action is content-level (token choice, retrieval target, tool call) and the group mean approximates a stationary prior over rollout types. When the action set itself mutates the chain that produces the rollout, however, the same group mean drifts with sampling noise and provides no fixed reference against which to score a specific structural decision; credit for individual edits becomes entangled with cross-rollout variance, and the policy overfits to the edit prior of its training domain. This paper introduces SECPO to anchor the baseline against a forced null-edit rollout, decoupling per-edit credit from sampling noise.

3 Approach

3.1 Problem Setup and Notation

A task instance $x \in \mathcal{X}$ enters a persistent Global Router (GR) parameterized by θ, which owns a lifelong Role Library $\mathcal{R} = \{\rho^{(j)}\}_{j=1}^{|\mathcal{R}|}$ where each role $\rho = (r, \psi, \mathcal{U}_r)$ specifies a name, a system prompt, and a capability tag set. Execution proceeds along a *RoleFlow Chain* $\mathrm{RFC}_t = (\mathbf{V}_t, \mathbf{A}_t)$ at cycle index $t \in \{0, \ldots, T_{\max} - 1\}$, a directed acyclic graph over $\mathbf{V}_t \subseteq \mathcal{R} \cup \{\textsc{Planner}\}$ with the fixed coordinator $\textsc{Planner}$ at the head and an ordered sequence of specialists drawn from $\mathcal{R}$ between head and output. A deterministic domain validator V assesses each cycle's output and decides termination; σ_t is the per-cycle scratch memory of intermediate traces and validator history. At every cycle GR commits one structural edit $a_t \in \mathbb{E} = \{\textsc{Add}, \textsc{Prune}, \textsc{Swap}, \textsc{Reroute}, \varnothing\}$, with $\varnothing$ the *null edit* ($\mathrm{RFC}_t \oplus \varnothing = \mathrm{RFC}_t$). We index rollouts by $i \in \{0, \ldots, K-1\}$ with the anchor fixed at $i=0$, cycles by t, and tokens of the router's plan utterance by ℓ.

Cycle dynamics. Each cycle executes the current chain, validates, and (on rejection) commits exactly one edit:

$$\begin{aligned}(y_t, z_t, c_t) &= \mathrm{RFC}_t(\sigma_t, x),\\ (h_t, s_t, d_t) &= V(y_t, z_t),\\ a_t &\sim \pi_{\mathrm{GR},\theta}(\,\cdot \mid \mathrm{RFC}_t, \sigma_t, d_t), \quad a_t \in \mathbb{E},\\ \mathrm{RFC}_{t+1} &= \mathrm{RFC}_t \oplus a_t.\end{aligned} \tag{1}$$

The single-edit-per-cycle constraint makes consecutive chains differ by exactly one structural unit, which is the condition under which the single-edit counterfactual estimator below is well-defined.

Policy objective. Let $T = \min(\{t : h_t = 1\} \cup \{T_{\max}\})$ be the terminal cycle index. The trajectory $\tau_\theta(x) = (\mathrm{RFC}_0, a_0, \ldots, \mathrm{RFC}_T)$ has reward $R(\tau) := h_T + \lambda_s\, s_T$ combining the hard validator gate with the soft quality score, and training maximises $\mathbb{E}_x[R(\tau_\theta(x))]$. Sections 3.3–3.5 supply the per-edit gradient signal that this objective requires in three coupled pieces: a single-edit counterfactual advantage estimator, a token-level supplement targeting the edit tokens, and a library-update rule gated by positive counterfactual gain.

3.2 RoleFlow: Per-Query Chain Adaptation

The RFC is grown for each query rather than fixed in advance. Given x, GR hypothesizes an initial chain at cycle zero by drawing roles from $\mathcal{R}$ and executes it. Whenever the validator rejects ($h_t = 0$), GR commits a single edit that adds, removes, swaps, or reroutes one role (Eq. 1). When GR proposes a role whose name is not in $\mathcal{R}$, a LibraryExpander synthesizes a typed specification $\hat{\rho}$ from the proposed name and the current validator feedback and inserts $\hat{\rho}$ provisionally into the chain; at cycle zero the expander may add several provisional specialists at once, of which those failing to contribute are removed at step end. Whether $\hat{\rho}$ persists into $\mathcal{R}$ across queries is decided later under counterfactual evidence (§3.5). Panels A and B of Figure 2 summarize this per-query loop; learning the router policy $\pi_{\mathrm{GR},\theta}$ requires a gradient signal at the granularity of single structural edits, which §3.3 introduces.

3.3 SECPO: Single-Edit Counterfactual Advantage

SECPO inherits PPO's clipped surrogate with KL regularization (Schulman et al., 2017) and GRPO's critic-free, K-per-prompt group structure (Shao et al., 2024); only the advantage estimator is new. It replaces the group-mean baseline with a fresh re-execution from the same state in which the sampled

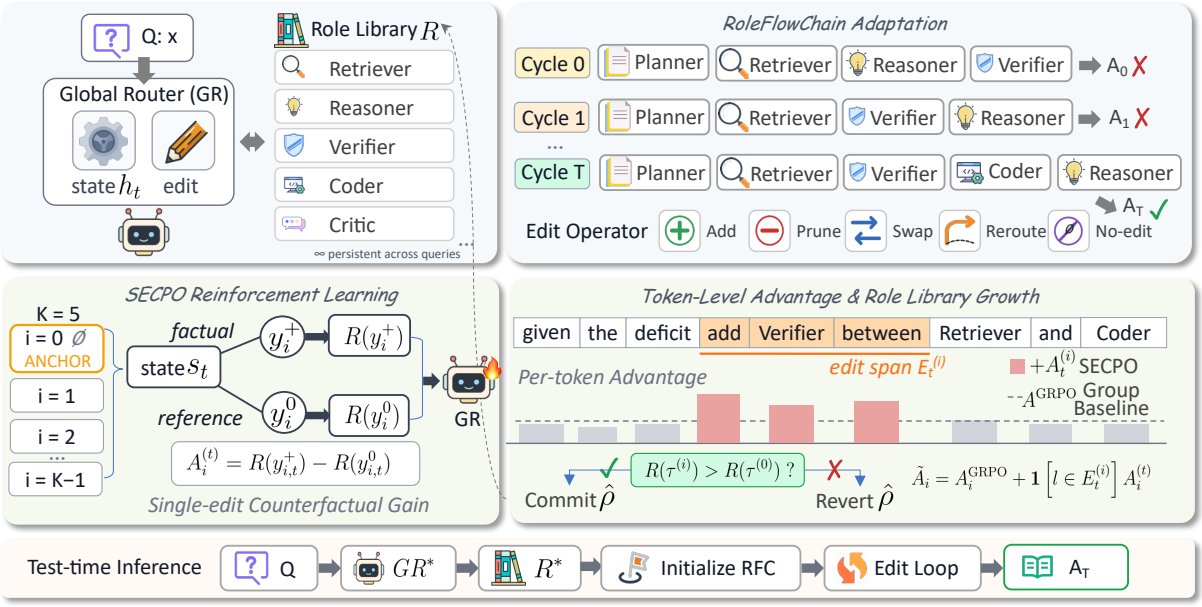


Figure 2: RoleFlow + SECPO overview. (A) Global Router GR over a learned Role Library $\mathcal{R}$. (B) One structural edit per cycle to the chain RFC_t. (C) SECPO compares each sampled edit against a forced no-edit anchor. (D) The advantage supplements GRPO at edit tokens; the library grows on positive counterfactual gain.

edit is suppressed. The three natural alternatives all fail in the structural-edit regime: vanilla PPO needs a learned value critic over a non-stationary state distribution that includes the chain, which is intractable for mutating DAGs; GRPO's group mean dilutes the structural signal across rollouts of different chain shapes; and the analytic counterfactual baseline of Foerster et al. (2018) (COMA) does not scale to our combinatorial edit space (five operators $\times$ $|\mathcal{R}|$ roles $\times$ insertion positions). To our knowledge, SECPO's re-execution-based estimator is the only critic-free counterfactual estimator that scales to mutating DAGs of arbitrary topology.

