WHEN DOES MULTI-AGENT RL IMPROVE LLM WORKFLOWS? WORKFLOW, SCALE, AND POLICY-SHARING TRADEOFFS

Anonymous authors
Paper under double-blind review

ABSTRACT

Outcome-reward reinforcement learning (RL) can train every role of a multi-agent LLM workflow end to end. Its payoff is hard to predict, because the reward does not credit individual roles and a shared policy trades pooled data against specialization. We ask when this training improves a workflow over its base model and single-agent RL, and what each role learns. We train three workflows (Eval-Opt, Voting, Orch-Workers) with GRPO on math and code at three Qwen3 scales, with one policy shared by all roles (Shared-Policy, SP) or one per role type (Isolated-Policy, IP). RL lifts every workflow above its untrained version. The solving roles, which write candidate solutions, end no better than single-agent RL, so a workflow exceeds single-agent RL only through the structure above them, whose gain is set by the simple rule its decision role learns. Eval-Opt gains most because its revision loop gives truncated first attempts another turn, extending the token budget. IP learns faster and peaks higher than SP. Under SP, averaging each output's loss over its tokens lets a decision role with much shorter outputs take well over its uniform share of the shared gradient, and the solving roles learn slowly. Under IP, the adapter of a role run three times per episode can become unstable after the peak, and only that role's policy-ratio χ^2 and perplexity rise. A stopping rule on training success fires only after the peak but can return a checkpoint near it, and SP runs decline little.

1 INTRODUCTION

Multi-agent LLM workflows improve on a single model call by splitting generation, evaluation, aggregation, or delegation across several calls (Yang et al., 2026b; Wu et al., 2023; Madaan et al., 2023; Wang et al., 2023; Schluntz & Zhang, 2024). Their roles are usually played by a model trained to answer on its own, so a generator never adapts to the evaluator that checks it. Reinforcement learning with verifiable rewards (RLVR) improves LLM reasoning on mathematics and code (Shao et al., 2024; Guo et al., 2025; Yu et al., 2026; Zhao et al., 2025a) and trains multi-step agents from an end-of-episode outcome signal (Jin et al., 2025; Wei et al., 2025; Wang et al., 2025), so the same reward can train every role of a workflow end to end.

Whether this training pays off is hard to predict. The reward says whether the final answer is correct, not which role made it so, yet every role receives it (Fig. 1(a)). In cooperative multi-agent RL, sharing parameters improves sample efficiency but can interfere with learning distinct roles, and its benefit depends on the environment (Christianos et al., 2021). Multi-agent extensions of Group Relative Policy Optimization (GRPO) (Shao et al., 2024) compare shared and per-role policies on LLM workflows with one agent per role (AT-GRPO (Zhao et al., 2025c), Dr. MAS (Feng et al., 2026); §2), and in AT-GRPO the better choice changes with task and model size. We ask a controlled empirical question: **when does RL training improve a multi-agent workflow over its base model, and what does each role learn?**

We train three workflows (Eval-Opt, Voting, Orch-Workers) at three model scales (0.6B, 1.7B, 4B) on two tasks (math and code) under two policy-sharing strategies, one policy for all roles (Shared-Policy, SP) or one per role type (Isolated-Policy, IP). *Solving roles* write candidate solutions and a *decision role* sets the final answer; Voting and Orch-Workers run three instances of one solving

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.

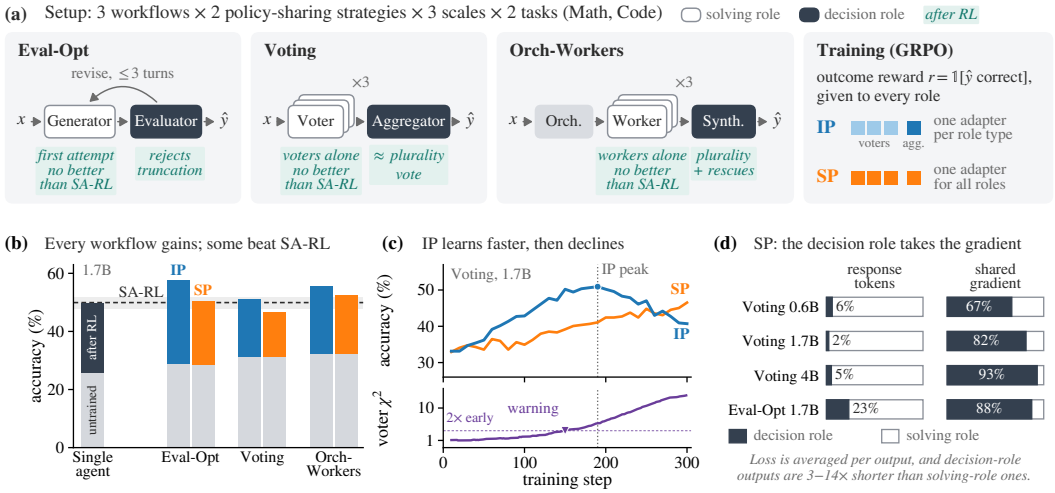

Figure 1: **Setup and main findings.** (a) IP: one adapter per role type (the three voters or workers share one); SP: one for all roles. Every role receives the outcome reward; tags: behavior after RL. (b) Accuracy at 1.7B before (gray) and after RL; SA-RL: single-agent RL; band: ties with SA-RL. (c) Voting at 1.7B: IP and SP validation accuracy, and the IP voter adapter's policy-ratio χ^2 over its early median, marked where it first exceeds twice that median. (d) SP decision role's share of response tokens and of the shared gradient.

role, the *repeated role* (Fig. 1(a)). Each of the 36 *cells* is compared with its base workflow and with single-agent RL (SA-RL), which separates the gain from training the workflow from the gain RL already gives a single model.

We answer the question in four parts.

- **Which workflows gain from RL, and where the gain comes from.** Every workflow gains over its base workflow (Fig. 1(b)). Under the broadcast reward the solving roles end no better than SA-RL, so only the structure above them can lift a workflow past SA-RL. Under IP the Eval-Opt loop and the Orch-Workers synthesizer do so, while the Voting aggregator stays close to a plurality vote (Fig. 1(a); §4.1).
- **Why Eval-Opt gains most.** Its evaluator rejects truncated attempts, so the revision loop extends their token budget; it rarely recovers a finished wrong attempt (§4.2).
- **Why isolated policies beat a shared policy.** IP learns faster than SP in nearly every matched pair and peaks higher in most (Fig. 1(c)). Because each output's loss is averaged over its own tokens, under SP a decision role with much shorter outputs takes well over its uniform share of the shared gradient, and the solving roles learn slowly (Fig. 1(d); §4.3).
- **Why isolated policies decline after their peak, and how to keep the peak.** The repeated role's adapter can become unstable, and only that role's χ^2 and perplexity rise (Fig. 1(c)). Training success falls with validation, so a stopping rule on it fires only after the peak but can return a near-peak checkpoint without a validation set; no SP or SA-RL run declines comparably (§4.4).

IP thus learns faster and peaks higher but can decline after its peak, while SP declines little but trains its solving roles slowly.

2 Related Work

Multi-agent RL training of LLM workflows. Several works extend GRPO to multi-agent LLM systems with structure-specific objectives: agent- and turn-wise grouping in AT-GRPO (Zhao et al., 2025c), centralized group-relative advantages in MAGRPO (Liu et al., 2025), alignment of variable sub-agent invocations in M-GRPO (Hong et al., 2025), heterogeneous group advantages in MHGPO (Chen et al., 2025), and per-agent LoRA adapters in MARFT (Liao et al., 2025). AT-GRPO

and Dr. MAS (Feng et al., 2026) compare shared and role-specialized policies, long studied in cooperative multi-agent RL (Terry et al., 2020; Christianos et al., 2021), and LLM systems use a separate model per agent (Motwani et al., 2025; Subramaniam et al., 2025) or shared parameters (Wan et al., 2025). Self-Refine (Madaan et al., 2023) and self-consistency (Wang et al., 2023) are inference-time counterparts of Eval-Opt and Voting. We use the agent-wise advantage normalization of Dr. MAS, which AT-GRPO further splits by turn (Appendix E).

