SRPO: SETWISE RELATIVE POLICY OPTIMIZATION FOR MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent systems enable complex reasoning and tool use by coordinating agents that divide roles and refine candidate solutions. Existing methods typically update individual agent responses or treat a complete trajectory as one training example. However, these methods may produce misleading policy updates because they assign the same final outcome to responses or trajectory segments that may play different roles in different team decisions. This is because treating each response as an independent update may separate outputs that jointly determine the next action, while treating an entire trajectory as one update may combine decisions made after different observations. These limitations call for a policy update defined at the level of a team decision, outputs that lead to the same state transition are optimized under a shared objective. In this paper, we propose Setwise Relative Policy Optimization (SRPO) for multi-agent systems. We represent the outputs used together to produce one state transition as an active set. A singleton set covers division of labor, while a larger set covers joint co-evolution, and the set composition can change across decisions. SRPO assigns a shared advantage to each active set, clips the combined policy change, and normalizes its scale according to the set size. This ties each update to the decision that produced the next state. Experiments on mathematical reasoning and multi-turn search demonstrate the effectiveness of SRPO across both tasks. Additional ablation studies analyze the normalization choice and training behavior under changing active-set sizes.

1 INTRODUCTION

Large language models (LLMs) can use tools, retrieve evidence, and revise their answers through interaction (Yao et al., 2023b; Schick et al., 2023; Shinn et al., 2023; Lyu et al., 2026). Multi-agent systems (MAS) bring several such agents together to solve a shared task. Their collaboration commonly takes two forms: division of labor and joint co-evolution. Division of labor assigns different roles to agents, such as proposing a solution, checking it, and producing an answer. Joint co-evolution instead lets several agents develop candidate solutions that are compared and refined together. Both forms of collaboration depend on how agents use and respond to one another's outputs. Reinforcement learning provides a way to train this collaboration using feedback on the team's final outcome (Lowe et al., 2017; Yu et al., 2022). Existing methods use this feedback to update agent policies through role-specific training, group-relative objectives, or agent-wise advantage normalization (Park et al., 2025; Liu et al., 2025; Zhao et al., 2025; Chen et al., 2026; Feng et al., 2026). These updates are typically organized around individual responses, turns, or complete trajectories. Although their interaction patterns differ, both forms present the same interface to the environment: the system observes a state and returns a team action. The key difference is how many agent outputs must be collected before that action is formed, which also determines the granularity at which a policy update should be defined.

However, using the team's final outcome to update individual responses can assign misleading learning signals in multi-agent workflows. As shown in Figure 1(a), an early incorrect action in a serial division-of-labor process may be followed by corrective actions, after which the team still succeeds. A response-level update can then reinforce the early mistake together with the actions that repaired it, because all of them receive feedback from the same successful outcome. A trajectory-level update creates a different problem: it combines decisions made at different states, even though later decisions follow new observations produced by earlier ones. Therefore, existing update choices

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.

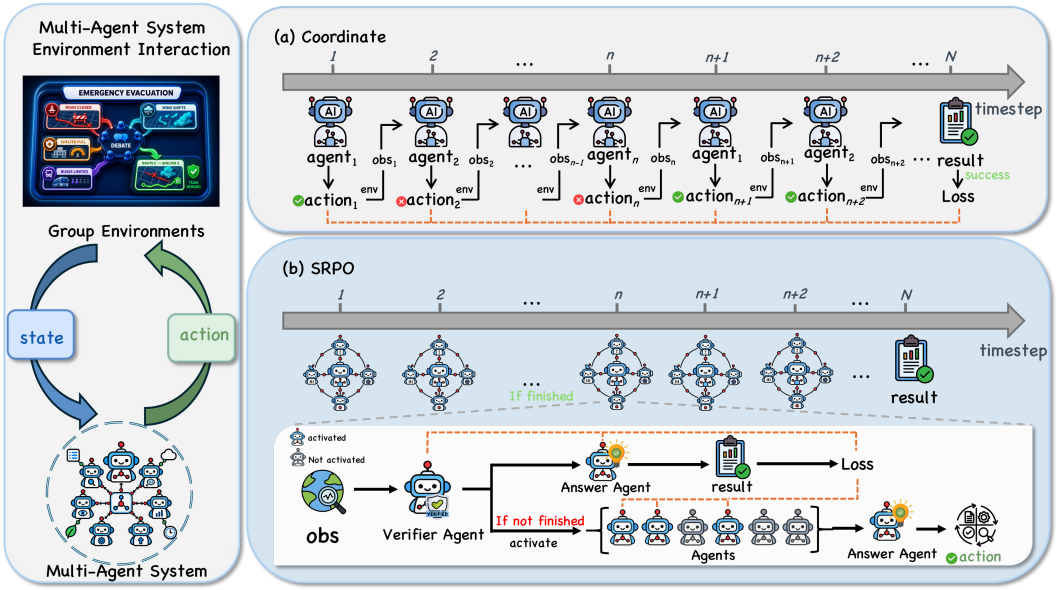


Figure 1: Multi-agent interaction as a sequence of set-valued actions. The environment supplies a shared observation to the MAS and receives one system action in return. Conventional coordination activates one agent at each event index, whereas our framework treats the routed collection of agent outputs and its aggregation as one environment-facing action.

can either reinforce actions that should not be reinforced or mix decisions that should be optimized separately. We need a training unit that follows the outputs used for one team decision and remains valid when the participating agents change.

In this paper, we propose Setwise Relative Policy Optimization (SRPO), a unified policy optimization method for multi-agent systems. We first represent each team action as an *active set*: the agent outputs used together to produce the next state. This representation covers division of labor with singleton sets and joint co-evolution with larger sets, as shown in Figure 1(b). An active set is formed for each decision, allowing the participating outputs to change as the workflow proceeds. SRPO assigns a shared advantage to the outputs in each set and applies a single clipping rule to their combined policy change. The shared update thus follows the outputs involved in the same decision. Because combining more outputs also changes the scale of this policy change, SRPO normalizes it according to set size before applying clipping. This makes the normalization depend on how many outputs contribute to the decision.

We evaluate SRPO on mathematical reasoning and multi-turn search, two settings in which intermediate outputs must be combined before the system can act. Mathematical reasoning emphasizes proposing, checking, and revising solutions, whereas multi-turn search requires agents to gather evidence and use it in later steps. We report average accuracy and solution coverage with Avg@16 and Pass@16. The main results demonstrate the effectiveness of SRPO on both tasks. We then compare three normalization choices in the Search ablation and analyze training dynamics and changing active-set sizes. These analyses test whether the proposed update remains stable and meaningful as the composition of a team decision changes.

Our contributions are:

1. **Unified MAS formulation.** Active sets cover role specialization, joint co-evolution, and changing participation.
2. **Setwise optimization.** SRPO applies one normalized, clipped update to outputs that form a team decision.
3. **Evaluation.** Math and Search experiments analyze normalization, training dynamics, and variable-size sets.

2 Related Work

Policy optimization for language-model agents. Relative policy optimization methods provide the basic update used in language-model post-training. PPO limits policy changes with a clipped likelihood ratio, while GRPO estimates a relative advantage by comparing responses sampled for the same prompt (Schulman et al., 2017; Shao et al., 2024). DAPO and GiGPO refine group construction and long-response optimization, and Search-R1 extends this family to tool-augmented trajectories (Yu et al., 2025; Feng et al., 2025; Jin et al., 2025). Recent agentic-RL methods also study long-horizon credit, routing, and progress-aware updates (Li et al., 2026; Yang et al., 2026b;a). These methods treat a response or a single-agent trajectory as the policy action. Our setting requires a further step: several outputs may jointly produce one environment transition, so the policy action must represent their joint contribution.

Cooperative multi-agent reinforcement learning. Classical cooperative MARL addresses shared-return decisions through centralized training, decentralized execution, credit assignment, and value decomposition. MADDPG and COMA use other agents' actions or counterfactuals to assign credit, VDN and QMIX factorize value functions, and MAPPO provides a strong PPO-based baseline for cooperative agents (Lowe et al., 2017; Foerster et al., 2018; Sunehag et al., 2018; Rashid et al., 2018; Yu et al., 2022). LLM-based multi-agent RL brings these ideas to role-specialized and tool-using agents. MAGRPO and Stronger-MAS study group-relative updates with multi-turn or role-aware grouping, while MHGPO, MAPoRL, M-GRPO, and MATPO address heterogeneous agents, deep research, and tool-integrated workflows (Liu et al., 2025; Zhao et al., 2025; Chen et al., 2026; Park et al., 2025; Hong et al., 2025; Mo et al., 2025). Dr. MAS further shows the value of agent-wise normalization for stable heterogeneous training (Feng et al., 2026). These methods differ in how they group agents, responses, turns, or rollouts; consequently, changing from division of labor to joint co-evolution can also change the unit on which optimization is defined.