SECPO draws K rollouts $\{\tau^{(i)}\}_{i=0}^{K-1} \sim \pi_{\mathrm{GR},\theta_{\mathrm{old}}}(\cdot \mid x)$ from the same task, with one rollout (indexed $i=0$) forced to apply $\varnothing$ at every cycle; we call it the anchor. The remaining $K-1$ rollouts sample edits freely and run to termination, each on a private library overlay $\hat{\mathcal{R}}^{(i)}$ to which its own role insertions are written, so that sibling insertions are mutually invisible until the step-end commit.

At cycle t of rollout i, after the policy samples a candidate edit $\tilde{a}_t^{(i)}$, the chain is executed twice from the same state under matched stochasticity: with $\tilde{a}_t^{(i)}$ applied, producing $y_{i,t}^+ = (\mathrm{RFC}_t^{(i)} \oplus \tilde{a}_t^{(i)})(\sigma_t^{(i)}, x)$, and with the edit suppressed, producing $y_{i,t}^0 = \mathrm{RFC}_t^{(i)}(\sigma_t^{(i)}, x)$. Writing $R(\cdot)$ for the per-cycle reward, the single-edit advantage is

$$A_i^{(t)} = R(y_{i,t}^+) - R(y_{i,t}^0). \tag{2}$$

Sharing chain state, tool-call outcomes, and overlay snapshot between branches (common random numbers) makes the two branches differ only in $\tilde{a}_t^{(i)}$, so $A_i^{(t)}$ estimates the marginal effect of that edit at the current state. The anchor at $i=0$ does not enter Eq. 2 directly; it stabilizes the GRPO group-mean baseline used in §3.4 and supplies the trajectory-level reference for library curation (§3.5). Figure 1 situates the estimator against GRPO's freely-sampled group-mean baseline.

The SECPO loss takes PPO's clipped form, averaged over rollouts, cycles, and tokens of the router's plan utterance:

$$\mathcal{L}^{\mathrm{SECPO}}(\theta) = -\frac{1}{K}\sum_{i=0}^{K-1}\frac{1}{|o_{\mathrm{GR}}^{(i)}|}\sum_{\ell=1}^{|o_{\mathrm{GR}}^{(i)}|} \min\big(\rho_\ell \tilde{A}_\ell,\ \mathrm{clip}(\rho_\ell, 1-\epsilon, 1+\epsilon)\,\tilde{A}_\ell\big) + \beta\, D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{ref}}), \tag{3}$$

with per-token importance ratio $\rho_\ell(\theta) = \pi_\theta(o_\ell \mid o_{<\ell})/\pi_{\theta_{\mathrm{old}}}(o_\ell \mid o_{<\ell})$, clip $\epsilon = 0.2$, KL weight $\beta = 0.1$ (Appendix A.2), and $\tilde{A}_\ell$ the token-level advantage defined next. Beyond gradient flow, the advantage $A_i^{(t)}$ also gates the chain mutation: rollout i commits the candidate edit only when its advantage clears a threshold $\delta > 0$,

$$a_t^{(i)} = \begin{cases} \tilde{a}_t^{(i)}, & A_i^{(t)} > \delta \\ \varnothing, & \text{otherwise}, \end{cases} \tag{4}$$

so each rollout's chain stays free of edits whose

advantage is statistical noise while the gradient still consumes every counterfactual signal.

3.4 Token-Level Single-Edit Advantage

Broadcasting $A_i^{(t)}$ uniformly across every token of $o_{\mathrm{GR}}^{(i)}$ would dilute the single-edit attribution; we project it only onto the tokens that realize the edit. The router emits its edit operator as a textual span inside a longer plan utterance (e.g., `"...add a Verifier between Retriever and Reasoner..."`). Following GRPO, each token ℓ is assigned the broadcast group-relative trajectory advantage

$$A_\ell^{\mathrm{GRPO}} = \frac{R(\tau^{(i)}) - \mathrm{mean}_j\, R(\tau^{(j)})}{\mathrm{std}_j\, R(\tau^{(j)})}, \quad (5)$$

and SECPO adds a counterfactual supplement only on the token range $\mathcal{E}_t^{(i)}$ that encodes rollout i's cycle-t edit:

$$\tilde{A}_\ell = A_\ell^{\mathrm{GRPO}} + \mathbf{1}\left[\ell \in \mathcal{E}_t^{(i)}\right] A_i^{(t)}. \quad (6)$$

The supplement is sparse: only a handful of tokens per cycle carry the additive correction. Setting $A_i^{(t)} \equiv 0$ recovers GRPO, so SECPO strictly extends GRPO and reduces to it when no counterfactual signal is available. Figure 2 (Panel D) renders this as a token strip whose advantage spikes above the GRPO baseline only inside $\mathcal{E}_t^{(i)}$.

3.5 Counterfactual-Gated Library Growth

Library expansion turns role specifications into a learned, persistent artefact rather than a hand-curated input. When GR proposes a name outside the library, the LibraryExpander synthesizes a spec $\hat{\rho}$ into the calling rollout's private overlay $\hat{\mathcal{R}}^{(i)}$. At step end, $\hat{\rho}$ enters the shared $\mathcal{R}$ only if it appears in an edited rollout whose terminal return exceeds the anchor's:

$$\mathcal{R}_{t+1} = \mathcal{R}_t \cup \{\hat{\rho} \mid \exists\, i \geq 1 \colon \hat{\rho} \in \hat{\mathcal{R}}^{(i)},\ R(\tau^{(i)}) > R(\tau^{(0)})\}. \quad (7)$$

Roles failing this gate still consume gradient through Eq. 3 via their cycle-level counterfactual contrast; only the persistent specification is discarded. The single-edit advantage thus shapes each rollout's chain within a query, while the per-rollout terminal return governs which roles persist across queries: both tests reference the same anchor at different aggregation granularities.