Parallel same-role agents and per-agent sample counts. Some works train only the aggregator: AggLM (Zhao et al., 2025b) trains it over solutions from a generator kept fixed in its main experiments, and Kimi K2.5 (Kimi Team et al., 2026) freezes its sub-agents to avoid ambiguous credit assignment and training instability. Others train repeated instances jointly, with one adapter per agent in MAPoRL (Park et al., 2025), one shared model in WideSeek-R1 (Xu et al., 2026) and Recursive Agent Optimization (RAO) (Gandhi et al., 2026), and separate leader and sub-agent models in M-GRPO. Several correct for unequal sample counts: Dr. MAS normalizes advantages per agent and M-GRPO duplicates or drops sub-agent trajectories to a fixed count. RAO and WideSeek-R1 reweight trajectories so sub-agents do not dominate the gradient, and SRPO (Yang et al., 2026a) divides the summed log-ratio of parallel outputs by the square root of their number. These corrections reweight samples within one gradient; we study how many optimizer steps each role's adapter takes. With mini-batches fixed in sequences, as in verl (Sheng et al., 2024), a role with more samples takes more optimizer steps per training step under per-role batches (§4.4). Scaling each mini-batch to its role's sample count fixes that number (Appendix E).

3 Experimental Setup

Each of the 36 cells is trained once with a fixed seed (Fig. 1(a)). The controls are the base model alone, each workflow with the base model in every role (the *base workflow*, or untrained workflow), and single-agent RL (SA-RL) at each task and scale (Appendix B.3). Training uses GRPO without an explicit KL penalty and samples $n = 8$ full-workflow rollouts, or *episodes*, per problem. Rewards are normalized per problem and role, identically under both strategies, which differ only in the adapter that receives each role's loss (Appendix B.2).

Data and models. Math is DAPO-Math-17K (Yu et al., 2026) and Code is DeepCoder (Luo et al., 2025); validation accuracy is pass@1 on the 1412-problem DAPO-Math and 1000-problem DeepCoder-PrimeIntellect validation splits. We use Qwen3 (Yang et al., 2025) at 0.6B, 1.7B, and 4B, and cap each Math response at 5,120 tokens (Appendix B.1).

3.1 Multi-Agent Workflows

The workflows adapt the evaluator-optimizer, parallelization (voting), and orchestrator-workers patterns of Schluntz & Zhang (2024). In *Eval-Opt*, a generator answers and an evaluator returns a verdict and critique; any verdict other than correct sends the answer back for revision, up to three generator turns on Math. In *Voting*, three generators (the voters) answer independently and an aggregator selects among or combines their answers. In *Orch-Workers*, an orchestrator proposes a plan, three workers write candidate solutions, and a synthesizer writes the final answer. The *decision roles* are the evaluator, aggregator, and synthesizer, whose output sets the outcome. The voters, workers, and Eval-Opt generator are the *solving roles*; the generator's first turn is its *first attempt*. The voters and workers (three per episode) are the *repeated roles*.

3.2 Policy-Sharing Strategy

Under Shared-Policy (SP), all roles of a workflow use one policy, so every role's gradient updates the same parameters. Under Isolated-Policy (IP), each role type has its own adapter, shared by the role's instances. Both strategies use rank-64 LoRA adapters (Hu et al., 2022) on a frozen base model.

3.3 Diagnostics

Training diagnostics. Let $r_t = \pi_\theta(y_t \mid x, y_{<t})/\pi_{\text{rollout}}(y_t \mid x, y_{<t})$, where π_{rollout} is the policy that sampled y. For each adapter we log the policy-ratio $\chi^2 = \mathbb{E}_t[(r_t - 1)^2]$, the training perplexity,

the per-token entropy, and the gradient norm, each relative to its early median over the first 30 training steps. χ^2 and perplexity are computed on each micro-batch before the optimizer step of its mini-batch, so a role whose sequences fit in one mini-batch, such as the Voting aggregator or the Orch-Workers synthesizer, is measured before its own optimizer step, so its χ^2 reflects only the numerical mismatch between the sampling and training implementations. *Training success* is the fraction of training episodes with a correct final answer, and the *cap-hit rate* is the fraction of training responses that reach the length cap.

Validation and the selected peak. The *selected peak* of a run is its highest validation pass@1 among checkpoints at step ≤ 300; accuracy is reported there unless stated otherwise, and post-peak declines are read over each run's full validated horizon. Peak differences below about 2 points, within checkpoint-to-checkpoint noise, are read as ties.

Role-level decomposition. Each role output is graded on its own answer, independently of the episode's reward (Appendix B.4), and is *truncated* if it reaches the length cap and *finished* otherwise. The *plurality vote* is the most common boxed answer of the voters or workers, with ties broken at random. A *selection loss* is a Voting or Orch-Workers episode with a correct voter or worker and a wrong final answer; a *rescue* is an Orch-Workers episode with no correct worker and a correct final answer. An Eval-Opt episode is *recovered* when its first attempt is wrong and its final answer correct; the loop's *net contribution* is final minus first-attempt accuracy. Every role output receives the episode's outcome reward, the *broadcast reward* (Appendix B.2), so an output whose own answer is wrong in a rewarded episode is *rewarded but wrong*.

Table 1: Validation accuracy (%) at the selected peak, the highest validation pass@1 among checkpoints at step $\leq$ 300. Base model: the untrained model answering alone (single) or inside the workflow. SA-RL: single-agent RL at the same task and scale, shared across workflows. Residual: IP or SP minus SA-RL in percentage points; bold marks |residual| ≥ 5. Math has 1412 validation problems and Code (DeepCoder) 1000. All base values use the training response-length cap.

			Base model		Selected peak			Residual (pp)	
Task	Scale	Workflow	Single	Workflow	SA-RL	IP	SP	IP	SP
Math	0.6B	Eval-Opt	10.1	11.3	31.4	39.5	29.5	**+8.2**	−1.9
		Voting		11.7		28.5	28.3	−2.8	−3.1
		Orch-Workers		13.0		36.0	31.6	+4.7	+0.2
	1.7B	Eval-Opt	25.9	28.7	49.9	57.6	50.5	**+7.6**	+0.6
		Voting		31.2		50.9	46.5	+1.0	−3.4
		Orch-Workers		32.1		55.5	52.3	**+5.5**	+2.4
	4B	Eval-Opt	46.1	48.2	66.6	75.2	65.8	**+8.6**	−0.8
		Voting		53.0		68.1	56.2	+1.6	**−10.3**
		Orch-Workers		54.8		67.9	69.6	+1.4	+3.1
Code	0.6B	Eval-Opt	5.5	6.5	12.3	15.2	14.4	+2.9	+2.1
		Voting		6.9		12.6	13.0	+0.3	+0.7
		Orch-Workers		6.7		16.1	15.4	+3.8	+3.1
	1.7B	Eval-Opt	12.6	12.0	19.4	25.8	19.0	**+6.4**	−0.4
		Voting		12.7		19.5	18.8	+0.1	−0.6
		Orch-Workers		15.0		22.7	20.5	+3.3	+1.1
	4B	Eval-Opt	22.7	25.5	34.2	33.0	40.6	−1.2	**+6.4**
		Voting		26.6		33.9	29.4	−0.3	−4.8
		Orch-Workers		27.8		35.1	32.5	+0.9	−1.7

4 RESULTS

RL training improves every workflow over its untrained version, but the solving roles end no better than single-agent RL, and below it on Math, so a workflow beats single-agent RL only through what its structure adds (§4.1). Under isolated policies on Math, Eval-Opt does so at every scale (§4.2) and Orch-Workers at the two smaller scales, while Voting, whose aggregator adds little beyond a plurality vote, ties with or trails single-agent RL (Table 1). On Code, Eval-Opt's loop lifts some cells above