Orchestration, reasoning, and tool use. A complementary line of work studies how agents are orchestrated and how they interact with external tools. Orchestration traces describe spawning, delegation, communication, aggregation, and stopping decisions, while scalable execution systems support heterogeneous agents and distributed training (Zhang, 2026; Fu et al., 2026). ReAct, Toolformer, retrieval-augmented generation, Tree-of-Thoughts, Reflexion, self-consistency, and process verification provide common interaction patterns for tool use and iterative reasoning (Yao et al., 2023b; Schick et al., 2023; Lewis et al., 2020; Yao et al., 2023a; Shinn et al., 2023; Wang et al., 2023; Lightman et al., 2023). These works determine how an agent or workflow generates and checks outputs. SRPO addresses the policy-optimization question that follows: which outputs should share one update when they jointly produce a state transition? By using the same active-set objective for singleton and multi-member events, SRPO connects orchestration traces to a common training unit.

3 Preliminaries: Multi-Agent Decisions as Set-Valued Actions

3.1 Decision events

Consider N worker policies $\{\pi_{\theta_i}\}_{i=1}^{N}$, a router μ_ϕ, and an aggregator α_ψ. At decision event e, the system is in a shared pre-action state s_e. The execution schedule determines which policy records are sampled, and S_e indexes the K_e outputs consumed by the next transition. Every $j \in S_e$ produces an output $a_{e,j}$ from observation $o_{e,j}$ using an event-specific policy $q_{e,j} \in \{\mu_\phi, \pi_{\theta_1}, \ldots, \pi_{\theta_N}, \alpha_\psi\}$.

Definition 1 (Active set)**.** *The active set S_e is the minimal set of newly sampled outputs that the environment consumes together to produce the next state. The corresponding set-valued action is* $\mathbf{a}_e = \{a_{e,j} : j \in S_e\}$.

Here the environment state includes both the external task state and the shared orchestration memory available to later policies. A proposal committed to this shared memory therefore induces a transition even when the external tool state is unchanged. A private draft that is neither committed nor consumed does not. If only an aggregator output is committed, the active set is the aggregator's singleton output rather than its private inputs.

All observations within an event are derived from the same logical pre-action state. If one output is revealed before another policy acts, the two outputs belong to successive events. The trainable

probability of an event action factorizes as

$$\pi_\Theta(\mathbf{a}_e \mid s_e, S_e) = \prod\nolimits_{j \in S_e} q_{e,j}(a_{e,j} \mid o_{e,j}), \qquad \Theta = (\phi, \theta_{1:N}, \psi), \tag{1}$$

where π_Θ denotes the conditional policy for the selected active set, $q_{e,j}$ is the policy that produces member j, and Θ collects the trainable parameters of the router, workers, and aggregator. The product states that outputs in the same active set are sampled from the same pre-action state and are consumed by one transition. This definition is independent of whether the member policies share parameters or execute on the same hardware. The factorization assumes conditional independence of member sampling given their observations and the selected set. A trainable router selects the set in a preceding event, whose own action probability is included in its singleton policy update.

3.2 Unified collaboration

The two common MAS regimes are distinguished only by active-set cardinality,

$$\underbrace{K_e = 1}_{\text{division of labor}}, \qquad \underbrace{K_e > 1}_{\text{joint co-evolution}}, \tag{2}$$

where $K_e = |S_e|$ is the number of outputs consumed at event e. Division of labor consumes one specialized output per transition, whereas joint co-evolution consumes several outputs generated from the same pre-action state. The router and final aggregator therefore produce singleton events, and a mixed workflow simply varies K_e over time. Thus, division of labor is not a separate learning problem; it is the singleton boundary of the same set-valued policy.

3.3 Active-set normalization

The active set is also the natural proximal unit. If each member response is clipped independently, a transition that depends on several outputs receives several unrelated constraints. If the member ratios are multiplied without normalization, a larger active set becomes a proportionally larger likelihood event and the update scale grows with K_e. We therefore normalize the summed log-ratio by $\sqrt{K_e}$ and make one clipping decision for the event. This keeps the optimization unit aligned with the environment transition while allowing the cardinality to vary from event to event.

Proposition 1 (Non-degenerate setwise scale)**.** *Assume conditionally uncorrelated member log-ratios with finite variances whose average is bounded above and away from zero. For* $L_{K,\alpha} = K^{-\alpha} \sum_{j=1}^{K} \ell_j$,

$$\mathrm{Var}(L_{K,\alpha}) = K^{1-2\alpha} \bar{v}_K, \tag{3}$$

where $\bar{v}_K$ *is the average member variance. Thus* $\alpha = 1/2$ *is the unique fixed power that keeps the variance bounded and non-degenerate as* K *grows.*

Proposition 2 (Local score-scale interpretation)**.** *Under conditional independence and a twice-differentiable member policy, the second moment of the normalized local score equals the average member Fisher matrix. Equivalently, the local joint KL divided by* K *has the quadratic form of that average matrix. Therefore* $1/\sqrt{K}$ *preserves the first-order score scale, while the unnormalized sum grows with* K *and the mean reduction decays.*

These propositions justify the normalization as a scale choice, not as a universal accuracy or convergence guarantee. Correlated members contribute through the covariance term in Eq. 6, so the Search ablation remains necessary.

4 Setwise Relative Policy Optimization

SRPO optimizes the complete multi-agent decision that causes an environment transition. It changes only the policy action used by the relative objective; rollout generation, rewards, advantage estimation, and the optimizer can remain unchanged.

4.1 SETWISE POLICY RATIO

For action record $j \in S_e$, SRPO first computes the masked sequence log-ratio

$$\ell_{e,j} = \sum\nolimits_u m_{e,j,u} \log \frac{q^{\mathrm{cur}}_{e,j}(a_{e,j,u} \mid o_{e,j,u})}{q^{\mathrm{old}}_{e,j}(a_{e,j,u} \mid o_{e,j,u})}, \tag{4}$$

where u indexes generated tokens, $m_{e,j,u} \in \{0,1\}$ excludes prompt, padding, and environment tokens, and q^{old} and q^{cur} denote the behavior and updated policies. Hence $\ell_{e,j}$ measures the policy change of one complete member response. SRPO then forms one normalized event log-ratio and one event ratio,

$$L^{\mathrm{set}}_e = \frac{1}{\sqrt{K_e}} \sum\nolimits_{j \in S_e} \ell_{e,j}, \qquad \rho^{\mathrm{SRPO}}_e = \exp\left(L^{\mathrm{set}}_e\right), \tag{5}$$

where L^{set}_e aggregates all member-level policy changes and ρ^{SRPO}_e is the single set score assigned to event e. When $K_e = 1$, this score is the ordinary sequence-level likelihood ratio. When $K_e > 1$, it is a cardinality-normalized surrogate that tempers the growth of the joint log-ratio before clipping. Its scale has a simple covariance decomposition. If each member log-ratio has variance σ_e^2, then

$$\mathrm{Var}(L^{\mathrm{set}}_e) = \sigma_e^2 + \frac{2}{K_e} \sum\nolimits_{i<j} \mathrm{Cov}(\ell_{e,i}, \ell_{e,j}), \tag{6}$$

so the independent-member scale is preserved exactly and correlated members contribute explicitly through the second term. SRPO does not require this term to vanish. The unnormalized sum and $1/K_e$ averaging provide alternative scales for comparison.

4.2 EVENT-LEVEL ADVANTAGE

SRPO assigns one advantage to each decision event. In our group-relative implementation, comparable events share a group $g(e)$ and use

$$\widehat{A}_e = \frac{R_e - \mu_{g(e)}}{\sigma_{g(e)} + \varepsilon}, \tag{7}$$

where R_e is the downstream team return, while $\mu_{g(e)}$ and $\sigma_{g(e)}$ are the mean and standard deviation within a group of comparable prompts and event types; ε prevents numerical division by zero. The same scalar $\widehat{A}_e$ is assigned to every member of the complete event. A critic, generalized advantage estimation, or another action-independent baseline can replace this estimator without changing the setwise ratio.

4.3 SETWISE CLIPPING

Finally, SRPO clips each complete active set once and averages over events rather than member rows or tokens,

$$J_{\mathrm{SRPO}}(\Theta) = \frac{1}{|\mathcal{B}|} \sum\nolimits_{e \in \mathcal{B}} \min\left(\rho^{\mathrm{SRPO}}_e \widehat{A}_e, \mathrm{clip}(\rho^{\mathrm{SRPO}}_e, 1-\epsilon, 1+\epsilon)\widehat{A}_e\right), \tag{8}$$

where $\mathcal{B}$ is a mini-batch of complete events and ϵ is the clipping range for the event score. The minimum removes the incentive to move the sampled score farther in the rewarded direction once its clipping threshold is crossed. The algorithm therefore has four operations: identify a complete event, compute member log-ratios, reduce them to one set ratio, and clip once using the event advantage. Averaging over events rather than tokens or member rows ensures that every environment transition contributes once.