Algorithm 1: SECPO Router Learning

Input: task x; group size K; max cycles $T_{\max}$.
Initialize K rollouts $\{\tau^{(k)}\}_{k=0}^{K-1}$ from x; force $\tau^{(0)}$ to apply $\varnothing$ at every cycle.
for $k = 0, \ldots, K-1$ *in parallel* **do**
  Attach overlay $\hat{\mathcal{R}}_k$; instantiate $(\mathrm{RFC}_0^{(k)}, \sigma_0^{(k)})$.
  for $t = 0, \ldots, T_{\max}-1$ **do**
    $y_{k,t}^0 \leftarrow \mathrm{RFC}_t^{(k)}(\sigma_t^{(k)}, x)$;
    $(h_t^{(k)}, s_t^{(k)}, d_t^{(k)}) \leftarrow V(y_{k,t}^0)$.
    if $h_t^{(k)} = 1$ **then break**
    if $k = 0$ **then**
      $a_t^{(k)} \leftarrow \varnothing$; $A_k^{(t)} \leftarrow 0$
    else
      $\tilde{a} \sim \pi_{\mathrm{GR}_\theta}(\cdot \mid \mathrm{RFC}_t^{(k)}, \sigma_t^{(k)}, d_t^{(k)})$;
      $y_{k,t}^+ \leftarrow (\mathrm{RFC}_t^{(k)} \oplus \tilde{a})(\sigma_t^{(k)}, x)$
      $A_k^{(t)} \leftarrow R(y_{k,t}^+) - R(y_{k,t}^0)$ `// Eq. 2`
      $a_t^{(k)} \leftarrow \mathbf{1}[A_k^{(t)} > \delta]\, \tilde{a}$ `// Eq. 4`
    $\mathrm{RFC}_{t+1}^{(k)} \leftarrow \mathrm{RFC}_t^{(k)} \oplus a_t^{(k)}$.
  $R^{(k)} \leftarrow R(\tau^{(k)})$.
For each rollout k, cycle t, and token ℓ:
  $\tilde{A}_\ell^{(k,t)} \leftarrow A_\ell^{\mathrm{GRPO},(k)} + \mathbf{1}[\ell \in \mathcal{E}_t^{(k)}]\, A_k^{(t)}$
  `// Eqs. 5, 6`
$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}^{\mathrm{SECPO}}(\theta; \tilde{A})$ `// Eq. 3`
$\mathcal{R} \leftarrow \mathcal{R} \cup \{\hat{\rho} : \exists k \geq 1,\ \hat{\rho} \in \hat{\mathcal{R}}_k,\ R^{(k)} > R^{(0)}\}$
  `// Eq. 7`
Output: updated θ and $\mathcal{R}$.

The test-time inference procedure is given in Appendix A.1.

4 Experiments

§4.2 reports six-benchmark main results and cost-efficiency; §4.4 isolates SECPO design choices and reports cross-backbone transfer. Per-domain breakdowns, qualitative case studies, and hyperparameter sweeps are in the appendices.

4.1 Experimental Setup

Datasets. HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022) for multi-hop QA (closed-book, 200 validation samples each); MATH-500 (Lightman et al., 2023) for math; SWE-bench-Verified (Jimenez et al., 2024) (100-instance stratified subset) for bug fixing; WebShop (Yao et al., 2022) (AgentBench v0.2 strict (Liu et al., 2024), 500 tasks, indices 0–499) for web interaction. RoleFlow trains SECPO on 200 examples per in-domain benchmark (HotpotQA, SWE-bench-Verified, WebShop) plus 100 MATH-500 examples; 2WikiMultiHopQA and MuSiQue are evaluated zero-shot from the HotpotQA checkpoint. Splits, licenses, and preprocessing scripts are in Appendix B.

Baselines. We organize baselines by the granularity at which collaboration structure receives a learning signal and whether structure adapts per query. Direct prompting baselines (GPT-4o-mini, Claude-3.5-Haiku, Gemini-2.5-Pro, Qwen2.5-7B-Instruct (Yang et al., 2024a), Qwen3-8B, Qwen-2.5-72B-AWQ) carry no multi-agent structure and share a 3-shot CoT template combining the FlashRAG (Jin et al., 2025b) closed-book instruction with the Search-R1 (Jin et al., 2025a) think-answer template (Appendix A.5). Static pipelines (Chain-of-Agents (Zhang et al., 2024b), SWE-agent (Yang et al., 2024b), Agentless (Xia et al., 2024)) hand-craft a structure that never updates across queries. Offline-search methods (AFlow (Zhang et al., 2024a), MAE, CoMAS) learn structure offline and freeze it at inference. Online adaptation splits into test-time heuristic orchestration without learning (GoA (Yun et al., 2026), LinearRAG), group-normalized online RL with a group-mean baseline (AgentFlow (Li et al., 2025), R1-Searcher (Song et al., 2025)), and our per-edit counterfactual advantage with a forced no-edit anchor. ChatDev and MaAS are discussed in Related Work but omitted for benchmark non-overlap.

Metrics. F1 for multi-hop QA, Accuracy for MATH-500 (SymPy (Meurer et al., 2017) with LLM-judge fallback), Resolved% for SWE-bench, Success Rate for WebShop. In Table 1, **bold** marks the best within the trained agentic cohort and underline the runner-up.

Implementation. SECPO trains the Global Router and Planner end-to-end with full-parameter updates under FSDP. Each group uses $K{=}5$ rollouts with one forced no-edit anchor, KL coefficient $\beta{=}0.1$, PPO clip ratio 0.2, and at most three structural cycles per task. Backbone choice, training budgets, optimizer schedule, compute envelope, role-library curation policy, and reproduced cohort hyperparameters are detailed in Appendix A.2.

4.2 Main Results

Within the trained agentic cohort (Table 1), RoleFlow leads all three multi-hop QA columns (+2.9 HotpotQA, +6.7 2Wiki, +4.6 MuSiQue over the second-best cohort method), is runner-up on MATH-500 (76.2, −0.8 from LinearRAG), and leads WebShop (+0.9 over AgentFlow). The exception is SWE-bench, where the single-edit chain does not address the unified-diff format failure mode and falls 5.6 points below AgentFlow. The 2Wiki and MuSiQue columns are zero-shot from the HotpotQA-trained checkpoint; MATH-500, SWE-bench, and WebShop use in-distribution checkpoints under the per-benchmark training budgets above.

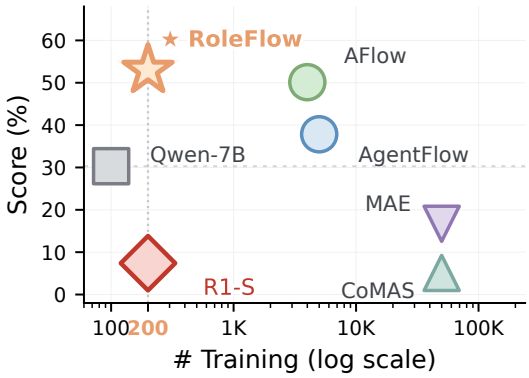
Analysis of the cohort gains. The three multi-hop QA gains share a common driver: the HotpotQA-trained Role Library transfers to 2Wiki and MuSiQue without retraining, and the Comparator and FactChecker specialists curated by the LibraryExpander on HotpotQA's comparison-bridge subset (Appendix C) handle the analogous bridge structure of the two out-of-distribution datasets. The +6.7 F1 gain on 2Wiki therefore validates the central design choice of a per-cycle structural edit space whose primitives are curated by counterfactual evidence: structural primitives compose across same-task domains without per-domain retraining. The WebShop result has a separate efficiency-driven character: SECPO's per-cycle validator gate terminates the rollout once the search-and-click trajectory is accepted, so RoleFlow's average rollout consumes roughly one-sixth of AgentFlow's tokens (Table 2) at slightly higher task success. Together, the out-of-distribution multi-hop result and the WebShop cost result validate two distinct mechanisms: cross-domain primitive reuse via the Role Library, and per-cycle adaptive termination via the SECPO anchor.

Cost-efficiency. Figure 3 and Table 2 report sample efficiency and the cost-accuracy trade-off (Inf Tok/Q and Inf Cost/Q at gpt-4o-mini list prices; Qwen-7B uses local vLLM at no API cost). RoleFlow sits at the top of the Pareto frontier. Three direct comparisons against representative cohort methods isolate this position: at CoA's per-query cost ($0.0014) RoleFlow gains +8.4 F1; at one-sixth of AgentFlow's token budget it gains +15.2 F1; against AFlow's training budget of 5,000 examples (25× ours) RoleFlow retains a +2.9 F1 lead while remaining within 3.3× the per-query inference cost. The sample-efficiency curve in Figure 3a additionally shows that RoleFlow's gain over the strongest cohort baseline emerges well before the 200-sample budget is reached, supporting the claim that SECPO's dense per-cycle counterfactual signal accelerates structural learning relative to terminal-reward-only group-relative baselines.