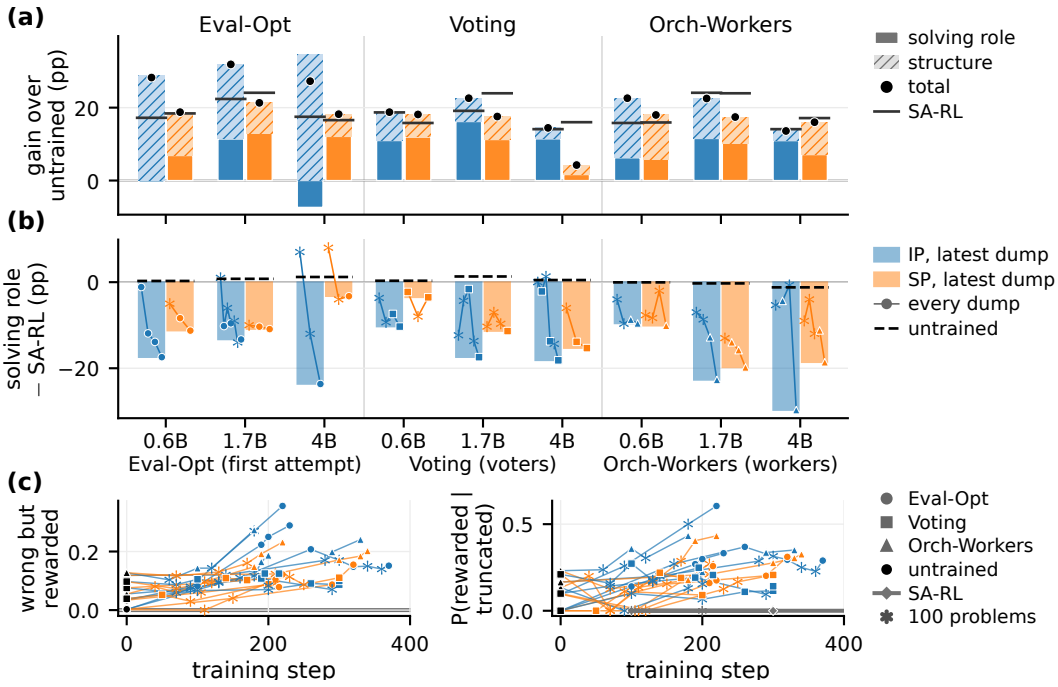

Figure 2: RL gain and solving roles under the broadcast reward on Math. (a) Gain over the untrained workflow at the checkpoint nearest the selected peak: solving-role gain (solid) plus structural gain, the change in workflow minus solving-role accuracy (hatched); dot: total; line: SA-RL's gain over the untrained single model at the nearest evaluated step. (b) Solving role minus SA-RL on the same problems at the nearest evaluated step; bars: latest sampled checkpoint; dots: all sampled checkpoints, in step order; dashed: untrained. (c) Share of solving-role outputs that are rewarded but wrong, and share of truncated ones that are rewarded, at the checkpoints of (b); gray: SA-RL. All values on the 1412 validation problems except those marked $*$, on 100 sampled problems. Values: Table 6.

single-agent RL. Isolated policies learn faster and peak higher than a shared policy, whose solving roles learn slowly (§4.3), but a repeated role's adapter can become unstable after the peak, while a shared policy declines little (§4.4).

4.1 Which Workflows Gain From RL, and Where Does the Gain Come From?

Every workflow gains from RL, but under the broadcast reward its solving roles end no better than single-agent RL, so only the structure above them can lift a workflow above single-agent RL. On Math, under isolated policies, Eval-Opt's revision loop does so at every scale and the Orch-Workers synthesizer at the two smaller scales, while Voting's aggregator adds little beyond a plurality vote. On Code the solving roles stay close to single-agent RL, and Eval-Opt's loop lifts some cells above it.

Result. Before training, every workflow scores above the untrained single model on Math, by margins that grow with scale for Voting and Orch-Workers (Table 1). RL lifts every cell above its base workflow on both tasks (Fig. 6, Appendix C.1) and reorders the workflows. Eval-Opt goes from the weakest workflow on Math to the strongest under IP, and Voting ranks last among the three workflows in most combinations of task, scale and strategy. On Code most cells end within a few points of SA-RL, and `Eval-Opt-IP-1.7B-Code` and `Eval-Opt-SP-4B-Code` exceed it clearly.

Analysis: where the gain lands. A workflow's RL gain splits into its solving role's gain and a structural gain (Fig. 2(a)). The structural gain is the change in what the plurality vote, the synthesizer or the revision loop adds above the solving role. In every Math cell the solving role gains less than the single model under SA-RL, so a workflow's margin over SA-RL is what its structure adds minus the solving role's deficit (Table 6). Under IP, Eval-Opt has the largest structural gain at every scale, from its revision loop (§4.2), and Orch-Workers the second largest at 0.6B and 1.7B, exactly the cells that

Table 2: Decision-role rules on Math at the latest sampled IP and SP checkpoints, all 1412 validation problems. k/n: count and denominator. Voting: workflow minus plurality-vote accuracy (points); lone correct voters picked against a wrong pair. Eval-Opt: evaluator's first verdict on truncated (T) or finished (F) wrong first attempts. Orch-Workers: share of answer-only synthesizer outputs (these return the workers' plurality answer) and of episodes with a selection loss or a rescue (§3.3). Steps, untrained workflow, other columns: Table 7; definitions: §C.3.

	Voting aggregator				*Eval-Opt evaluator*				*Orch-Workers synthesizer*					
	Δ vs. plurality		Lone correct		Accepts T-wrong		Rejects F-wrong		Answer-only		Loss		Rescue	
Scale	IP	SP	IP	SP	IP	SP	IP	SP	IP	SP	IP	SP	IP	SP
0.6B	+0.4	−0.1	0/15	2/25	0/1257	24/1007	3/16	2/162	0.06	0.41	0.02	0.03	0.10	0.06
1.7B	+0.5	+0.3	4/21	3/40	73/756	28/714	2/99	12/145	0.39	0.35	0.01	0.04	0.14	0.07
4B	+3.1	+1.7	5/24	15/46	0/837	3/373	1/11	115/200	0.41	0.55	0.01	0.02	0.16	0.09

exceed SA-RL. At 4B the structural gains of Orch-Workers and Voting are both small, and neither workflow exceeds SA-RL (Table 1). Orch-Workers gains through its workers early and through its synthesizer late (Table 6); the trained IP synthesizer loses fewer worker answers and rescues more than the untrained one (Table 7, Appendix C.3). Voting's gain is mostly its voters' own (Table 2).

Analysis: solving roles under the broadcast reward. Graded on their own answers, the voters, the workers and the Eval-Opt first attempt all end below same-scale SA-RL on Math, though some match it early (Fig. 2(b)). The broadcast reward scores each output by its episode, so a wrong or truncated output is rewarded whenever the episode succeeds; a truncated SA-RL output almost never is (Fig. 2(c)). SA-RL's cap-hit rate rises and then falls back; the workflows' rarely does (Fig. 7, Appendix C.2). A control of `Voting-IP-1.7B-Math` that changes only the reward, scoring each voter on its own answer, leads the reference at every matched validation step (Tables 19 and 20, Appendix D.4). Its cap-hit rate falls back like SA-RL's and earlier than the reference's, yet it still declines after its peak (§4.4). Rewarding the evaluator for its verdict's accuracy instead can make it reject every attempt (Appendix D.4).

Analysis: what the decision roles learn. Each trained decision role applies a simple rule (Table 2). The Voting aggregator scores close to a plurality vote below 4B and a few points above it at 4B. A lone correct voter against a wrong pair is rare, and the trained aggregator picks it in a minority of these cases in every cell, so Voting's structure adds little beyond the plurality rule. The Eval-Opt evaluator rejects truncated first attempts and, outside `Eval-Opt-SP-4B-Math`, rarely a finished wrong one, so the loop gives truncated first attempts another turn (§4.2). The answer-only Orch-Workers synthesizer returns the workers' most common answer (Table 7), so the rescues come from its long form. Output form changes under both strategies with no measured accuracy cost (Table 8).

Analysis: code. On Code the solving roles stay close to SA-RL, a few points below at 4B (Table 10). They reach the length cap at least as often as on Math, but almost only on problems no instance solves, so truncated and wrong outputs are rarely rewarded and the reward stays close to each role's own correctness. Instances disagree on correctness far less often than on Math, and the decision role loses few correct answers below 4B (Tables 10 and 9). A workflow's margin over SA-RL is then its structure's: Eval-Opt's loop lifts some cells above it, Orch-Workers under IP adds a few points at the two smaller scales, and Voting mostly ties (Tables 11 and 1).

4.2 Why Does Eval-Opt Gain Most?

Its revision loop extends the token budget of truncated first attempts. The evaluator rejects truncated, not wrong, answers, so the loop rarely recovers a finished wrong attempt. Eval-Opt's whole margin over single-agent RL is the loop's, as is all of its gain over the base workflow at 0.6B and 4B and half or more at 1.7B, where the first attempt also improves.