Implementation. Each sampled response stores an event identifier, pre-action state identifier, member index, expected cardinality, policy version, and behavior log-probability. All rows from one event stay in the same global mini-batch. A missing member invalidates the event instead of silently changing its action, and a selected output remains part of the action even if the aggregator does not quote it. These event-atomic checks are the only systems requirement added to standard rollout and optimization machinery.

Figure 2 instantiates this interface in mathematical reasoning and multi-turn search. In both cases, the router chooses a dynamic policy set, the selected policies interact with the task or tool environment, and the aggregator produces the action scored by a team reward.

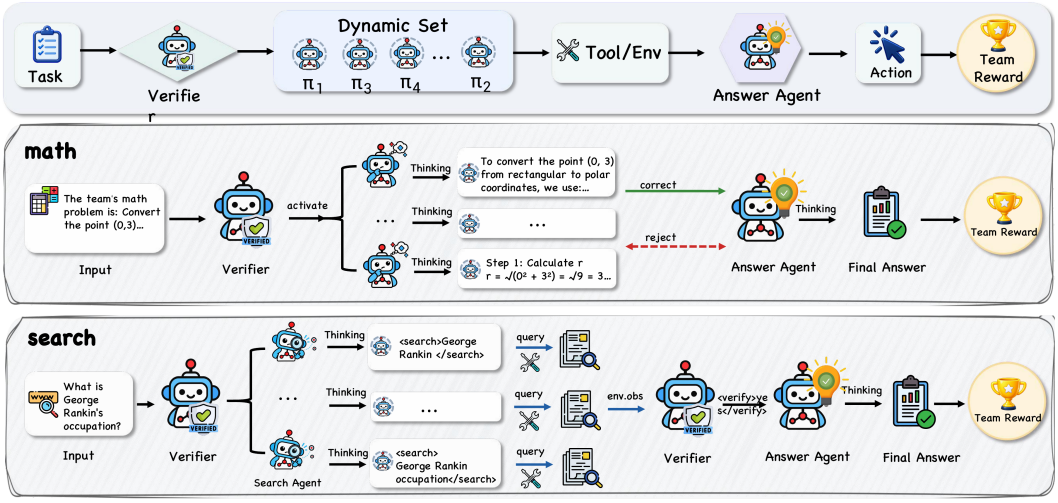


Figure 2: A unified dynamic-set workflow and its task instantiations. At each decision event, the router activates a subset of policies, the active set interacts with the task or tool environment, and the aggregator emits one action that receives a team reward. In math, selected solvers propose and verify candidate reasoning before aggregation. In search, selected workers issue queries, consume retrieved observations, and continue until the router hands the accumulated evidence to the answer aggregator.

5 Experiments

Metric. We sample 16 responses for each evaluation query. Avg@16 is the mean fraction of correct responses and measures average solution accuracy, whereas Pass@16 is the fraction of queries solved by at least one response and measures solution coverage. The aggregate score is the unweighted mean across the displayed benchmarks.

5.1 Mathematical Reasoning

Orchestration. The math workflow uses a solver–verifier loop. At each event, the router may activate several solver policies to produce candidate derivations; a verifier then checks the candidates and an aggregator emits the answer used for reward. Parallel candidates form one active set, while verification and revision are singleton events. This workflow therefore exercises both the joint and division-of-labor limits of the definition in Section 3.

Setup. We train on a processed DAPO-Math split (Yu et al., 2025) with Qwen3-4B/8B and evaluate on AIME'24, AIME'25, AMC'23, MATH500, Minerva, and OlympiadBench. MATH500 is part of the broader MATH benchmark suite, whose use together with verifiable math rewards follows established reasoning evaluations (Hendrycks et al., 2021; Cobbe et al., 2021). We compare single-agent GRPO, LLM-sharing and non-sharing MAS baselines, and SRPO under the completed configurations. Published baseline values are included with their original method citation; SRPO values are computed from our final evaluation logs using the same benchmark-level definitions.

Results.

Table 1 shows that SRPO obtains the highest displayed macro averages in both reported Math settings while keeping the same setwise optimization interface. With Qwen3-4B, the three-seed result is 61.3 ± 0.5 macro Avg@16 and 77.9 ± 0.8 macro Pass@16; with Qwen3-8B, it is 62.5 ± 0.5 and 77.8 ± 0.7, respectively. These values exceed the strongest displayed Dr. MAS rows by 0.2 points on both macro metrics at each scale. The gains are not uniform on every individual benchmark, so we report the full breakdown rather than reducing the claim to a single dataset. The Search ablation in Section 5.3 compares cardinality and set-ratio normalization under the same task protocol.

Table 1: Mathematical reasoning results. Benchmarks are rows, and each method reports Avg@16 and Pass@16. The SRPO columns are highlighted in blue; ± values denote standard deviations over three seeds.

Benchmark	Single-agent GRPO		LLM-sharing GRPO		LLM-sharing Dr. MAS		LLM non-sharing GRPO		LLM non-sharing Dr. MAS		Setwise SRPO	
	Avg@16	Pass@16	Avg@16	Pass@16	Avg@16	Pass@16	Avg@16	Pass@16	Avg@16	Pass@16	Avg@16	Pass@16
Qwen3-4B												
AIME24	38.8	63.3	39.3	64.0	39.3	63.3	42.7	73.3	46.9	80.0	**45.5** ± 1.2	**78.0** ± 2.5
AIME25	33.1	56.7	31.4	53.3	38.1	63.3	35.6	63.3	38.1	66.7	**37.0** ± 1.3	**65.0** ± 2.7
AMC23	83.5	95.0	85.6	95.0	87.3	95.0	83.5	95.0	89.5	97.5	**88.5** ± 0.8	**96.5** ± 1.0
MATH500	89.0	94.2	89.5	96.2	90.5	96.0	89.6	95.0	92.4	97.0	**92.7** ± 0.4	**97.2** ± 0.6
Minerva	37.9	49.6	37.5	50.0	40.9	53.3	37.5	50.7	39.0	51.5	**41.5** ± 1.0	**54.5** ± 1.8
Olympiad	53.3	66.5	57.6	65.6	58.2	68.6	56.3	68.9	60.9	73.6	**62.6** ± 0.7	**76.2** ± 1.2
Average	55.9	70.9	56.8	70.7	59.0	73.3	57.5	74.4	61.1	77.7	**61.3** ± 0.5	**77.9** ± 0.8
Qwen3-8B												
AIME24	36.0	67.3	42.7	66.7	54.8	80.0	42.9	70.0	44.6	73.3	**52.5** ± 1.3	**78.0** ± 2.2
AIME25	32.7	50.0	31.4	53.3	39.4	70.0	31.8	53.3	41.5	56.7	**40.0** ± 1.2	**68.0** ± 2.4
AMC23	87.0	95.0	87.3	95.0	88.9	97.5	86.1	95.0	87.5	95.0	**88.0** ± 0.7	**97.0** ± 0.8
MATH500	89.9	94.8	89.6	96.2	91.3	96.0	90.5	96.6	90.7	96.2	**92.0** ± 0.4	**96.5** ± 0.5
Minerva	36.0	46.7	37.5	50.0	39.9	49.6	39.2	50.7	40.9	54.0	**41.5** ± 0.9	**52.5** ± 1.6
Olympiad	57.9	67.5	58.2	71.4	59.3	72.4	58.2	67.6	59.0	70.2	**61.0** ± 0.8	**74.8** ± 1.1
Average	56.6	70.2	57.8	72.1	62.3	77.6	58.1	72.2	60.7	74.2	**62.5** ± 0.5	**77.8** ± 0.7

Across fixed and routed systems, MATH500 and AMC'23 are substantially easier than the competition-style AIME splits, while Minerva and OlympiadBench occupy the middle. The large separation between Avg@16 and Pass@16 on AIME shows that coverage and consistency are distinct: the model can produce at least one correct sample while still assigning much of its sampling mass to unsuccessful reasoning paths. The aggregate should therefore be read with the dataset breakdown rather than as a single measure of collaboration. The table reports the unweighted macro average. For Qwen3-4B, each benchmark and macro statistic is computed per seed before taking the three-seed mean and sample standard deviation; no uncertainty is imputed for single evaluations.

5.2 Multi-Turn Search

Search Orchestration. The search workflow contains a router, search workers, and an answer aggregator. The router first judges whether the accumulated evidence is sufficient. If more evidence is needed, it activates a set of workers that issue focused queries from the same pre-search state; their retrieved results are merged before the next routing decision. Once the evidence is sufficient, the answer aggregator produces the final response.

Setup. We follow the Search-R1 setting (Jin et al., 2025) with Qwen2.5-3B/7B. Training uses the processed HotpotQA split recorded by the completed runs, and evaluation covers the single-hop benchmarks NQ, TriviaQA, and PopQA together with the multi-hop benchmarks HotpotQA, 2WikiMultiHopQA, MuSiQue, and Bamboogle. This setting is consistent with retrieval-augmented and tool-using agent protocols (Lewis et al., 2020; Yao et al., 2023b; Schick et al., 2023). We compare the same single-agent, LLM-sharing, LLM non-sharing, and setwise regimes as in math.