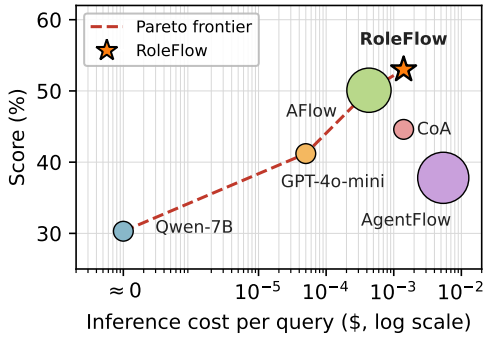
Table 1: Main results across six benchmarks. We **bold** and shade the best score within the trained agentic cohort and underline the runner-up; ↑ marks the gain of RoleFlow over the next-best cohort method per column. OOD denotes out-of-distribution evaluation.

Method	*Multi-hop QA*			*Reasoning*	*Coding*	*Web Interactive*
	HotpotQA	2Wiki^{OOD}	MuSiQue^{OOD}	MATH-500	SWE-bench	WebShop
GPT-4o-mini	41.2	20.1	24.0	75.8	8.3	55.4
Claude-3.5-Haiku	22.7	4.5	9.2	68.0	5.0	50.5
Gemini-2.5-Pro	15.0	8.1	4.8	27.8	15.0	40.5
Qwen-2.5-7B	30.3	18.5	11.6	75.6	5.3	58.2
Qwen3-8B	31.2	24.3	13.1	81.6	2.0	60.9
Qwen-2.5-72B-AWQ	44.8	21.7	19.4	81.8	0.0	62.4
CoA	44.6	25.9	17.7	73.6	9.8	52.3
GPTSwarm	48.5	26.7	21.1	73.8	8.5	49.7
AFlow	50.1	31.3	31.1	70.2	11.4	54.8
MAE	16.7	17.8	9.1	64.6	3.2	28.5
CoMAS	5.1	6.2	2.1	63.6	2.8	24.5
LinearRAG	29.8	13.6	24.9	**77.0**	7.4	48.2
AgentFlow	37.8	32.3	28.2	70.8	**14.6**	58.4
R1-Searcher[§]	7.4	4.2	2.9	22.4	2.1	28.7
RoleFlow (Ours)	**53.0** ↑+2.9	**39.0** ↑+6.7	**35.7** ↑+4.6	76.2	9.0	**59.3** ↑+0.9

[§] R1-Searcher: authors' GRPO recipe, 200-sample budget.



(a) Sample Efficiency.



(b) Cost-accuracy Pareto.

Figure 3: Sample-efficiency and cost-accuracy on HotpotQA. Bubble area in (b) is proportional to the square root of the number of training samples.

Table 2: Cost-efficiency at gpt-4o-mini list prices (2025-12).

Method	Score	Train Samples	Inf Tok/Q	Inf Cost/Q
Qwen-7B	30.3	0	267	$0
GPT-4o-mini	41.2	0	267	$0.00005
CoA	44.6	0	5,640	$0.0014
AFlow	50.1	5,000	1,288	$0.00043
AgentFlow	37.8	10,000	30,000	$0.0054
RoleFlow	**53.0**	**200**	**5,000**	**$0.0014**

4.3 Role Library Discovery

After training on HotpotQA, the Role Library $\mathcal{R}$ contains seven specialists: Planner and Validator (the structural anchors of every chain), OutputFormatter and InformationExtractor (formatting and recall), Reasoner (logical inference), and two roles synthesized by the LibraryExpander in response to validator deficit signals, Comparator (attribute matching) and FactChecker (consistency checking). Appendix C gives full specifications. The emergence of Comparator and FactChecker mirrors the comparison-bridge subset of HotpotQA on which the Validator fires deficit tags most frequently, and the same two roles drive the +6.7 F1 gain on 2WikiMultiHopQA out of distribution, where the bridge-question template is preserved. The library is therefore not a fixed set of hand-designed agents but a learned by-product of the SECPO training loop: roles enter when their counterfactual gain is posi-

tive and persist when re-used across queries. This emergent composition validates the design principle that structural primitives should be acquired from validator feedback rather than committed at design time.

4.4 Ablation Studies and Cross-Backbone Transfer

Component ablation. Table 3 disables each RoleFlow component in turn on the HotpotQA 200-val split: the Global Router GR (freezing the chain throughout the rollout), the no-edit Anchor (reverting to the vanilla GRPO group-mean baseline), the cross-query Role Library $\mathcal{R}$ (each query starts from the seed library), and the cycle budget $T_{\max}$ (a single pass). Removing GR yields the largest drop (-11.8 F1), confirming that the structural-edit policy contributes the largest single mechanism to RoleFlow's performance. The four components contribute proportionally to their causal scope: GR shapes every cycle, $\mathcal{R}$ (-5.4 F1) shapes every query through cross-query primitive sharing, $T_{\max}$ (-4.5 F1) enables multi-cycle revision under the validator's accept gate, and the Anchor (-3.6 F1) calibrates per-edit credit on top of the structural-editing policy. No single component dominates; the joint contribution validates the central design principle that structural adaptation requires both a learnable editing policy and counterfactual-anchored credit assignment, as argued in §1.

Cross-backbone transfer. Substituting Qwen3-8B for Qwen-2.5-7B as the GR backbone under the same 200-sample HotpotQA recipe yields $F_1 = 54.1$, $+1.1$ above the Qwen-2.5-7B configuration of Table 1. The structural-edit policy and SECPO objective therefore retain their gain on a newer Qwen generation without any change to the role library or the edit operator set. This monotonic improvement supports the deployment scenario where the underlying backbone is upgraded periodically while the structural-edit infrastructure (GR, $\mathcal{R}$, and the edit alphabet $\mathbb{E}$) persists: the SECPO advantage estimator is decoupled from the trained weight configuration and benefits from backbone improvements without re-engineering.

Table 3: Component ablation. Each row toggles one component off; the rest stay at the RoleFlow default.

Variant	GR	**Anchor**	$\mathcal{R}$	$T_{\max}$	**Score**
RoleFlow	✓	✓	✓	✓	**53.0**
– GR	✗	✓	✓	✓	41.2
– $\mathcal{R}$	✓	✓	✗	✓	47.6
– $T_{\max}$	✓	✓	✓	✗	48.5
– Anchor	✓	✗	✓	✓	49.4

5 Conclusion

We introduced SECPO, Single-Edit Counterfactual Policy Optimization, a strict extension of group-relative policy gradients that reserves one forced no-edit rollout per group as a fixed counterfactual anchor against which each candidate structural edit is differenced. SECPO addresses a specific failure mode of the standard GRPO recipe: when the action set itself mutates the chain that produces the rollout, the freely sampled group mean conflates edit effect with edit-type prior, and per-edit credit cannot be cleanly attributed. The estimator reduces to GRPO whenever no structural change occurs, so SECPO sits inside the GRPO family rather than replacing it.

We instantiated SECPO inside RoleFlow, a per-query structural-adaptation framework whose single-edit-per-cycle constraint is dictated by the counterfactual estimator's well-definedness. Across six benchmarks spanning multi-hop question answering, mathematical reasoning, software engineering, and interactive web environments, 200 in-domain examples per benchmark suffice for RoleFlow to lift out-of-distribution F1 on 2WikiMultiHopQA by 6.7 points over the strongest cohort baseline and to match the leading online-RL agent on WebShop at one-sixth of its per-query token cost. Beyond RoleFlow, the same construction applies to any agentic system whose policy edits the chain it runs on, suggesting that anchored counterfactual baselines are a general remedy whenever the action space is structural.