Result. Eval-Opt under IP is the only workflow that exceeds SA-RL at every Math scale, under every horizon rule (Table 1; Table 5, panel B). Its first attempt ends below SA-RL on Math, so the whole margin is the revision loop's (Fig. 3(a)). The loop recovers truncated first attempts and rarely a finished wrong one, most often in `Eval-Opt-SP-4B-Math` (panel (b)). The loop also carries all

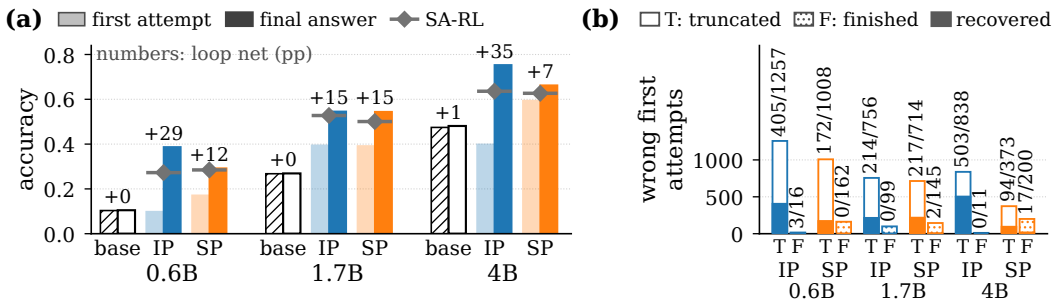

Figure 3: Eval-Opt on Math, each generator turn graded on its own answer, all 1412 validation problems. (a) First-attempt (light) and final (dark) accuracy at the latest sampled checkpoint and for the untrained workflow (first attempt hatched, final open), with SA-RL at the nearest evaluated step (gray); numbers: the loop's net contribution. (b) Wrong first attempts at the same checkpoints, truncated (T) or finished (F); filled: recovered. Values: Table 11.

or most of the gain over the untrained workflow (Fig. 3(a); Fig. 2(a)). At the selected peak the IP advantage over SP is entirely the loop's at every scale, since the IP first attempt is no more accurate than the SP one (Fig. 2(a)); at 1.7B the two strategies tie at a common step (Table 11, bottom).

Analysis: role outputs. On Math the generator gets up to three turns where SA-RL gets one, and late IP episodes use more generator tokens than one SA-RL response holds (Table 11). At a matched inference budget, a plurality vote over three SA-RL samples comes within a few points of Eval-Opt under IP at every Math scale and matches or exceeds every Voting and Orch-Workers cell (Table 4, Appendix B.3). Under the evaluator's truncation rule (§4.1), a finished wrong first attempt usually ends the episode, and the loop almost never loses a correct one (same table). The IP generator comes to finish its first attempt mainly when it is likely right (Fig. 8, Appendix C.5). On Code the loop adds less in most cells, and the lead of `Eval-Opt-SP-4B-Code` over `Eval-Opt-IP-4B-Code` is within noise at a common horizon (Table 11; Table 5, panel B).

4.3 Why Do Isolated Policies Beat a Shared Policy?

Isolated policies learn faster and peak higher. On Math, under a shared policy, averaging each output's loss over its own tokens gives the decision role, whose outputs are much shorter, well over its uniform share of the gradient, and the solving roles learn slowly. Shared-policy runs decline little after their peak (§4.4).

Result. IP learns faster than SP in nearly every matched pair, every Math pair included, and peaks higher in most (Fig. 4, panels (b) and (a)). Because SP runs decline little while several IP runs do (§4.4), the advantage lies at the peak, and SP leads at the last common validated step in most of those pairs (Table 5, panel B). On Code the two tie at a matched horizon in most pairs.

Analysis: gradients and parameter change. SP is not slow for lack of movement, since its shared adapter moves farther per training step than every IP adapter of the matched run in most pairs (Table 16, panel D3; Fig. 9, Appendix D.3). Because each sequence's loss is averaged over its own tokens, short outputs carry large per-sequence gradients, and the length model, which applies this weighting, fits the SP shares far better than the token share (Fig. 4, panel (c)). On Math the Voting and Eval-Opt decision roles write much shorter outputs than the solving roles and take most of the shared gradient, leaving the solving roles well below their uniform share (Table 16, panel D1). Projected on the summed gradient, which nets out gradient that other roles cancel, the decision role's share is larger still (same panel). In the probed Code runs the lengths are closer, and the solving role's share stays near its uniform share, far above its Math shares, consistent with the tie (same panel). In `Voting-IP-1.7B-Math` the aggregator holds a comparable share (panel D2) but updates only its own adapter, not the parameters the solving roles use.

Analysis: role outputs. Graded on their own answers, the SP solving roles end below SA-RL on Math and usually reach the length cap less often than the IP ones (Table 6). The

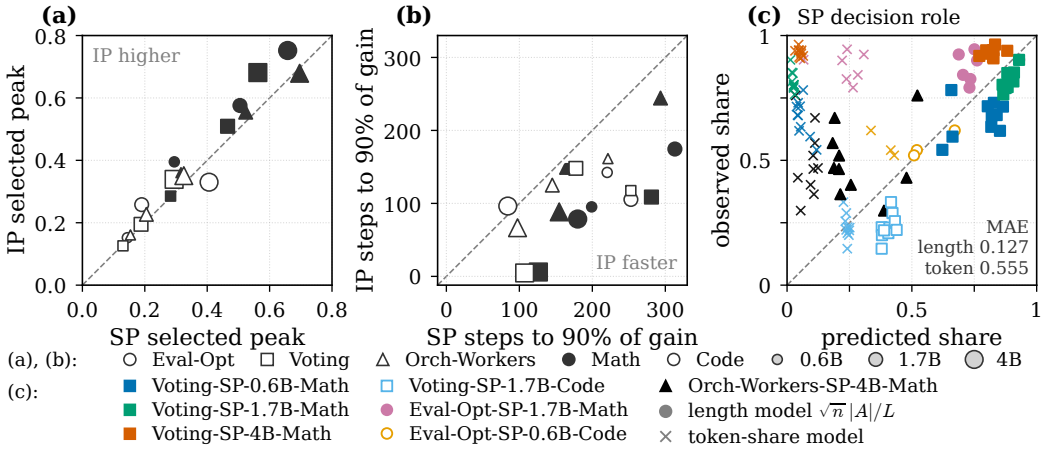

Figure 4: IP against SP per matched pair. (a) Selected peak. (b) Training steps to 90% of the pair's common gain (Appendix C.1). (c) SP decision role's share of the shared gradient at each probed checkpoint of seven runs, against the length model $\sqrt{n}\,|A|/L$ (n sequences, mean absolute advantage $|A|$, mean length L) and the token share; inset: mean absolute error over all roles.

`Voting-SP-4B-Math` voters, with the smallest measured gradient share, stay near their base accuracy while the IP voters first gain (same table). Sharing thus leaves the solving roles too little learning signal.

4.4 Why Do Isolated Policies Decline After Their Peak?

On Math the repeated role's adapter can become unstable after the peak, and only that role's χ^2 and perplexity rise. Training success falls with validation, so a stopping rule on it fires only after the peak but can return a checkpoint near it. No shared-policy, single-agent or Code run declines comparably.

Result. On Math, five IP runs with a repeated role decline significantly after their validation peak, Voting at every scale and Orch-Workers at 1.7B and 4B, and `Eval-Opt-IP-1.7B-Math` declines late in its run (Fig. 5; Table 18). Elsewhere only `Orch-Workers-SP-4B-Math` ends significantly below its peak, by a few points.

Analysis: training metrics. Training success is graded like validation, so its fall means lost problems, not reward exploitation (Table 18). At 4B the decline begins within the first pass over the training set, so it is not overfitting (Fig. 5). In the Voting and Orch-Workers IP runs only the repeated role's χ^2 and perplexity rise; neither role of `Eval-Opt-IP-1.7B-Math` shows a comparable rise (Table 14). Most shared adapters, also measured after earlier optimizer steps of the training step, show no comparable rise; the two that do belong to the two SP runs that end a few points below their peak (Tables 19 and 14; Appendix C.6). The rise follows the batch's per-token entropy, not its response length (Appendix D.3).

Analysis: role outputs. At 1.7B and 4B the share of problems with a correct voter or worker falls after the peak (Table 17). The share of answers the decision role loses stays flat or falls, so accuracy is lost in the solving instances (same table). In the two 1.7B runs the instances' outputs lengthen, reach the cap more often, hedge more and share fewer phrases (Table 15).