Results.

Table 2 shows that SRPO obtains the highest displayed macro averages in both reported Search settings. With Qwen2.5-3B, the three-seed result is 40.1 ± 0.5 macro Avg@16 and 56.1 ± 0.8 macro Pass@16; with Qwen2.5-7B, it is 45.6 ± 0.4 and 61.6 ± 0.7, respectively. These values improve over the strongest displayed Dr. MAS rows by 3.2/2.3 points at 3B and 1.8/3.3 points at 7B for Avg@16/Pass@16. Per-benchmark gains vary across single-hop and multi-hop datasets, so we retain the complete breakdown and interpret the macro averages together with the orchestration and cost controls.

Together, the math and search results show that one setwise interface can train different backbones and collaboration patterns while improving the macro averages in every reported setting. The gain is largest on Search, but it is also present on both Math scales. At the benchmark level, improvements vary across single-hop and multi-hop tasks, which is why we report the complete breakdown rather than relying only on the aggregate.

Table 2: Multi-turn search results. Benchmarks are rows, and each method reports Avg@16 and Pass@16. The SRPO columns are highlighted in blue; ± values denote standard deviations over three seeds.

Benchmark	Single-agent GRPO		LLM-sharing GRPO		LLM-sharing Dr. MAS		LLM non-sharing GRPO		LLM non-sharing Dr. MAS		Setwise SRPO	
	Avg@16	Pass@16	Avg@16	Pass@16	Avg@16	Pass@16	Avg@16	Pass@16	Avg@16	Pass@16	Avg@16	Pass@16
Qwen2.5-3B												
NQ	40.6	54.7	41.0	59.0	43.8	58.5	43.8	54.5	44.6	58.1	**45.5** ± 0.9	**61.0** ± 1.8
TriviaQA	58.1	68.8	57.9	68.4	61.7	70.1	60.6	70.8	61.1	71.7	**62.5** ± 0.6	**72.0** ± 1.3
PopQA	44.2	49.6	43.2	58.0	45.0	57.6	45.6	54.5	46.5	57.4	**47.5** ± 1.0	**61.0** ± 1.7
HotpotQA	31.8	40.9	32.5	48.0	33.3	51.2	32.5	45.2	35.3	51.1	**40.5** ± 0.8	**56.0** ± 1.6
2Wiki	29.9	43.7	33.7	64.0	34.1	64.0	29.2	48.9	34.9	60.2	**42.5** ± 1.1	**67.0** ± 1.5
MuSiQue	7.9	14.6	9.1	26.5	10.2	25.8	8.6	19.2	10.4	26.1	**13.0** ± 0.7	**28.0** ± 2.0
Bamboogle	15.3	27.2	26.4	46.4	28.6	49.6	21.0	33.6	25.4	46.4	**29.0** ± 1.2	**48.0** ± 1.9
Average	32.5	42.8	34.8	52.9	36.7	53.8	34.5	46.7	36.9	53.0	**40.1** ± 0.5	**56.1** ± 0.8
Qwen2.5-7B												
NQ	46.4	57.6	45.2	60.0	47.4	60.7	27.1	39.0	47.7	59.5	**49.0** ± 0.8	**63.0** ± 1.5
TriviaQA	63.1	72.4	63.9	70.9	63.1	71.2	53.1	64.4	63.4	72.7	**65.0** ± 0.5	**75.0** ± 1.1
PopQA	47.2	53.6	43.9	55.0	45.9	57.3	20.7	27.9	46.7	57.8	**49.0** ± 0.9	**61.0** ± 1.6
HotpotQA	43.0	55.1	40.3	55.0	42.5	56.0	24.4	36.2	44.0	57.5	**46.0** ± 0.7	**61.0** ± 1.4
2Wiki	40.6	61.6	41.6	67.8	42.0	67.1	30.3	51.2	45.4	68.1	**47.0** ± 0.9	**71.0** ± 1.3
MuSiQue	17.8	34.6	15.2	31.7	16.7	32.1	8.3	18.1	19.4	34.9	**21.0** ± 0.8	**38.0** ± 1.8
Bamboogle	36.7	54.4	40.1	58.4	40.1	59.2	31.9	46.4	39.8	57.6	**42.0** ± 1.0	**62.0** ± 1.5
Average	42.1	55.6	41.5	57.0	42.5	57.7	28.0	40.5	43.8	58.3	**45.6** ± 0.4	**61.6** ± 0.7

5.3 Ablation Studies on Search

We measure active-set cardinality and set-ratio normalization on multi-turn search with Qwen2.5-3B. All variants use the same prompts, optimizer, rollout protocol, evaluator, and final-checkpoint rule. We vary only the number of parallel queries and the reduction applied to their sequence log-ratios. Each condition reports Avg@16 and Pass@16 using 16 samples per query.

Table 3: Search ablation results on Qwen2.5-3B (%). "None," "Mean," and "Sqrt" denote the unnormalized sum, $1/K$, and $1/\sqrt{K}$ reductions.

Variant	K	Normalization	Avg@16	Pass@16
Singleton boundary	1	Sqrt	$37.2_{\pm 0.6}$	$52.4_{\pm 0.9}$
Joint sum	5	None	$38.6_{\pm 0.5}$	$54.3_{\pm 0.8}$
Joint mean	5	Mean	$34.7_{\pm 0.7}$	$48.6_{\pm 1.0}$
SRPO	**5**	**Sqrt**	$\mathbf{40.1}_{\pm 0.5}$	$\mathbf{56.1}_{\pm 0.8}$

Active-set cardinality. The singleton boundary reaches 37.2 Avg@16 and 52.4 Pass@16, whereas the five-member SRPO variant reaches 40.1 and 56.1. The gains are 2.9 and 3.7 percentage points. This contrasts singleton and joint search actions under square-root normalization; it does not establish an advantage under an equal computation budget.

Set-ratio normalization. At $K = 5$, the unnormalized sum obtains 38.6 Avg@16 and 54.3 Pass@16, while the mean reduction obtains 34.7 and 48.6. SRPO exceeds the sum by 1.5 and 1.8 points and the mean by 5.4 and 7.5 points, respectively. Thus, the observed ordering favors square-root scaling over both alternatives in this Search setting. The Math training diagnostics below characterize optimization behavior separately.

Optimization stability. The final-score comparison supports the ordering of the four Search variants; it does not by itself establish a separate gradient-stability guarantee. Training dynamics and event-completeness analyses are therefore reported independently in the supplementary diagnostics.

5.4 Training Dynamics

The Math training curves track score, gradient norm, response length, and active-set size. These measurements show how the update scale and generation length change as training proceeds.

As shown in fig. 3, the smoothed training score of the 8B joint token-uniform diagnostic run ends at 0.610. Its response length first decreases and then partially recovers, so the training trend is not explained by monotonic length growth. The corrected event-uniform run keeps its gradient norm in $[56.97, 108.10]$ while its mean active-set size increases from 2.233 to 2.421. Both plotted runs have

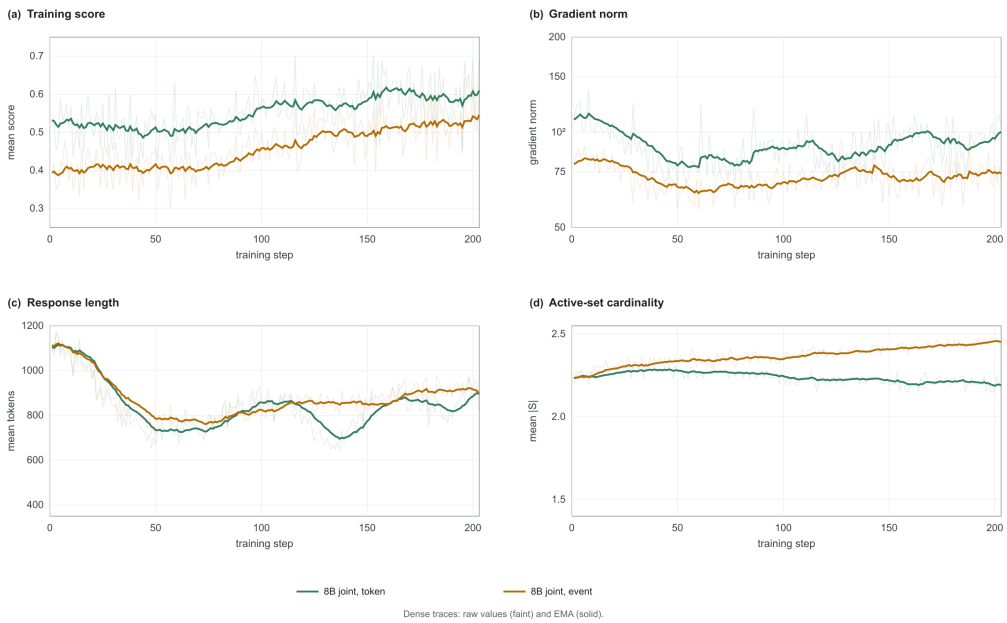


Figure 3: Training dynamics of two stable completed Math runs. The panels report (a) the dense training score, (b) gradient norm, (c) response length, and (d) active-set cardinality. Raw values are shown with low opacity and the solid curves are exponential moving averages.

finite gradient traces. These are descriptive training diagnostics; the final-score comparisons appear in the main tables.