Limitations

Our study evaluates RoleFlow on six benchmarks spanning four task domains: multi-hop question answering, mathematical reasoning, software engineering, and web-shopping interaction. These four domains form a controlled cohort that isolates the contribution of structural adaptation across heterogeneous reasoning regimes, and the framework extends to long-horizon dialogue with persistent memory, multimodal and vision-language reasoning, and open-ended creative generation as natural directions in which the single-edit counterfactual

signal applies without modification to the underlying estimator.

We further fix the in-domain training budget at the cohort-standard scale (200 examples per benchmark, 100 for MATH-500) so that comparisons against recent self-evolving baselines remain apples-to-apples. Characterizing how structural learning and the discovered role library evolve under order-of-magnitude larger budgets is a direction we plan to pursue.

Ethics Statement

RoleFlow is evaluated on public research datasets (HotpotQA, 2WikiMultiHopQA, MuSiQue, MATH-500, SWE-bench, and WebShop) under their original licenses, with no human-subjects involvement. Code, learned role libraries, and adapter weights will be released under the MIT license to enable independent replication. RoleFlow inherits the standard safety considerations of the language models to which role execution dispatches, and we recommend human oversight in any deployment beyond the research setting. Large language model assistants were used for manuscript polishing and code refactoring during preparation; all research design, experiments, and analyses were performed by the authors.

References

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, and 9 others. 2022. Improving language models by retrieving from trillions of tokens. *Preprint*, arXiv:2112.04426.

Ignacio Cases, Clemens Rosenbaum, Matthew Riemer, Atticus Geiger, Tim Klinger, Alex Tamkin, Olivia Li, Sandhini Agarwal, Joshua D. Greene, Dan Jurafsky, Christopher Potts, and Lauri Karttunen. 2019. Recursive routing networks: Learning to compose modules for language understanding. pages 3631–3648. Association for Computational Linguistics.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *Preprint*, arXiv:2002.08909.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. *Preprint*, arXiv:2011.01060. 2WikiMultiHopQA benchmark.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

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

Jiajie Jin, Yutao Zhu, Guanting Dong, Yuyao Zhang, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, Zhicheng Dou, and Ji-Rong Wen. 2025b. FlashRAG: A modular toolkit for efficient retrieval-augmented generation research. In *Proceedings of the ACM Web Conference 2025*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. *Preprint*, arXiv:2210.02406.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Preprint*, arXiv:2005.11401.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. 2025. AgentFlow: In-the-flow agentic system optimization for effective planning and tool use. *Preprint*, arXiv:2510.05592. Flow-GRPO algorithm proposed therein.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050. MATH-500 subset proposed therein.

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 LLMs as agents. In *International Conference on Learning Representations (ICLR)*.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2023. Memory-assisted prompt editing to improve gpt-3 after deployment. *Preprint*, arXiv:2201.06009.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, and 8 others. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103.

Zhihong Pan, Kai Zhang, Yuze Zhao, and Yupeng Han. 2025. Route to reason: Adaptive routing for llm and reasoning strategy selection. *Preprint*, arXiv:2505.19435.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ChatDev: Communicative agents for software development. *Preprint*, arXiv:2307.07924.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. 2017. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *Preprint*, arXiv:1711.01239.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

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

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. *Preprint*, arXiv:2402.03300.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-Searcher: Incentivizing the search capability in LLMs via reinforcement learning. *Preprint*, arXiv:2503.05592.

ULab Team. 2025. Multi-agent evolve: LLM self-improve through co-evolution. *Preprint*, arXiv:2510.23595.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024. Mixture-of-agents enhances large language model capabilities. *Preprint*, arXiv:2406.04692.

Siyu Wang, Ruotian Lu, and 1 others. 2025a. AgentConductor: Topology evolution for multi-agent competition-level code generation.

Wenxiao Wang, Priyatham Kattakinda, and Soheil Feizi. 2025b. Maestro: Joint graph & config optimization for reliable ai agents. *Preprint*, arXiv:2509.04642.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. 2025. EvolveR: Self-evolving LLM agents through an experience-driven lifecycle. *Preprint*, arXiv:2510.16079.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2024. Agentless: Demystifying LLM-based software engineering agents. *arXiv preprint arXiv:2407.01489*.

Anonymous Xue and 1 others. 2025. CoMAS: Co-evolving multi-agent systems via interaction rewards. *Preprint*, arXiv:2510.08529.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024a. Qwen2.5 technical report.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024b. SWE-agent: Agent-computer interfaces enable automated software engineering. *Preprint*, arXiv:2405.15793.

Yingxuan Yang, Weinan Zhang, and 1 others. 2025. AgentNet: Decentralized evolutionary coordination for LLM-based multi-agent systems. *Preprint*, arXiv:2504.00587.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *Preprint*, arXiv:1809.09600.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. *Preprint*, arXiv:2210.03629.

Sukwon Yun, Jie Peng, Pingzhi Li, Wendong Fan, Jie Chen, James Zou, Guohao Li, and Tianlong Chen. 2026. Graph-of-agents: A graph-based framework for multi-agent LLM collaboration. *Preprint*, arXiv:2604.17148.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. 2025. AgentEvolver: Towards efficient self-evolving agent system. *Preprint*, arXiv:2511.10395.

Guibin Zhang, Kaijie Chen, and 1 others. 2025a. EvoFlow: Evolving diverse agentic workflows on the fly. *Preprint*, arXiv:2502.07373.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. 2025b. Multi-agent architecture search via agentic supernet. *Preprint*, arXiv:2502.04180.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. 2024a. AFlow: Automating agentic workflow generation. *Preprint*, arXiv:2410.10762.

Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. 2025c. Agentorchestra: A hierarchical multi-agent framework for general-purpose task solving. *Preprint*, arXiv:2506.12508.

Yichi Zhang, Fangzheng Xie, Shu Yang, and Chong Wu. 2025d. Meta-router: Bridging gold-standard and preference-based evaluations in large language model routing. *Preprint*, arXiv:2509.25535.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Arik. 2024b. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237.

A Implementation and Reproducibility Details

This section documents the implementation details, algorithms, hyperparameters, software stack, and reproducibility checklist for RoleFlow.

A.1 Test-Time Inference Procedure

Once GR_{θ^*} and the role library $\mathcal{R}^*$ are trained, test-time inference applies a lightweight, validator-gated edit loop without any further weight updates. Algorithm 2 formalises this procedure: starting from an initial chain RFC_0 selected by the router from the learned library, each cycle executes the chain, queries the validator for a deficit signal, and if the validator does not accept, the router emits exactly one structural edit that mutates the chain in place. The loop terminates upon validator acceptance or after $T_{\max}$ cycles. This is the inference-time counterpart of the SECPO training procedure in Algorithm 1 (§3.3), with the counterfactual re-execution and commit-threshold tests removed since the policy is fixed.

Algorithm 2: RoleFlow Test-Time Inference

Input: task x; trained GR_{θ^*}; learned $\mathcal{R}^*$.
Initialize RFC_0 from $\mathcal{R}^*$; $\sigma_0 \leftarrow \emptyset$.
for $t = 0, \ldots, T_{\max}-1$ **do**
  $(y_t, z_t, c_t) \leftarrow \mathrm{RFC}_t(\sigma_t, x)$;
  $(h_t, s_t, d_t) \leftarrow V(y_t, z_t)$.
  if $h_t = 1$ **then break**
  $a_t \sim \pi_{\mathrm{GR}_{\theta^*}}(\cdot \mid \mathrm{RFC}_t, \sigma_t, d_t)$;
  $\mathrm{RFC}_{t+1} \leftarrow \mathrm{RFC}_t \oplus a_t$.
Output: final answer y_t.