Analysis: controls. With the mini-batch size fixed in sequences, the repeated role's adapter takes one optimizer step per instance in each training step (Table 19, Appendix D.4). Its parameter change per training step does not separate the declining runs from the rest (Appendix D.3). The single-role control keeps only this step count, one adapter taking as many steps without a workflow, and declines like `Voting-IP-1.7B-Math`, while SA-RL declines at no scale (Tables 19 and 20). Per-voter reward speeds the rise and per-voter prompts delay the peak, but both still decline; one adapter per voter instance, which also changes the reward, does not (Table 20). A decline therefore occurs

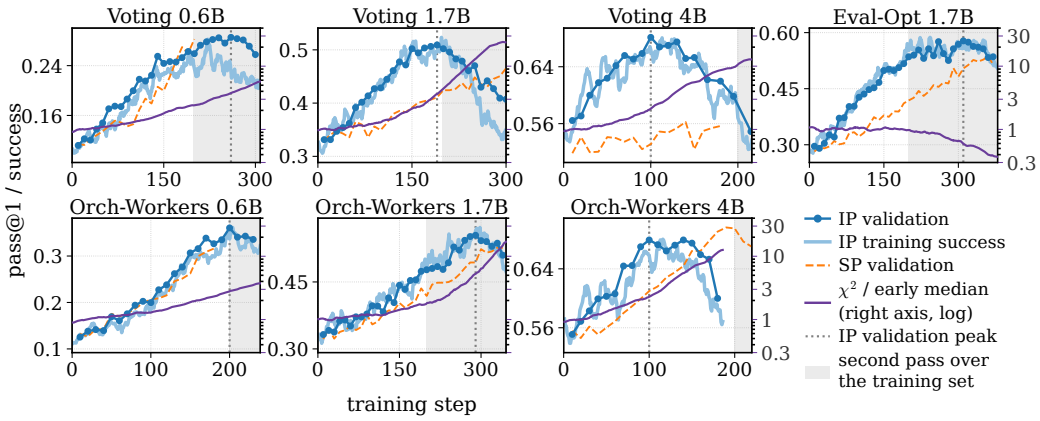

Figure 5: Validation accuracy, training success (10-step mean) and χ^2 of the isolated-policy Math runs with a repeated role and of `Eval-Opt-IP-1.7B-Math`; χ^2: the repeated role's (Eval-Opt: generator's) over its early median; shaded: second pass over the training set. All runs but `Orch-Workers-IP-0.6B-Math` ($p = 0.17$) end significantly below their peak ($p < 0.02$; exact McNemar or two-proportion test); elsewhere in the grid only `Orch-Workers-SP-4B-Math` does (2.6 points, $p = 0.008$).

without the workflow and without the broadcast reward. On Code the repeated adapters take as many optimizer steps, yet their χ^2 rises far less and no run declines (Tables 18 and 9). Since the per-voter-reward control still declines on Math, the Code reward accounting of §4.1 does not explain this difference alone.

Analysis: monitoring. Because the rise is confined to the repeated role, its χ^2 locates the instability without a validation set. It exceeds twice its early level in every Voting and Orch-Workers IP run on Math, with one false positive among the non-declining runs with a repeated role (Table 12, panel A). The last checkpoint before that crossing is often below the peak, so it marks the instability rather than a checkpoint to keep (Table 13). Global entropy, perplexity and the cap-hit rate fire late or also in runs without a decline, and the gradient norm follows output length (Table 12, panels A and B). In an online replay, a rule that stops once the trailing mean of training success stays below its running maximum gives no advance warning, but where it fires it returns a checkpoint close to the peak; it does not fire in every declining run, rarely fires elsewhere and needs no per-role logging (Tables 12 and 13).

5 Conclusion

We asked when end-to-end RL improves a multi-agent LLM workflow over its base model and what each role learns, across 36 cells of workflow, task, scale, and policy-sharing strategy. Workflow structure helps on Math before training, and RL lifts every workflow above its base workflow. Under the broadcast outcome reward the solving roles end no better than single-agent RL (below it on Math, close to it on Code), so a workflow exceeds single-agent RL only through what its structure adds, set by the simple rule each decision role learns. Eval-Opt gains most because its revision loop gives truncated first attempts another turn, extending the token budget; a plurality vote over three single-agent RL samples, at a matched budget, comes close to it. IP learns faster than SP and peaks higher in most matched pairs; under SP on Math the decision role, whose outputs are much shorter, takes well over its uniform share of the shared gradient, and the solving roles learn slowly. On Math, however, most IP runs with a repeated role decline after their peak, while SP and Code runs decline little. The decline is an instability of the repeated role, visible in its χ^2; a stopping rule on training success fires only after the peak but can return a checkpoint near it. Appendix A discusses design and monitoring implications and limitations.

AI USE STATEMENT

LLMs are the object of study in this paper: we train and evaluate LLM-based agent workflows, described in §3. Separately, an LLM-based assistant (Claude Code) was used for implementation, experiment and analysis scripting, and for editing the text; it is not part of the core methodology, and all generated code was reviewed and validated by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work trains and evaluates publicly available base models (Qwen3 (Yang et al., 2025)) on publicly available math and code datasets (DAPO-Math-17K (Yu et al., 2026) and DeepCoder (Luo et al., 2025)), all openly licensed for research use. It involves no human subjects, crowdsourcing, or personal data, and it releases no new datasets or pretrained models.

REPRODUCIBILITY STATEMENT

§3 specifies the task, scale, policy-sharing, and workflow grid, the datasets, the rollout and reward-normalization scheme, and the diagnostics and role-level grading used in §4. §B.1 lists the training hyperparameters for every cell, §B.2 gives the reward functions, the grouping and normalization rule, and which adapter each role's loss updates, §B.3 describes the single-agent baseline, §B.4 the sampling of the trajectory statistics, and §B.6 reports the hardware and compute. We will release the training and evaluation code together with the configuration files for every cell.

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. *arXiv preprint arXiv:2506.02718*, 2025.

Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V. Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing, 2021. URL https://arxiv.org/abs/2102.07475.

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.

Apurva Gandhi, Satyaki Chakraborty, Xiangjun Wang, Aviral Kumar, and Graham Neubig. Recursive agent optimization. *arXiv preprint arXiv:2605.06639*, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Haoyang Hong, Jiajun Yin, Yuan Wang, Jingnan Liu, Zhe Chen, Ailing Yu, Ji Li, Zhiling Ye, Hansong Xiao, Yefei Chen, Hualei Zhou, Yun Yue, Minghui Yang, Chunxiao Guo, Junwei Liu, Peng Wei, and Jinjie Gu. Multi-agent deep research: Training multi-agent systems with M-GRPO. *arXiv preprint arXiv:2511.13288*, 2025.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. 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.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, Yanru Chen, Yanxu Chen, Yicun Chen, Yimin Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Ziwei

Chen, Dazhi Cheng, Yean Cheng, Minghan Chu, Jialei Cui, Jiaqi Deng, Muxi Diao, Hao Ding, Mengfan Dong, Mengnan Dong, Yuxin Dong, Yuhao Dong, Angang Du, Chenzhuang Du, Dikang Du, Lingxiao Du, Yulun Du, Yu Fan, Shengjun Fang, Qiulin Feng, Yichen Feng, Garimugai Fu, Kelin Fu, Hongcheng Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Chengyang Gong, Xiaochen Gong, Zhuoma Gongque, Qizheng Gu, Xinran Gu, Yicheng Gu, Longyu Guan, Shuhao Guan, Yuanying Guo, Xiaoru Hao, Dailan He, Tianhong He, Weiran He, Wenyang He, Yibo He, Yunjia He, Chao Hong, Hao Hu, Jiaxi Hu, Yangyang Hu, Zhenxing Hu, Ke Huang, Ruiyuan Huang, Weixiao Huang, Zhiqi Huang, Chaobo Jia, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yu Jing, Guokun Lai, Aidi Li, C. Li, Cheng Li, Fang Li, Guanghe Li, Guanyu Li, Haitao Li, Haoyang Li, Jia Li, Jingwei Li, Junxiong Li, Lincan Li, Mo Li, Weihong Li, Wentao Li, Xinhang Li, Xinhao Li, Yang Li, Yanhao Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Weilong Liao, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zhishan Lin, Zichao Lin, Cheng Liu, Chenyu Liu, Hongzhang Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Tianyu Liu, Weizhou Liu, Xiangyan Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yuanxin Liu, Zhengying Liu, Zhongnuo Liu, Enzhe Lu, Haoyu Lu, Zhiyuan Lu, G. Luo, Junyu Luo, Tongxu Luo, Yashuo Luo, Long Ma, Shaoguang Mao, Yuan Mei, Xin Men, Fanqing Meng, Zhiyong Meng, Yibo Miao, Minqing Ni, Kun Ouyang, Siyuan Pan, Bo Pang, Yuchao Qian, Ruoyu Qin, Zeyu Qin, Jiezhong Qiu, Bowen Qu, Zeyu Shang, Youbo Shao, Tianxiao Shen, Zhennan Shen, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Feifan Song, Pengwei Song, Tianhui Song, Xiaoxi Song, Hongjin Su, Jianlin Su, Zhaochen Su, Lin Sui, Jinsong Sun, Junyao Sun, Tongyu Sun, Flood Sung, Yunpeng Tai, Chuning Tang, Heyi Tang, Xiaojuan Tang, Zhengyang Tang, Jiawen Tao, Shiyuan Teng, Chaoran Tian, Pengfei Tian, Bowen Wang, Chensi Wang, Chuang Wang, Congcong Wang, Dingkun Wang, Dinglu Wang, Dongliang Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hengzhi Wang, Huaqing Wang, Hui Wang, Jiahao Wang, Jinhong Wang, Jiuzheng Wang, Kaixin Wang, Linian Wang, Qibin Wang, Shengjie Wang, Shuyi Wang, Si Wang, Wei Wang, Xiaochen Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yipu Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhexu Wang, Zifan Wang, Zihan Wang, Zizhe Wang, Chu Wei, Ming Wei, Chuan Wen, Zichen Wen, Chengjie Wu, Haoning Wu, Junyan Wu, Rucong Wu, Wenhao Wu, Yuefeng Wu, Yuhao Wu, Yuxin Wu, Zijian Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Yuchong Xie, Bowei Xing, Boyu Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinbo Xu, Xinran Xu, Yangchuan Xu, Yichang Xu, Yuemeng Xu, Zelai Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Guangyao Yang, Hao Yang, Junwei Yang, Kai Yang, Ningyuan Yang, Xiaofei Yang, Xinlong Yang, Xinyu Yang, Ying Yang, Yi Yang, Yi Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhuorui Ye, Peng Yebo, Bohong Yin, Chengzhen Yu, Longhui Yu, Tao Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Xiaokun Yuan, Yang Yue, Weihao Zeng, Dunyuan Zha, Haobing Zhan, Dehao Zhang, Hao Zhang, Jin Zhang, Puqi Zhang, Qiao Zhang, Rui Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yushun Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Chenguang Zhao, Feifan Zhao, Jinxiang Zhao, Shuai Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Junfeng Zhong, Longguang Zhong, Weiming Zhong, M. Zhou, Runjie Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yuxuan Zhu, Zhen Zhu, Jingze Zhuang, Weiyu Zhuang, Ying Zou, and Xinxing Zu. Kimi K2.5: Visual agentic intelligence, 2026. URL https://arxiv.org/abs/2602.02276.