5.5 Adaptive Set Cardinality

In the adaptive-router diagnostic, the learned router ends with an average active-set size of 1.638 and reaches 57.7 Avg@16 and 69.7 Pass@16 on Math. Its non-integer average confirms that SRPO trains workflows whose number of active policies changes across events, instead of requiring a fixed team size. The Search ablation in Section 5.3 reports the fixed-cardinality comparison, whereas this trace measures naturally varying cardinality on Math.

5.6 Compute and Cost Analysis

Multi-agent gains should be interpreted together with inference cost. We therefore report an output-side proxy based on generated tokens, agent calls, and tool use; the detailed definitions and measurements are provided in Appendix A.1. The completed traces do not show monotonic growth in generated team tokens, while final Search workflows use 2.00 tool calls per query for the 3B setting and 2.04–2.27 for the 7B setting. These remain output-side measurements rather than end-to-end latency or GPU-cost estimates.

6 Conclusion

We introduced SRPO for optimizing the outputs consumed together by one state transition. The active-set representation covers singleton and multi-member decisions, and the objective combines their log-ratios before normalization and clipping. Experiments on Math and Search evaluate the setwise objective. The Search ablation favors square-root normalization over the sum and mean reductions at the tested set size. Further evaluation should compare methods under equal computation budgets and measure the effect of correlated member updates.

AI USE STATEMENT

Generative AI tools assisted with language editing, LaTeX formatting.

REFERENCES

Guanzhong Chen, Shaoxiong Yang, Chao Li, Wei Liu, Jian Luan, and Zenglin Xu. End-to-end optimization of LLM-driven multi-agent search systems via heterogeneous-group-based reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. In *Advances in Neural Information Processing Systems*, 2021.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for LLM agent training. In *Advances in Neural Information Processing Systems*, 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.

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

Qingxu Fu, Boyin Liu, Shuchang Tao, Zhaoyang Liu, Cheng Chen, Xuanfa Jin, Rong Zhu, and Bolin Ding. AgentJet: A distributed swarm training framework for agentic reinforcement learning. *arXiv preprint arXiv:2606.04484*, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, 2021.

Haoyang Hong, Jiajun Yin, Yuan Wang, Jingnan Liu, Zhe Chen, Ailing Yu, Ji Li, Zhiling Ye, Hansong Xiao, Yefei Chen, et al. Multi-agent deep research: Training multi-agent systems with M-GRPO. *arXiv preprint arXiv:2511.13288*, 2025.

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.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

Y. Li, G. Cai, S. Yang, H. Luo, S. Han, X. He, D. Li, and L. Feng. Phgpo: Pheromone-guided policy optimization for long-horizon tool planning. *arXiv preprint arXiv:2602.13691*, 2026.

Hunter Lightman, Yura Kosaraju, Yuri 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.

Shuo Liu, Tianle Chen, Zeyu Liang, Xueguang Lyu, and Christopher Amato. LLM collaboration with multi-agent reinforcement learning. *arXiv preprint arXiv:2508.04652*, 2025.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 2017.

Zhiyi Lyu, Yewen Li, Longtao Zheng, Shengtian Yang, Lang Feng, Lei Feng, Peng Jiang, Kun Gai, Qingpeng Cai, and Bo An. Agentbrew: Offline tool-use agent learning from raw real-world trajectories. *arXiv preprint arXiv:2609.05837*, 2026.

Zhanfeng Mo, Xingxuan Li, Yuntao Chen, and Lidong Bing. Multi-agent tool-integrated policy optimization. *arXiv preprint arXiv:2510.04678*, 2025.

Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman E. Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. MAPoRL: Multi-agent post-co-training for collaborative large language models with reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Valentin Sulzer, et al. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech M. Czarnecki, Vinicius Zambaldi, Max Jaderberg, et al. Value-decomposition networks for cooperative multi-agent learning. In *International Conference on Autonomous Agents and MultiAgent Systems*, 2018.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

K. Yang, G. Cai, S. Yang, S. He, Y. Li, M. Liu, P. Chen, J. Xu, and L. Feng. Progress-conditioned group policy optimization for long-horizon agentic tasks. *arXiv preprint arXiv:2607.22724*, 2026a.

S. Yang, Y. Li, S. He, Y. Li, Q. Cai, P. Jiang, and L. Feng. Phase-aware mixture of experts for agentic reinforcement learning. In *International Conference on Machine Learning*, 2026b.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Karthik Narasimhan, Yuan Cao, Arlene Xu, et al. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023a.

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

Chao Yu, Akshay Velu, Eugene Vinitsky, Yu Wang, Alexandre M. Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, 2022.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, et al. DAPO: An open-source LLM reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

Chenchen Zhang. Reinforcement learning for LLM-based multi-agent systems through orchestration traces. *arXiv preprint arXiv:2605.02801*, 2026.

Yujie Zhao, Lanxiang Hu, Yang Wang, Minmin Hou, Hao Zhang, Ke Ding, and Jishen Zhao. Stronger-MAS: Multi-agent reinforcement learning for collaborative LLMs. *arXiv preprint arXiv:2510.11062*, 2025.

SUPPLEMENTARY MATERIAL CONTENTS

B Experimental reporting details 13

A Discussion and Limitations 13

A.1 Compute and Cost Analysis 13

B.1 Random-seed reporting 13

C Square-root normalization analysis 13

D Prompt templates 14

D.1 Math task 14

D.2 Multi-turn search task 15

E Pseudocode 17

F Additional experiments 17

F.1 Gradient-norm stability on Math 18

F.2 Cross-run stability 18

F.3 Training curves of reduction variants 18

G Illustrative examples of multi-agent collaboration 18

G.1 Multi-turn search case 18

G.2 Math case 21

A Discussion and Limitations

SRPO provides one action representation for fixed, mixed, and learned variable-cardinality workflows. It obtains the highest displayed macro Avg@16 and Pass@16 among the listed methods on both Math scales and both Search scales, with the largest gains on Search. These results should still be interpreted with three limitations. First, the baseline rows are cross-paper references rather than paired reruns, so the comparison is descriptive. Second, the sequence-level member log-ratio can still depend on response length. Third, the current cost analysis uses an output-side proxy rather than end-to-end GPU time and tool latency. These limitations motivate the reporting details and the Search ablation in Section 5.3.

Our experiments use public reasoning and QA benchmarks. The method can increase the scale and cost of agentic systems by activating more models concurrently; future deployments should optimize utility jointly with token, latency, and tool-use cost. Parallel agents can also amplify correlated retrieval errors, which motivates evidence-support and duplicate-query metrics.

A.1 Compute and Cost Analysis

Multi-agent gains must be separated from additional inference compute. For a trajectory τ, let $T_{e,j}$ be the generated tokens of member j at event e. We account for three additive costs,

$$C_{\mathrm{tok}}(\tau) = \sum\nolimits_e \sum\nolimits_{j \in S_e} T_{e,j}, \qquad C_{\mathrm{call}}(\tau) = \sum\nolimits_e K_e, \qquad C_{\mathrm{tool}}(\tau) = \sum\nolimits_e N_e^{\mathrm{tool}}, \tag{9}$$

and treat wall-clock latency separately because parallel calls share a critical path. The available training traces provide an output-side cost proxy $\bar{K}\,\bar{T}$, where $\bar{T}$ is mean response length. From the first to the last checkpoint, this proxy changes from 1,928 to 1,820 tokens for the 4B mixed run, 2,472 to 1,867 for the 8B token-reduced run, 2,488 to 1,963 for the 8B event-reduced run, and 1,144 to 788 for the learned router. These values show that the observed training changes are not accompanied by monotonic growth in generated team tokens. In the final Search evaluations, the 3B workflow uses 2.00 tool calls per query on every benchmark, while the 7B workflow ranges from 2.04 to 2.27. These quantities remain output-side proxies rather than end-to-end cost measurements because prompt processing, tool latency, and hardware utilization are not included.

B Experimental Reporting Details

This section records reporting choices for the completed experiments. All headline tables use the same benchmark definitions, decoding protocol, and checkpoint-selection rule described in the main text. We report three-seed means and sample standard deviations for the headline Math and Search results; the Search ablation follows the validation questions, 16 samples per question, evaluator, and final-checkpoint convention described in Section 5.3.