A.2 Detailed Hyperparameters

RoleFlow is trained on Qwen-2.5-7B-Instruct via full-parameter SECPO; Table 4 lists the SECPO objective settings used throughout the main experiments. All RoleFlow scores reported in Table 1 (main results) and Table 3 (component ablation) are means over three independent training seeds (40, 41, 42); we report means rather than maxima to give a conservative estimate of expected performance.

Table 4: SECPO objective hyperparameters. K, ϵ, β match Eq. 3; $T_{\max}$ matches the cycle budget of Eq. 1; δ_{pass} is the validator soft-score threshold above which $h_t = 1$ in Eq. 1.

Hyperparameter	Value
Group size K	5
Clip threshold ϵ	0.2
KL coefficient β	0.1
Max cycles per task $T_{\max}$	3
Validator pass threshold δ_{pass}	0.85
Learning rate	1×10^{-6}
Gradient accumulation steps	8
Max gradient norm	1.0

A.3 Compute Disclosure

All experiments use a single multi-GPU node with the configuration listed below.

Resource	Value
GPUs	4×NVIDIA H200 141GB SXM5
CPUs	2×AMD EPYC 9554 (128 vCPU)
RAM	2 TB
Storage	30 TB NVMe (shared)
Training GPU-hours	∼160
Inference GPU-hours	∼240

A.4 LLM-Judge Configuration

For metrics that admit multiple correct phrasings (HotpotQA F1 on long answers and MATH-500

when SymPy fails to verify symbolic equivalence), we supplement string-matching with a fixed GPT-4o-mini judge at temperature 0 with a three-way output schema of correct, partially correct, and incorrect.

Judge prompt template (HotpotQA / 2Wiki / MuSiQue).

```
You are an expert evaluator. Compare a predicted answer to a gold answer.
Gold: {gold}
Pred: {prediction}
Question: {question}
Output exactly one of: "correct", "partially_correct", "incorrect". A predicted answer is considered correct if it conveys the same factual claim, even with different wording, punctuation, or ordering. Reasoning chains may differ; only the final answer is graded.
```

The MATH-500 judge prompt asks for symbolic equivalence with the same three-way output schema.

A.5 Direct Prompting Baseline Protocol

All direct prompting baselines (GPT-4o-mini, Claude-3.5-Haiku, Gemini-2.5-Pro, Qwen-2.5-7B, Qwen3-8B, Qwen-2.5-72B-AWQ) use a single shared 3-shot chain-of-thought template across HotpotQA, 2Wiki, and MuSiQue, matching the prompt convention of AgentFlow (Li et al., 2025), Search-R1 (Jin et al., 2025a), and R1-Searcher (Song et al., 2025). Sampling is closed-book, temperature 0.0, top-p 1.0, max_new_tokens 128, seed 40.

System message.

```
You are a helpful assistant answering multi-hop factual questions using only your parametric knowledge (no external retrieval). For each question, briefly think through the required hops, then output ONLY the final answer in 1-5 words. Do not output explanations.
```

User template (3-shot CoT).

```
Question: What award did the writer of the screenplay for "Big Stone Gap" win in 2003?
Let me think step by step. Big Stone Gap was written by Adriana Trigiani, who won the Italian-American Heritage Award in 2003.
Answer: Italian-American Heritage Award

Question: Were Scott Derrickson and Ed Wood of the same nationality?
Let me think step by step. Scott Derrickson is American. Ed Wood was also American.
Answer: yes

Question: 2014 S/S is the debut album of a South Korean boy group; the group was formed by who?
Let me think step by step. 2014 S/S is the debut album of Winner; Winner was formed by YG Entertainment.
Answer: YG Entertainment

Question: {question}
Let me think step by step.
```

Stop strings. `"\nQuestion:"` and `"\n\n"` (prevents rambling beyond a single Answer line). The final answer is extracted with the regex `r"Answer:\s*(.+?)(?:\n|$)"` taking the last match.

Rationale for a shared template. The three few-shot examples are HotpotQA-derived bridge, comparison, and bridge-entity questions, but the template is applied verbatim to 2Wiki and MuSiQue as well. This matches the standard practice of Search-R1, AgentFlow, and R1-Searcher, all of which use a single shared template across multi-hop QA benchmarks. Per-dataset few-shot tuning would interact with the cross-domain comparison, so we hold the template constant.

A.6 Baseline Evaluation Protocols

We document the evaluation protocols used for each cohort baseline in Table 1. All multi-hop QA numbers are computed under the same closed-book 200-validation protocol used for RoleFlow.

AgentFlow (Li et al., 2025) is run from the authors' released Qwen-2.5-7B planner checkpoint with the auxiliary executor, verifier and generator modules served by GPT-4o-mini; we enable Wikipedia retrieval but disable live web search, matching the strongest reproducible closed-Google configuration.

R1-Searcher (Song et al., 2025) is retrained on the same 200-sample HotpotQA split used by RoleFlow, following the authors' GRPO recipe verbatim. The 200-sample budget is insufficient for the structured output protocol to converge, which is the

apples-to-apples data point reported in Table 1.

For AFlow (Zhang et al., 2024a) we use the authors' pre-discovered HotpotQA workflow with the closed-book benchmark loader; the same workflow is applied zero-shot to 2Wiki and MuSiQue. MAE (Team, 2025) and CoMAS (Xue et al., 2025) are evaluated from their released checkpoints or proxy backbones with the authors' default solver templates, providing an out-of-domain transfer reference rather than a domain-adapted comparison.

WebShop numbers follow the standard AgentBench v0.2 (Liu et al., 2024) strict 500-task protocol with the env-replay parser, with the role chain emitting the required search and click actions consumed by the official harness.

B Datasets and Benchmarks

This section catalogues the six benchmarks used in Table 1, summarising their domain, source, sample budgets, and split provenance.

B.1 Benchmark Catalog

Table 5 catalogues the six benchmarks used for RoleFlow evaluation. RoleFlow trains an in-domain SECPO checkpoint on each of HotpotQA, MATH-500, SWE-bench-Verified, and WebShop; 2WikiMultiHopQA and MuSiQue receive no in-domain training and are evaluated as out-of-distribution multi-hop QA benchmarks using the HotpotQA-trained checkpoint.

B.2 Per-Benchmark Statistics

Table 6 reports the per-benchmark distribution of input length, output length, and number of supporting documents (when applicable). These statistics inform the `max_tokens` settings in Table 4 and the role-prompt context budget.

Table 6: Per-benchmark length distributions (tokens unless noted).

Benchmark	Input tok		Output tok		Aux
	median	p95	median	p95	
HotpotQA	1,450	2,820	14	38	2 docs
2WikiMultiHopQA	1,710	3,260	12	32	2 docs
MuSiQue	1,960	3,540	18	51	2–4 docs
MATH-500	124	386	240	624	none
SWE-bench-Verified	2,540	5,200	720	2,100	1 diff
WebShop	510	1,080	22	56	1 action

B.3 Split Provenance

Test splits are sampled deterministically from each benchmark's official release using seed 40 for the slice index. For HotpotQA, 2Wiki, and MuSiQue we sample 200 examples from the official distractor-dev / dev / answerable-dev splits respectively, with the HotpotQA training corpus drawn from 200 disjoint examples in the train split. MATH-500 uses the full 500 problems as released by Lightman et al. (2023), with 100 disjoint train examples drawn from the MATH train split for in-domain SECPO. SWE-bench-Verified uses a stratified 100-instance test subset and 200 disjoint train examples drawn from the official SWE-bench-Verified train split. WebShop uses the 500-task AgentBench v0.2 (Liu et al., 2024) strict subset at indices 0–499, with 200 disjoint training tasks drawn from the WebShop training pool.