Junwei Liao, Muning Wen, Jun Wang, and Weinan Zhang. MARFT: Multi-agent reinforcement fine-tuning. *arXiv preprint arXiv:2504.16129*, 2025.

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

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51, 2025. Notion Blog.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

Sumeet Ramesh Motwani, Chandler Smith, Rocktim Jyoti Das, Rafael Rafailov, Ivan Laptev, Philip H. S. Torr, Fabio Pizzati, Ronald Clark, and Christian Schroeder de Witt. MALT: Improving reasoning with multi-agent LLM training, 2025. URL https://arxiv.org/abs/2412.01928.

Chanwoo Park, Seungju Han, Xingzhi Guo, Asuman Ozdaglar, Kaiqing Zhang, and Joo-Kyung Kim. MAPoRL: Multi-agent post-co-training for collaborative large language models with reinforcement learning, 2025. URL https://arxiv.org/abs/2502.18439.

Erik Schluntz and Barry Zhang. Building effective AI agents. https://www.anthropic.com/engineering/building-effective-agents, 2024. Anthropic Engineering Blog.

John Schulman and Thinking Machines Lab. Lora without regret. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20250929. https://thinkingmachines.ai/blog/lora/.

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

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A flexible and efficient RLHF framework, 2024. URL https://arxiv.org/abs/2409.19256.

Vighnesh Subramaniam, Yilun Du, Joshua B. Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains, 2025. URL https://arxiv.org/abs/2501.05707.

Justin K. Terry, Nathaniel Grammel, Sanghyun Son, Benjamin Black, and Aakriti Agrawal. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020. doi: 10.48550/arXiv.2005.13625.

Ziyu Wan, Yunxiang Li, Xiaoyu Wen, Yan Song, Hanjing Wang, Linyi Yang, Mark Schmidt, Jun Wang, Weinan Zhang, Shuyue Hu, and Ying Wen. ReMA: Learning to meta-think for LLMs with multi-agent reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09501.

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.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning. *arXiv preprint arXiv:2504.20073*, 2025.

Zhepei Wei, Wenlin Yao, Yao Liu, Weizhi Zhang, Qin Lu, Liang Qiu, Changlong Yu, Puyang Xu, Chao Zhang, Bing Yin, Hyokun Yun, and Lihong Li. Webagent-r1: Training web agents via end-to-end multi-turn reinforcement learning. *arXiv preprint arXiv:2505.16421*, 2025.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. WideSeek-R1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning, 2026. URL https://arxiv.org/abs/2602.04634.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shengtian Yang, Ziyu Xiong, Yu Li, Yewen Li, Qingpeng Cai, and Lei Feng. SRPO: Setwise relative policy optimization for multi-agent LLMs, 2026a. URL https://arxiv.org/abs/2609.08452.

Yingxuan Yang, Chengrui Qu, Muning Wen, Laixi Shi, Ying Wen, Weinan Zhang, Adam Wierman, and Shangding Gu. Understanding agent scaling in LLM-based multi-agent systems via diversity. *arXiv preprint arXiv:2602.03794*, 2026b.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025a.

Wenting Zhao, Pranjal Aggarwal, Swarnadeep Saha, Asli Celikyilmaz, Jason Weston, and Ilia Kulikov. The majority is not always right: RL training for solution aggregation, 2025b. URL https://arxiv.org/abs/2509.06870.

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*, 2025c.

A Discussion and Limitations

A.1 Design Implications

Per-role rewards. The broadcast reward lets wrong and truncated solving-role outputs share a successful episode's reward (§4.1), so rewarding each solving role on its own answer is the direct design change. On Voting it raises validation accuracy but does not prevent the post-peak decline (Table 20). A judge role needs a reward that no fixed verdict maximizes. Rewarded on the accuracy of its verdict, the Eval-Opt evaluator can learn to reject every attempt (§D.4).

Baselines for revision and selection. Eval-Opt's margin over single-agent RL comes from the extra generator turns it gives truncated first attempts (§4.2). At a matched inference budget, a plurality vote over three SA-RL samples comes within a few points of Eval-Opt under IP at every Math scale and matches or exceeds every Voting and Orch-Workers cell (Table 4), so extra samples of the single-agent policy buy what the loop buys. The trained Voting aggregator scores close to a plurality vote (Table 2), so a fixed plurality rule is the reference against which any trained selection role should be measured.

Checkpoint selection. On Math, most IP runs with a repeated role decline after their peak (§4.4), so an IP checkpoint should be selected on validation. SP runs decline little, but they leave the solving roles little learning signal and peak lower in most matched pairs (§4.3).

A.2 Monitoring Recommendations

Three signals complement validation accuracy. The repeated role's χ^2 locates the instability in the declining IP runs (Table 12), but a role whose sequences fit in one mini-batch is measured before its own optimizer step and records little of its own change (§3.3). A stopping rule on the trailing training-success curve needs no per-role logging and returns a checkpoint close to the peak, although it fires only after the peak (Table 13). A decision role's output form can change without any change in accuracy. When the output is bimodal, such as a Voting aggregator that writes either a short boxed selection or a long review, the rate of each form records the change more directly than length percentiles (Table 8).

A.3 Limitations

Setting. The grid uses one model family (Qwen3 at 0.6B, 1.7B, and 4B), one dataset per domain (DAPO-Math-17K and DeepCoder), and three workflows. Agents solve non-interactive problems without tools or long horizons, and every role is trained on the final-answer reward. Results may differ for other models, datasets, topologies, agentic settings, or process rewards that score intermediate role outputs.

LoRA and optimizer. All runs train rank-64 LoRA adapters on a frozen base model with GRPO, AdamW, and no KL penalty (Schulman & Lab, 2025). The step-count model of parameter change in §D.3 assumes an adaptive optimizer, whose change per optimizer step depends little on the gradient's magnitude. We ran no LoRA-rank, full-parameter, alternative-optimizer, or KL-anchored ablations.

One seed per cell. Each cell is trained once. Cross-cell direction counts are descriptive, and the evaluation-set intervals condition on the selected checkpoint and exclude seed and checkpoint-selection variance.

Nominal resource matching. Isolated-Policy, Shared-Policy, and the single-agent baseline share the optimizer, LoRA rank, sampling configuration, sequence caps, and step budget. They are not matched on parameter count (Isolated-Policy trains one adapter per role type), role calls, generated tokens (§4.2), wall-clock time, or compute (§B.6). Comparisons between them are accuracy contrasts within one training stack, not compute-matched effects; Table 4 adds one inference-matched single-agent baseline on Math.