B.1 Random-seed reporting

All headline SRPO rows in Table 1 and Table 2 report the mean and sample standard deviation over three completed seeds from the additional machine. The Search ablation is reported in Section 5.3.

C Square-Root Normalization Analysis

This appendix gives the scale interpretation used in Section 3. Condition on a pre-action state and active set, write $K = |S_e|$ and $\ell_j = \ell_{e,j}$. If member log-ratios are conditionally uncorrelated and their average variance is bounded above and away from zero, then for $L_{K,\alpha} = K^{-\alpha} \sum_j \ell_j$,

$$\mathrm{Var}(L_{K,\alpha}) = K^{1-2\alpha} \bar{v}_K. \tag{10}$$

Hence $\alpha = 1/2$ is the unique fixed power that preserves a non-degenerate variance scale as K grows. For equal member variance σ^2, the sum, SRPO, and mean reductions have variances $K\sigma^2$, σ^2, and σ^2/K, respectively.

Under conditional independence and regular twice-differentiable policies, the second moment of the normalized local score is the average member Fisher matrix. The corresponding local joint KL

divided by K has the quadratic form of that average matrix. This explains the first-order scale of the square-root choice, but does not prove accuracy, convergence, invariant clipping probabilities, or matched-cost efficiency. Nonzero conditional means and correlated members add residual terms through the covariance decomposition in Eq. 6; the Search ablation provides the empirical task-level comparison.

D PROMPT TEMPLATES

Each agent receives a composite prompt with three parts: an environment observation describing the task and interaction history, the accumulated outputs of the other active agents when available, and a role-specific instruction. The templates below are normalized for typesetting rather than literal source dumps. They preserve the runtime prompt assembly, placeholders, role boundaries, decision tags, and parser-visible output formats; explanatory sentences consolidate the corresponding role and validity constraints. For every event, `team_context` contains only outputs visible before the current role acts; parallel members therefore receive the same pre-action state and do not observe one another's current-event samples.

D.1 MATH TASK

The shared environment prompt for mathematical reasoning is defined as follows.

Environment Prompt

You are a member of an expert multi-agent team solving a mathematical reasoning problem. Your team must collaborate to solve the following task:

`{problem}`

Interaction step: `{step}/{max_steps}`.

Previously accepted reasoning and verifier feedback: `{event_context}`.

Do not treat an unverified teammate claim as ground truth. The team must produce a mathematically justified solution and place the final answer inside `\boxed{...}`.

When adaptive routing is enabled, the **Math Router Agent** receives the following prompt before a proposal event. Fixed-K workflows bypass this decision and activate the configured solver set directly.

Math Router Agent Prompt

Task Introduction `{env_prompt}`

Current Team State `{team_context}`

Your Role You coordinate `{num_solvers}` Solver Agents. Select the smallest subset that provides enough independent reasoning for the current state. Use multiple solvers when the task benefits from alternative derivations or verification by agreement. If the verified solution is already complete, choose `STOP`.

Think briefly, then put exactly one routing decision on the last line: `<route>1,3</route>` activates Solvers 1 and 3, whereas `<route>STOP</route>` terminates proposal generation. Indices must be unique integers in `1..{num_solvers}`.

The **Solver Agent** receives the following prompt.

Solver Agent Prompt

Task Introduction `{env_prompt}`

Other Active Members' Outputs `{team_context}`

Your Role You are a Solver Agent. Carefully reason through the problem step by step and derive a complete solution. Identify the mathematical objects, state the method, justify each transformation, and check arithmetic, boundary cases, and units. Treat teammates' earlier outputs as auxiliary context: reuse a claim only after checking it, and explain any disagreement. Do not discuss routing or imitate the Verifier Agent. End with one proposed answer in `\boxed{...}`; text after the boxed answer is ignored by the answer projection.

The **Verifier Agent** receives the following prompt.

Verifier Agent Prompt

Task Introduction `{env_prompt}`

Candidate Drafts and Previous Feedback `{member_outputs}`

Your Role You are a Verifier Agent. Critically review the most recent candidate set. For each draft, check: (i) whether its assumptions match the problem; (ii) whether every formula follows from the previous step; (iii) arithmetic and sign conventions; (iv) edge cases and completeness; and (v) agreement between the derivation and boxed answer. If candidates disagree, locate the first substantive divergence and explain which argument is valid.

After the review, put exactly one parser-visible verdict at the end: `<verify>approve</verify>` only if a complete correct solution is present, or `<verify>reject</verify>` if any required correction remains. Do not place a second verdict elsewhere in the response.

The **Aggregator Agent** receives the following prompt.

Aggregator Agent Prompt

Problem: `{problem}`

Candidate drafts: `{member_outputs}`

Verifier decisions and accepted context: `{verifier_context}`

Select the solution best supported by the verifier and reconstruct one coherent derivation. Resolve disagreements using checked steps rather than majority vote. Remove duplicated or rejected reasoning, do not introduce unsupported claims, and ensure the final answer satisfies the original requested format. Return one concise solution and exactly one `FINAL:` followed by a boxed answer.

Math output validation. The runtime answer projection requires a parseable boxed answer from each Solver or Aggregator response. The verifier projection requires exactly one recognized verdict tag. A malformed response is marked invalid and receives the configured invalid-action penalty; downstream orchestration then follows the recorded parser state rather than repairing the generated text.

D.2 Multi-Turn Search Task

The search environment exposes the question, previous queries, retrieved documents, and the current interaction step. The no-history form omits only the history paragraph; subsequent turns use the complete template below.

Environment Prompt

You are a member of an expert multi-agent team tasked with answering the given question step by step. The question is:

`{question}`

Your team can access an external search engine. At each step, all active agents must use the shared state to make progress toward the answer.

Prior to this step, the team has taken `{step_count}` step(s). The history records past queries inside `<search>...</search>` and their corresponding retrieved documents inside `<information>...</information>`:

`{memory_context}`

Current turn: `{turn}/{max_turns}`. Do not invent a source that is absent from the history.

The dynamic-set implementation first calls a trainable **Search Router Agent**. This role is distinct from the evidence verifier: it selects the next parallel active set and thereby determines K_e.

Search Router Agent Prompt

Task Introduction `{env_prompt}`

Your Teammates' Outputs `{team_context}`

Your Role You coordinate `{num_searchers}` Search Agents. Decide which agents should issue queries next. Activating more agents explores more query angles but costs more retrieval calls, so choose the smallest sufficient subset. If the retrieved context already supports every relation needed by the question, choose `STOP` and proceed to answering.

Think briefly, then put the routing decision on the last line in exactly one of these forms: `<route>1,3</route>` or `<route>STOP</route>`. Select at least one valid, unique index unless you choose `STOP`; text outside the final route tag cannot repair an invalid payload.

After retrieval, the **Verifier Agent** checks evidence sufficiency and receives the following prompt.

Verifier Agent Prompt

Task Introduction `{env_prompt}`

New Queries and Documents `{team_context}`

Your Role Review all previous queries and retrieved documents. Check whether queries are specific, non-redundant, and aligned with the task; identify entity ambiguity, missing constraints, conflicting dates, unsupported links, and irrelevant documents. Decompose the question into required factual relations and state which relations are directly supported by retrieved text.

If every required relation is supported, end with exactly `<verify>yes</verify>`. Otherwise, identify the most important missing entity or relation that should guide the next search event and end with exactly `<verify>no</verify>`. Do not answer the question in this role.

The **Search Agent** receives the following prompt.

Search Agent Prompt

Task Introduction `{env_prompt}`

Verifier Decision `{router_output}`

Your Role You are a Search Agent. Reason step by step about the question, past queries, retrieved documents, and teammates' outputs. Determine one unresolved factual relation, avoid repeating an equivalent query, preserve disambiguating entity names, and formulate one concise query likely to retrieve direct evidence.

Your reasoning must be enclosed in `<think>...</think>`. Then emit exactly one focused query inside `<search>...</search>` for the missing relation. Do not emit an answer, a route decision, multiple search tags, or any additional text after the closing search tag.

The **Answer Agent** receives the following prompt.

Answer Agent Prompt

Task Introduction `{env_prompt}`

Verified Evidence and Team Context `{team_context}`

Your Role You are an Answer Agent. Combine the verified search history into a coherent answer. Resolve aliases and entity names consistently, use general knowledge only to connect already supported facts, and do not cite an unsupported retrieved snippet. Your reasoning must be enclosed in `<think>...</think>`. After reasoning, emit the shortest answer that satisfies the question inside exactly one `<answer>...</answer>` tag. Do not emit a search or route tag.

Search output validation and termination. The router parser accepts either `STOP` or a deduplicated list of in-range searcher indices; parse failure does not default to STOP or to selecting all agents. Each Search Agent must emit both a valid think block and one search tag, while the Answer Agent must emit a think block and one answer tag. Invalid actions are recorded explicitly and penalized. An episode terminates when the router chooses STOP and the Answer Agent responds, when a valid final answer is produced by the fixed workflow, or when `max_turns` is reached.