C Learned Role Library Specifications

This section documents the post-training role libraries for the two domains that exercise the LibraryExpander most heavily: HotpotQA (multi-hop QA) and SWE-bench-Lite (software bug fixing).

HotpotQA role library. Table 7 lists the seven post-training HotpotQA roles. The two roles added by the LibraryExpander, FactChecker and Comparator, target the verification and comparison deficits that the Validator surfaces on the comparison-bridge subset of HotpotQA.

SWE-bench role library. The SWE-bench library contains nine specialised roles for software bug fixing (Table 8). Because RoleFlow trains on HotpotQA only, these roles are synthesized at inference by the LibraryExpander in response to validator deficit tags. The co-existence of PatchGenerator and PatchRefiner is the most informative pattern: the trained GR inserts PatchRefiner once PatchGenerator has produced a first-pass patch, which is the inference-time analogue of the iterative-refinement hypothesis from §1.

D Chain Evolution Case Studies

This section presents detailed case studies showing how SECPO learns to modify role chains through counterfactual training.

D.1 Component Walkthroughs

To illustrate how each RoleFlow component contributes to overall behaviour, we walk through four qualitative case studies. Each case contrasts a controlled ablation against the full pipeline so as to isolate the marginal contribution of one design choice.

Table 5: Benchmark catalogue for the six-task suite.

Benchmark	Domain	Source	Train	Test	Metric
HotpotQA	Multi-hop QA	Yang et al. (2018)	200	200	F1
2WikiMultiHopQA	Multi-hop QA	Ho et al. (2020)	0	200	F1
MuSiQue	Multi-hop QA	Trivedi et al. (2022)	0	200	F1
MATH-500	Math	Lightman et al. (2023)	100	500	Acc
SWE-bench-Verified	Software	Jimenez et al. (2024)	200	100	Resolved%
WebShop	Web Interactive	Yao et al. (2022)	200	500 (AB v0.2)	SR

Table 7: HotpotQA role library specification.

Role	Capabilities
Planner	planning, coordination, edit_proposal
Validator	validation, quality_assessment
OutputFormatter	formatting, extraction
InformationExtractor	fact_finding, knowledge_recall
Reasoner	logical_inference, conclusion_drawing
Comparator	comparison, attribute_matching
FactChecker	verification, consistency_checking

Case 1: Role-driven decomposition. Figure 4 contrasts a single-LLM baseline with the RoleFlow execution on a representative query. When the prompt is dispatched directly to the backbone LLM, the output is disorganised and frequently incorrect. Under RoleFlow, the Global Router GR decomposes the task by assigning specialist roles in sequence; a Verifier role then audits intermediate outputs against domain heuristics, and an OutputFormatter normalises the final answer into the required schema. The role-level decomposition transforms an ill-formed direct response into a structured, validator-accepted answer, instantiating the per-query chain adaptation that drives the cross-domain gains in Table 1.

Case 2: Adaptive routing with GR. Figure 5 demonstrates why a per-query Global Router is essential. Without GR, the chain is committed at design time and cannot react to evidence accumulated during execution; when the validator rejects the cycle-0 answer due to insufficient intermediate reasoning, the static pipeline has no recourse and returns the rejected output. RoleFlow's GR, by contrast, observes the validator's deficit signal and dynamically inserts additional specialist roles, allowing cycle 1 to recover the correct answer. This is the qualitative analogue of the -11.8 F1 drop incurred when GR is removed from the ablation in Table 3.

Case 3: Learning GR and $\mathcal{R}$ jointly. Figure 6 illustrates the value of learning both GR and the Role Library $\mathcal{R}$ via SECPO. Prior to training, the chain relies on a single general-purpose Answerer role for every query, which is adequate for simple problems but ineffective for code-generation tasks. After SECPO training on HotpotQA, the trained GR learns to invoke the LibraryExpander on out-of-distribution coding queries and to compose the resulting specialists (Analyst, Coder, Debugger) adaptively, enabling the pipeline to solve a non-trivial coding problem that the pre-training pipeline cannot. This out-of-distribution synthesis is the same mechanism that produces the SWE-bench role library catalogued in Table 8.

Case 4: SECPO and cycle termination. Figure 7 shows how SECPO improves cycle-termination decisions through its counterfactual anchor. The forced no-edit rollout y^0 in each SECPO group provides a controlled reference against which the validator's hard-accept gate can be calibrated; as a result, the trained GR correctly identifies cycles in which further structural editing is unwarranted and terminates early, whereas the pre-SECPO pipeline conducts unnecessary additional cycles whose perturbations only introduce noise. The Anchor row in Table 3 quantifies the headline cost of removing this signal: -3.6 F1.

Table 8: SWE-bench role library specification.

Role	Capability
ProblemAnalyzer	problem_analysis
CodeLocalizer	code_localization
CodeReader	code_understanding
PatchGenerator	patch_generation
PatchValidator	patch_validation
PatchRefiner	patch_refinement
Planner	planning
Validator	validation
OutputFormatter	output_formatting

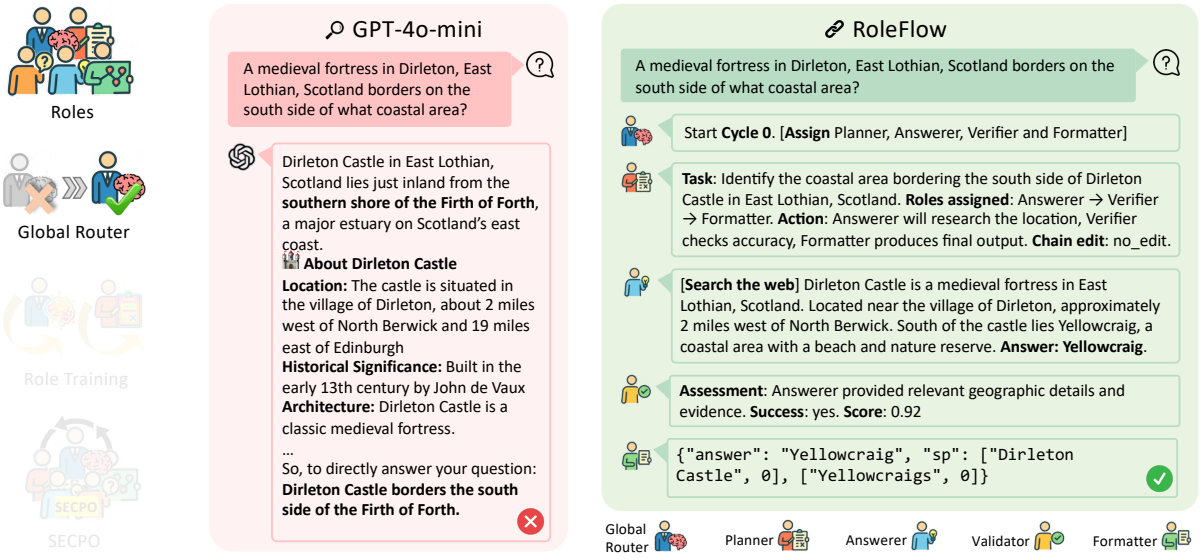


Figure 4: A direct LLM baseline (left) produces an unstructured, incorrect answer; the RoleFlow pipeline (right) decomposes the task across specialist roles, with the Verifier and OutputFormatter ensuring correctness and schema compliance.