Other multi-agent training rules. We did not train with AT-GRPO's per-turn candidate sampling and role-local rewards (Zhao et al., 2025c); §E relates our training rule to AT-GRPO and Dr. MAS.

B Experimental Details

B.1 Hyperparameters

Table 3 lists the training hyperparameters. Each training batch is sorted by sequence length, without shuffling, before it is split into mini-batches. A mini-batch holds a fixed number of sequences and takes one optimizer step, so an adapter that receives several sequences per episode takes several optimizer steps per training step.

Learning-rate warmup. Nine Code cells warm the learning rate up: Eval-Opt at 4B, Voting at 0.6B and Orch-Workers at 0.6B and 4B under both strategies, and `Voting-SP-4B-Code`. So do the 0.6B single-agent Code baseline and the per-voter-prompts, one-adapter-per-voter and single-role controls of `Voting-IP-1.7B-Math` (Table 19). Every other cell trains at the constant rate from the first step. The scheduler advances once for each adapter updated in a training step, so the warmup covers the first 15 training steps under Shared-Policy and in the single-agent baseline and the first 5 to 8 training steps under Isolated-Policy. Table 20 gives the step at which each control reaches the constant rate. The compared peaks and declines occur after the warmup.

B.2 Rewards and Advantage Normalization

Math. An episode's outcome reward is 1 if the last `\boxed{}` expression of its final answer parses and matches the ground-truth answer of DAPO-Math-17K, and 0 otherwise, including when the boxed answer is missing or malformed.

Code. An episode's outcome reward is 1 if the last Python code block of its final answer passes every test in the problem's hidden test suite and 0 otherwise, including code that fails to parse or compile or exceeds the per-test time limit.

Grouping, normalization, and adapter assignment. For each problem, we sample $n = 8$ independent full-workflow rollouts and compute one outcome reward per completed rollout, which every role's sequences in that rollout receive; no per-step or per-role process reward is used. Each advantage is the reward minus the mean of its group, divided by the group's standard deviation, with groups keyed by problem and role. Under Voting on mathematics, each of the three generator instances forms one group and the aggregator another, each with 8 sequences; under Voting on code, all generator instances of a problem form one group. Under Eval-Opt, all turns of a role across the problem's rollouts form one group. Under Orch-Workers, the groups are the orchestrator, the pooled workers, and the synthesizer. This is agent-wise normalization in the sense of Dr. MAS (Feng et al., 2026), not a global baseline pooling all roles (§E). The grouping rule is the same under Shared-Policy and Isolated-Policy and is applied before each role's loss is assigned to an adapter. Gradients are accumulated only over tokens emitted by a role, with other roles' tokens masked from that loss; under Shared-Policy all role losses update one adapter, whereas under Isolated-Policy each loss updates

Table 3: Training hyperparameters, fixed across the grid unless a per-cell value is given. †Cells listed in §B.1.

Group	Hyperparameter	Value
Optimization	Optimizer	AdamW, weight decay 0.01
	Learning rate	2×10^{-5}
	Learning-rate schedule	constant, no decay
	Warmup, all Math cells and nine Code cells	none
	Warmup, other nine Code cells†	linear from 0 over 15 scheduler updates
	Gradient clipping	1.0
	PPO clip ratio (low / high)	0.2 / 0.28
LoRA adapter	Rank r	64
	Alpha α	32
	Dropout	0.0
	Target modules	all linear modules
GRPO	Algorithm	GRPO (no explicit KL)
	Group size n	8
	Sampling unit	episode (independent full-workflow rollout)
	Advantage grouping	per problem and role (§B.2)
	Advantage normalization	group-relative, before adapter assignment
	Cross-role global baseline	none
	PPO mini-batch size	512 sequences (64 problems $\times$ n=8)
	PPO epochs per update	1
	Loss aggregation	mean over each sequence's tokens, then over sequences
Batching	Train batch size (problems / step)	64
	Validation batch size	2048
	Rollout temperature	0.7
Sequence length	Eval-Opt, Math (prompt / response)	30720 / 5120
	Eval-Opt, Code (prompt / response)	10240 / 2048
	Voting, Math (prompt / response)	20480 / 5120
	Voting, Code (prompt / response)	10240 / 2048
	Orch-Workers, Math (prompt / response)	20480 / 5120
	Orch-Workers, Code (prompt / response)	10240 / 2048
Workflow caps	Eval-Opt evaluation rounds per episode, Math	at most 3 (at most 2 revisions)
	Eval-Opt evaluation rounds per episode, Code	at most 2 (at most 1 revision)
	Voting generator calls per episode	3
	Orch-Workers worker calls per episode	at most 3
Training schedule	Validation interval (steps)	10
	Checkpoint interval (steps)	5 or 10, by cell
Adapter assignment	Isolated-Policy	one LoRA adapter per role type
	Shared-Policy	one shared LoRA adapter across all roles

its role-type adapter. Thus normalization groups and parameter-sharing groups are distinct design choices.

B.3 Single-Agent RL Baseline

The SA-RL column of Table 1 trains one LoRA adapter on a single-role workflow whose prompt matches the multi-agent generator prompt, with the same base model, task, optimizer, LoRA configuration, GRPO implementation, and step budget as the multi-agent cells at that scale (resource matching: §A.3).

Matched inference budget. Table 4 draws three samples per Math validation problem from the SA-RL checkpoint at its selected peak, each under the training length cap, and scores their plurality answer. Three samples use no more generator calls or tokens than a Voting episode or a full Eval-Opt episode. Their mean accuracy reproduces the SA-RL column of Table 1.

B.4 Trajectory Statistics

The trajectory statistics in Tables 8 and 15 are computed on one episode per problem on the first 100 problems of the DAPO-Math and DeepCoder-PrimeIntellect validation splits of §3. Table 15 samples each cell at three checkpoints; Table 8 uses the base model and the earliest and latest sampled checkpoint under each strategy, which differ between IP and SP and are not matched by step. The other role-level displays grade one episode per problem as well. On Math they use all 1412 validation

Table 4: Single-agent RL at a matched inference budget on Math (1412 validation problems, training length cap). Three samples of the SA-RL checkpoint at its selected peak per problem; one sample: their mean accuracy; plurality: the most common boxed answer, with ties broken at random (expected accuracy; range: every tie resolved wrongly or correctly); pass@3: at least one sample correct. Workflow columns: selected peaks of Table 1. Accuracy in %.

	SA-RL, three samples				Eval-Opt		Voting		Orch-Workers	
Scale	one sample	plurality	range	pass@3	IP	SP	IP	SP	IP	SP
0.6B	30.8	36.9	33.0–42.1	43.6	39.5	29.5	28.5	28.3	36.0	31.6
1.7B	49.6	57.8	54.8–61.5	63.0	57.6	50.5	50.9	46.5	55.5	52.3
4B	66.8	74.0	72.7–75.6	76.8	75.2	65.8	68.1	56.2	67.9	69.6

problems where a checkpoint has been evaluated on the full set and these 100 sampled problems otherwise, with * marking the values on the 100 sampled problems; Table 10 uses the 100 sampled problems at every checkpoint of both tasks. On Code they use the 100 sampled problems. SA-RL values are taken at its nearest evaluated or sampled step and marked by problem set in the same way. Every role output is graded on its own answer (§3.3), the last boxed answer on Math and the last code block on Code, independently of the episode's reward. Token counts are completion tokens. The remaining measurement definitions are in §B.5.

B.5 Measurement Definitions

The appendix displays share the definitions below, in addition to the diagnostics of §3.3. Each display states whether its rates are fractions or percentages; accuracy differences are in points, and a dash (–) marks a value that is not defined or not available.

Roles. Ratios between roles pair each workflow's decision role with its solving role: the aggregator with the generator under Voting, the evaluator with the generator under Eval-Opt, and the synthesizer with the worker under Orch-Workers. Under IP the three instances of a repeated role train one adapter and are measured as one role. The instances of an episode *disagree* when one or two of the three are correct; under Eval-Opt, when the first attempt and the final answer differ in correctness.

Graded outcomes. The *oracle* holds when at least one solving-role output of an episode is correct. An Eval-Opt episode is *lost* when its first attempt is correct and its final answer wrong. A wrong first attempt is *T-wrong* if it is truncated and *F-wrong* if it finishes under the cap. A role output is *rewarded* when its episode's final answer is correct, since the broadcast reward then gives every role of the episode reward 1. On a problem, *no instance solves* when none of the episode's three voters or workers is correct, and *SA-RL solves* when the SA-RL checkpoint of the same scale at the nearest sampled step is correct.