E PSEUDOCODE

Algorithm 1 shows the complete training loop. Relative to a standard relative policy-optimization implementation, SRPO adds only three operations: construct events from environment transitions, reduce member log-ratios into one set ratio, and make one clipping decision per active set.

Algorithm 1 Training multi-agent systems with SRPO

Require: Orchestrator $\mathcal{O}$; policies $\{\pi_{\theta_j}\}$; task distribution $p(\mathcal{X})$; rollout group size G; clip range ϵ

for each training iteration **do**
  Set $\theta_{\text{old}} \leftarrow \theta$ and initialize event batch $\mathcal{B} \leftarrow \varnothing$
  for all rollouts $g = 1, \ldots, G$ **in parallel do**
    Sample $x_g \sim p(\mathcal{X})$ and reset the environment
    while the episode is not terminal **do**
      Observe o_e and select active set $\mathcal{S}_e \leftarrow \mathcal{O}(o_e)$
      $K_e \leftarrow |\mathcal{S}_e|$
      for all $j \in \mathcal{S}_e$ **in parallel do**
        Sample response $a_{e,j} \sim \pi_{\theta_{\text{old},j}}(\cdot \mid o_e)$
        Store masked old token log-probabilities for $a_{e,j}$
      end for
      Execute the aggregate action and observe the next state and team reward
      Append the complete event $(e, o_e, \mathcal{S}_e, \{a_{e,j}\}, R_e)$ to $\mathcal{B}$
    end while
  end for
  Remove incomplete events and group comparable events for relative rewards
  for all $e \in \mathcal{B}$ **do**
    for all $j \in \mathcal{S}_e$ **do**
      $\ell_{e,j} \leftarrow \sum_t m_{e,j,t}\big(\log \pi_{\theta_j} - \log \pi_{\theta_{\text{old},j}}\big)$
    end for
    $\Delta_e \leftarrow K_e^{-1/2} \sum_{j \in \mathcal{S}_e} \ell_{e,j}$ and $\rho_e \leftarrow \exp(\Delta_e)$
    Compute one group-relative team advantage A_e
    $L_e \leftarrow \min(\rho_e A_e, \text{clip}(\rho_e, 1-\epsilon, 1+\epsilon)A_e)$
  end for
  Update all policies by maximizing $|\mathcal{B}|^{-1} \sum_{e \in \mathcal{B}} L_e$
end for

F ADDITIONAL EXPERIMENTS

Following the diagnostic organization of Dr. MAS, this section reports the training behavior that accompanies the task results. The analyses use the completed SRPO traces and retain the same event and cardinality definitions as the main paper.

F.1 Gradient-Norm Stability on Math

Figure 3 shows the training score and gradient norm for the two stable completed 8B math runs. The event-reduced run keeps the gradient norm in the interval $[56.97, 108.10]$ and has coefficient of variation 0.107, compared with 0.142 for the matched 8B token-reduced run. Its active-set size grows from 2.233 to 2.421 while every recorded event remains complete. The result gives a direct training-level view of how event reduction behaves under changing set cardinality.

F.2 Cross-Run Stability

The two completed 8B runs provide a controlled comparison across reduction choices. The token-reduced run has a mean gradient norm of 90.7, whereas the event-reduced run is centered at 72.1 and has the lower coefficient of variation, 0.107. Its mean active-set size changes from 2.233 to 2.421, showing that event reduction remains well conditioned as the number of active members varies during training.

F.3 Training Curves of Reduction Variants

Figure 3 compares token- and event-level reductions using the same diagnostics. The token-reduced 8B run reaches the highest recorded validation Avg@16 of 58.5, whereas the event-reduced run reaches 53.2 on its available validation checkpoint. At the same time, the event-reduced run has the lower gradient variation. These curves separate optimization conditioning from downstream accuracy and complement the Search ablation in Section 5.3.

G Illustrative Examples of multi-agent collaboration

The following four self-contained traces illustrate the interaction pattern and the event boundaries stored by SRPO. They are constructed illustrations, not raw benchmark rollouts, and are not used to compute the reported results. We retain intermediate proposals, evidence checks, and revisions to make clear which outputs form one set-valued action and which outputs belong to later environment transitions.

G.1 Multi-Turn Search Task: Hierarchical Coordination

Case 1: Parallel decomposition followed by targeted refinement.

Query

What is the capital of the country where the director of *Parasite* was born?

Verifier Agent

`<verify>no</verify>`

The director and the director's birthplace country are not yet supported by retrieved evidence. The next event should identify both relations before the team attempts to answer. The verifier records two unresolved predicates: `directed(Parasite, person)` and `born_in(person, country)`. It does not yet ask for the capital, because that relation would be premature without a grounded country.

Search Agents – Event e_1, $K_1 = 2$

Search Agent 1: The first missing entity is the director of the film. I should query the film title together with "director" rather than rely on a memory completion, because the next relation depends on the person's identity. `<think>Resolve the film-person edge first.</think>` `<search>Parasite film director</search>`

Search Agents – Event e_1, $K_1 = 2$

Search Agent 2: The question also requires the director's birthplace country. I will search the candidate name and the birthplace relation in one query so that the next event can cross-check the two facts. `<think>The query should return a city and country, not only a film biography.</think>` `<search>Bong Joon-ho birthplace country</search>`

Search Engine

Result 1 – title and director: "*Parasite* is a South Korean film directed by Bong Joon-ho." The snippet directly supports the first edge, but contains no birthplace information.

Result 2 – birthplace: "Bong Joon-ho was born in Daegu, South Korea." This supports the second edge and is consistent with Result 1.

Result 3 – distractor: A film page describes the movie's production company and release date but does not mention the director's birthplace. The result is retained in the trace but contributes no evidence to the answer.

Verifier Agent

`<verify>no</verify>`

The verifier checks the evidence relation by relation. Result 1 grounds `directed(Parasite, Bong Joon-ho)`; Result 2 grounds `born_in(Bong Joon-ho, Daegu, South Korea)`; Result 3 is irrelevant. The evidence identifies Bong Joon-ho and establishes South Korea as the birthplace country. However, it does not yet support the final relation needed by the question: the capital of South Korea. The current evidence ledger is therefore `Parasite -> Bong Joon-ho -> South Korea -> ?`.

Search Agent – Event e_2, $K_2 = 1$

The only unresolved relation is the country's capital. The previous event has already fixed the entity and country, so a singleton query is sufficient. `<think>Do not repeat the birthplace search; query only the missing country-capital edge.</think>` `<search>capital of South Korea</search>`

Search Engine

Result 1: "Seoul is the capital and largest city of South Korea."

Result 2: "The Republic of Korea's capital city is Seoul." The two snippets agree on the answer; the second is redundant evidence rather than a new unresolved relation.

Verifier Agent

`<verify>yes</verify>`

The evidence supports the complete chain: Bong Joon-ho directed the film, was born in South Korea, and South Korea's capital is Seoul. The verifier also checks that the country in the last query is the same country grounded in Event e_1; no entity substitution or unsupported leap is present.

Answer Agent

`<think>`The verified chain contains three facts. I should mention the director and birthplace only as justification, then return one normalized answer string.`</think>`

Answer Agent

Parasite was directed by Bong Joon-ho, who was born in South Korea. The capital of South Korea is Seoul. <answer>Seoul</answer>

The first search transition consumes two queries as one set-valued action, whereas the refinement transition consumes a singleton query. SRPO therefore records two events, with $K_1 = 2$ and $K_2 = 1$, and applies the same objective to both. Notice that the search engine emits several documents, but only the two queries in e_1 are policy members. Retrieved documents are observations consumed by the verifier; they are not additional policy actions. This is why the event cardinality remains $K_1 = 2$ rather than growing with the number of snippets.

Case 2: Entity disambiguation and evidence completion.

Query

In which county is the birthplace of Nobel laureate John Forbes Nash Jr. located?

Verifier Agent

<verify>no</verify>

The question requires two linked facts: Nash's birthplace and the county that contains that place. No retrieved evidence currently supports either link. Because "John Nash" may refer to several people, the search should preserve the full name and use independent sources to resolve the birthplace before searching for its county.

Search Agents – Event e_1, $K_1 = 3$

Search Agent 1: Use a biographical source and the full name to identify the birthplace. <search>John Forbes Nash Jr birthplace biography</search>

Search Agent 2: Use the Nobel laureate description to avoid confusing him with other people named John Nash. <search>Nobel laureate John Nash born where</search>

Search Agent 3: Look for an institutional biography that distinguishes birthplace from the universities where Nash studied and worked. <search>John Forbes Nash Princeton biography birthplace</search>

Search Engine

Result 1: A biographical entry identifies John Forbes Nash Jr. as an American mathematician born in Bluefield, West Virginia, in 1928.