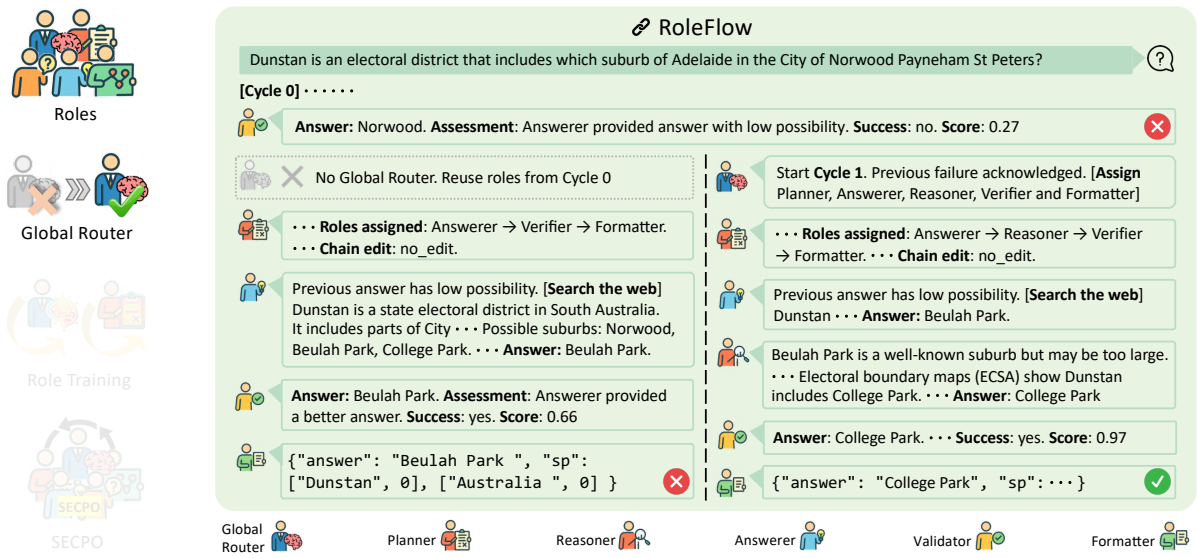


Figure 5: Without the Global Router (top), the chain is fixed and cannot recover from a rejected cycle. With GR (bottom), the chain dynamically incorporates additional roles in response to the validator's deficit signal, producing a correct answer in cycle 1.

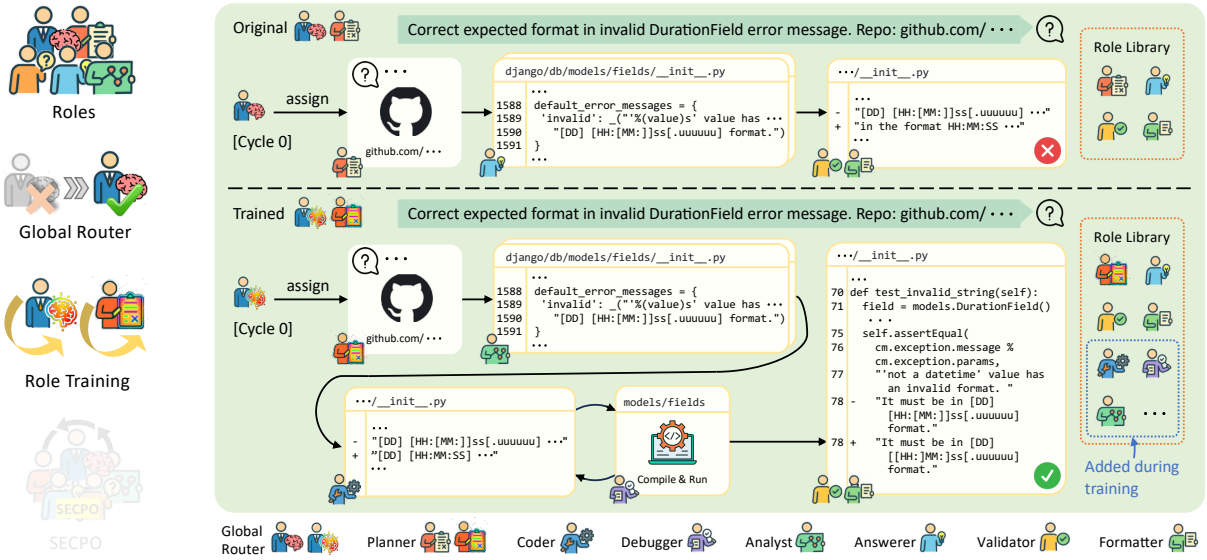


Figure 6: Before SECPO training (top), the library contains only a general-purpose Answerer and the pipeline fails on a coding task. After training (bottom), GR has populated $\mathcal{R}$ with domain-specific specialists (Analyst, Coder, Debugger) and learned to compose them, resolving the coding task.

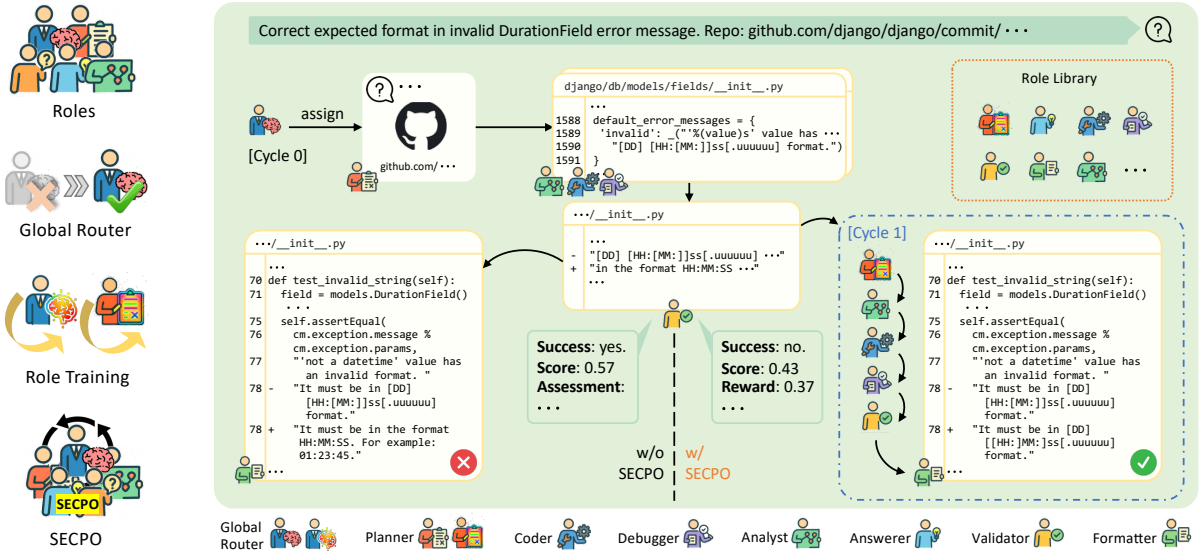


Figure 7: Without SECPO (top), the pipeline conducts additional cycles even when the current answer is already correct, perturbing the output. With SECPO (bottom), the counterfactual anchor enables calibrated cycle-termination decisions, producing a clean answer in fewer cycles.