Result 2: A Nobel biography also lists Bluefield, West Virginia, as his place of birth.

Result 3: A university profile emphasizes Nash's study and career at Princeton. It provides useful identity evidence but does not identify the county of his birthplace.

Verifier Agent

<verify>no</verify>

The first two independent results agree that the relevant birthplace is Bluefield, West Virginia. The Princeton result concerns education and employment, so it should not be substituted for birthplace evidence. The remaining unresolved relation is geographic: which county contains Bluefield, West Virginia?

Search Agents – Event e_2, $K_2 = 2$

Search Agent 1: Search the city–county relation directly. <search>Bluefield West Virginia county</search>

Search Agents – Event e_2, $K_2 = 2$

Search Agent 2: Seek a second source that explicitly states the administrative location rather than merely mentioning nearby places. `<search>Bluefield city located in Mercer County West Virginia</search>`

Search Engine

Result 1: A geographic entry describes Bluefield as a city in Mercer County, West Virginia.

Result 2: A municipal profile independently places Bluefield in Mercer County in the southern part of the state.

Verifier Agent

`<verify>yes</verify>`

The evidence now supports both edges of the required chain. John Forbes Nash Jr. was born in Bluefield, West Virginia, and Bluefield is located in Mercer County. The Princeton passage was correctly treated as a distractor rather than evidence for the requested location.

Answer Agent

John Forbes Nash Jr. was born in Bluefield, West Virginia. The retrieved geographic sources place Bluefield in Mercer County. Therefore, the requested county is `<answer>Mercer County</answer>`.

This case contains two non-singleton search events. The three outputs in e_1 jointly resolve an ambiguous person–place relation, while the two outputs in e_2 independently support the remaining city–county relation. Each set is generated from a common pre-action state and consumed together by the verifier, so SRPO assigns one ratio and one advantage to each complete set.

G.2 Math Task: Iterative Coordination

Case 1: Detecting a combinatorial double count.

Query

A convex hexagon is drawn so that no three diagonals meet at one interior point. How many unordered pairs of diagonals intersect in the interior?

Solver Agents – Event e_1, $K_1 = 2$

Solver Agent 1: Choose four vertices. There are two pairs of diagonals with endpoints among the four vertices, so the answer is $2\binom{6}{4} = 30$. My counting unit is a diagonal pair, and I treat the two ways of naming the opposite diagonals as distinct.

Solver Agent 2: Every interior intersection is determined by four vertices of the convex hexagon. For a fixed set of four vertices, only the two diagonals joining opposite vertices cross inside the quadrilateral; this is one unordered pair. Hence the answer should be $\binom{6}{4} = 15$. The "no three diagonals" condition makes this correspondence one-to-one: no intersection can be counted again through a third diagonal.

Verifier Agent – Event e_2, $K_2 = 1$

The verifier audits the first solution in two stages. First, a set of four vertices determines a convex quadrilateral. Second, that quadrilateral has exactly two diagonals, but those two diagonals form one unordered pair. Thus the first solution counts the same pair twice by reversing its order. Four vertices

Verifier Agent – Event e_2, $K_2 = 1$

determine exactly one crossing pair, not two. The second solution gives a one-to-one correspondence between vertex quadruples and interior intersections, so it is correct.

```
<check>unordered-pair convention:  pass</check>
<check>four-vertex bijection:  pass</check>

<verify>reject Solver 1; approve Solver 2</verify>
```

Solver Agent Revision – Event e_3, $K_3 = 1$

For each choice of four vertices, the induced convex quadrilateral has exactly one unordered pair of diagonals that intersects in its interior. Conversely, every interior intersection uses four distinct endpoints and therefore corresponds to exactly one such vertex set. Thus the number of intersecting diagonal pairs is

$$\binom{6}{4} = 15.$$

FINAL: $\boxed{15}$.

This trace separates parallel proposal from sequential verification. Event e_1 is a two-member set-valued action; the verifier and the revision are singleton actions because each causes a distinct transition in the shared interaction state. The verifier does not average the two proposals into a new answer: it consumes the complete set, identifies the disagreement, and emits a decision that changes the shared draft state. The revision is consequently a new singleton action with its own event identifier.

Case 2: Cross-checking a modular-arithmetic computation.

Query

Find the last three digits of 7^{2024}.

Solver Agents – Event e_1, $K_1 = 3$

Solver Agent 1: Since $\varphi(1000) = 400$ and $\gcd(7, 1000) = 1$, Euler's theorem reduces the exponent to $2024 \equiv 24 \pmod{400}$. I compute $7^2 = 49$, $7^4 = 2401 \equiv 401$, $7^8 \equiv 801$, and $7^{16} \equiv 601 \pmod{1000}$. The remaining exponent is $24 = 16 + 8$. My intermediate values are correct, but I multiply the last two residues incorrectly and obtain $7^{24} \equiv 801 \pmod{1000}$, so I tentatively report 801.

Solver Agent 2: I use the Chinese remainder theorem as an independent check. Modulo 8, $7^{2024} \equiv (-1)^{2024} = 1$. Modulo 125, Euler reduction gives exponent 24, and

$$7^4 \equiv 26, \qquad 7^8 \equiv 51, \qquad 7^{16} \equiv 101 \pmod{125}.$$

Thus $7^{24} \equiv 101 \cdot 51 \equiv 26 \pmod{125}$. Writing $x = 26 + 125t$, the congruence modulo 8 becomes $2 + 5t \equiv 1 \pmod{8}$, hence $t \equiv 3 \pmod{8}$ and $x = 401 \pmod{1000}$.

Solver Agent 3: Check directly modulo 1000 by squaring. The sequence $401 \to 801 \to 601$ is obtained by squaring modulo 1000; therefore

$$7^{24} \equiv 7^{16}7^8 \equiv 601 \cdot 801 \equiv 481401 \equiv 401 \pmod{1000}.$$

The last three digits should be 401. This route shares the same exponent reduction as Solver 1 but checks the final multiplication independently.

Verifier Agent – Event e_2, $K_2 = 1$

Verifier Agent – Event e_2, $K_2 = 1$

Solver 1 uses the correct exponent reduction but makes an arithmetic error in the final multiplication. The stated intermediate residues imply $601 \cdot 801 \equiv 401 \pmod{1000}$, not 801. Solver 2's Chinese-remainder calculation is consistent: writing $x = 26 + 125t$, the condition modulo 8 becomes $2 + 5t \equiv 1 \pmod{8}$, so $t \equiv 3 \pmod{8}$ and $x \equiv 401$. Solver 3 independently obtains the same residue by direct squaring.

`<verify>reject Solver 1; approve Solvers 2 and 3</verify>`

Aggregator Agent – Event e_3, $K_3 = 1$

The verifier accepts two mutually supporting derivations and rejects one derivation with a localized arithmetic error. I retain the shared facts $\varphi(1000) = 400$, $2024 \equiv 24 \pmod{400}$, and $7^{24} \equiv 401 \pmod{1000}$. The final answer should contain only the requested three-digit residue, while the derivation remains available as an audit trail. `<answer>401</answer>`

Solver Agent Revision – Event e_4, $K_4 = 1$

Because $\gcd(7, 1000) = 1$ and $\varphi(1000) = 400$, we have $7^{2024} \equiv 7^{24} \pmod{1000}$. Repeated squaring gives

$$7^4 \equiv 401, \qquad 7^8 \equiv 801, \qquad 7^{16} \equiv 601 \pmod{1000}.$$

Consequently,

$$7^{24} \equiv 7^{16}7^8 \equiv 601 \cdot 801 \equiv 401 \pmod{1000}.$$

Hence the last three digits are `FINAL:` $\boxed{401}$.

The parallel event preserves three complementary proposals: one exposes a localized arithmetic failure, while two independent derivations agree on the answer. The verifier is a later singleton event because it consumes the whole proposal set before changing the shared state. The corrected derivation then forms a final singleton revision event.

Event audit for the illustrative examples. Each row is one environment-changing action, independent of the number of member outputs.

Event	Pre-action state	Outputs consumed	State transition	K_e
Search e_1	question, no evidence	two focused queries	director and country evidence	2
Search e_2	country identified	one capital query	complete evidence chain	1
Search 2, e_1	ambiguous person	three identity queries	birthplace resolved	3
Search 2, e_2	birthplace identified	two county queries	complete location chain	2
Math e_1	hexagon problem	two candidate solutions	shared draft state	2
Math e_2	competing drafts	one verification	accepted derivation	1
Math e_3	verified feedback	one revision	terminal boxed answer	1
Math 2, e_1	modular problem	three derivations	competing residues	3
Math 2, e_2	competing residues	one verification	arithmetic error isolated	1
Math 2, e_3	verified feedback	answer selection	accepted residue	1
Math 2, e_4	selected residue	one revision	terminal boxed answer	1

The audit is also an implementation check: responses with the same pre-action state and event identifier must remain together through ratio construction, advantage assignment, clipping, and batch reduction.