Training diagnostics. Each diagnostic is divided by its median over training steps 1–30. The *late ratio* is the mean over the last 30 training steps divided by that median, and the *peak ratio* is the maximum over the run divided by it. Under IP χ^2, perplexity and gradient norm are measured per adapter, under SP on the shared adapter. The gradient norm is that of one optimizer step, averaged over the optimizer steps of a training step. The Voting aggregator, the Orch-Workers synthesizer and, on most training steps, the Orch-Workers orchestrator fit in one mini-batch, so their χ^2 measures only the mismatch between the rollout and training engines. The *mismatch ratio* is the late ratio of that engine mismatch itself, measured before the update and pooled over the batch. A GRPO group has a non-zero advantage only if its rollouts include both correct and incorrect outcomes; we call such groups *mixed*.

Smoothed training series. The cap-hit rate is pooled over roles and smoothed with a 10-step centered mean; it *falls back* when it ends at least 35% below its maximum over steps up to 300. For the early-warning signals, the *onset* is the first step at which the trailing 10-step running median crosses the threshold and stays above it for 20 consecutive logged steps, or until the end of the run if at least 5 steps remain. The training-success rule smooths training success with a 15-step centered mean, fires when that mean stays at least 0.03 below its running maximum under the same persistence condition, and returns the last checkpoint at or before the step of that maximum. A signal's *lead* is the validation peak step minus its onset, positive when the onset precedes the peak; for

the training-success rule it is measured at the step of the running maximum, which precedes the step at which the rule fires. The online replay (Table 13) uses only steps already logged: the training-success mean is trailing, the end-of-run clause is dropped, an alarm is raised once the persistence is observed, and its lead is measured at the alarm.

Parameter change. N is the number of optimizer steps an adapter takes per training step, its sequences per training step divided by the fixed mini-batch size: three for a repeated role under IP, one for a single-instance IP adapter and for SA-RL, and four for the SP adapter under Voting and five under Orch-Workers. Under Eval-Opt it varies with the revision rounds: k, the mean number of generator turns per episode, for each IP adapter and $2k$ for the shared adapter. The *rate* of an adapter is the distance between its checkpoints 50 training steps apart divided by 50, in units of the learning rate times the square root of its parameter count. It is averaged over windows that start every 10 training steps from step 20 and end by step 300. The noise model is AdamW driven by independent Gaussian gradients with the same N; it has no free parameter and predicts both the rate and the cosine between consecutive 50-step changes.

Output statistics. The hedging-phrase rate is the fraction of outputs containing one of *wait*, *alternatively*, *actually*, *hmm*, *let me reconsider*, *on second thought*, *not correct* or *this is wrong*. Within-problem 3-gram Jaccard is the pairwise Jaccard similarity of the instances' 3-gram sets, averaged within a problem and then across problems.

Tests and horizons. Paired comparisons of two checkpoints use the two-sided exact McNemar test on per-problem outcomes, or a two-proportion test where per-problem outcomes are not available. Bootstrap intervals resample problems. Displays of `Eval-Opt-IP-1.7B-Math` end at step 379 (its last sampled checkpoint is at step 370), because its training rewards after that step were computed with an earlier version of the math grader.

B.6 Compute

All training runs use a single compute node, a frozen base model, LoRA adapters, and a vLLM rollout engine. The frozen model and rollout engine dominate total device memory, while adapter and optimizer states differ with the number of trainable adapters.

Under Isolated-Policy all role adapters stay in GPU memory, and each rollout request and gradient update uses its role's adapter (resource matching: §A.3).

Most runs train on two GPUs and some on four. Most 1.7B and 4B cells use H100 GPUs (80 GB) and most 0.6B and some 1.7B cells use L40S GPUs (48 GB). The summed wall-clock time of all runs is about 235 node-days.

C Additional Results

C.1 Selected Peak and Last Checkpoint per Cell

Fig. 6 plots each cell's gain over its base workflow for §4.1. Panel A of Table 5 lists every cell's selected peak and last validated checkpoint, for the post-peak comparisons of §4.3. Runs end at different steps, so the gap between peak and last is read within a run and is not compared across cells; where the two coincide, the run was at its best at its final validated checkpoint.

Panel B compares IP with SP and each strategy with SA-RL, for §4.1, §4.2, and §4.3, under four horizon rules. *Peak* uses the selected peaks of Panel A. *Capped* takes both runs' peaks up to the earlier of their last validated steps, and for IP against SP only among checkpoints at step ≤ 300. *Last common* compares the two runs at the last step validated in both. *Full* takes each run's peak over its whole logged horizon.

In the learning-speed comparison of Fig. 4 panel (b), a pair's common gain is the smaller of the two runs' validation gains within their shared horizon. Gains are measured on 3-point smoothed validation series, from the untrained workflow on Math and from the run's step-0 validation on Code (for `Eval-Opt-IP-4B-Code`, that of the SP run, which starts from the same untrained workflow). On the two Voting-4B pairs the SP gain, and hence the common gain, is small.

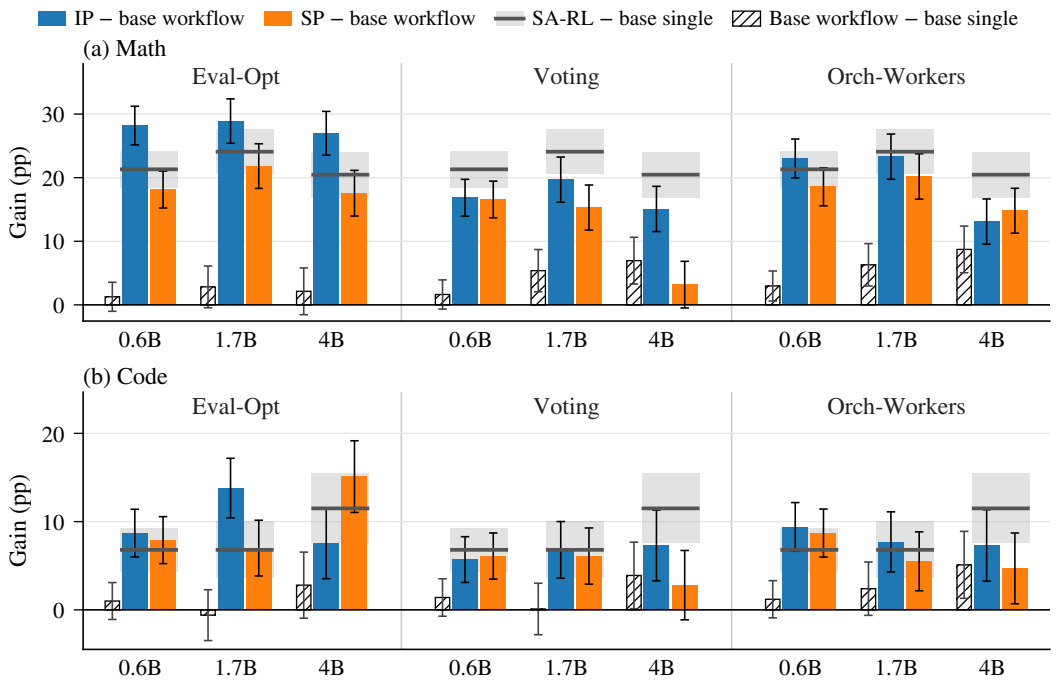

Figure 6: Gain of each cell over its base workflow on (a) Math and (b) Code. Bars: selected-peak accuracy of the IP and SP runs minus the untrained workflow. Gray line: SA-RL minus the untrained single model at the same task and scale, 95% interval shaded; a bar above it gains more over its base workflow than SA-RL gains over the untrained single model. Hatched bar: the untrained workflow minus the untrained single model. Error bars: 95% intervals of each difference, from the two accuracies' Wilson intervals combined in quadrature. All values are evaluated at the training length caps.

C.2 Solving Roles Against Single-Agent RL

Table 6 gives the per-checkpoint values for §4.1 and §4.3. For each Math cell and sampled checkpoint it sets the solving role's accuracy on its own answer against SA-RL's on the same problems and splits the workflow's accuracy into the solving role's accuracy and the structural share above it. Under Eval-Opt the oracle counts a correct answer at any generator turn. The rewarded share of truncated outputs is shown only for rows with at least five truncated outputs. In the SA-RL rows the reward is the grade of the policy's own answer.

Fig. 2(a) uses one checkpoint per cell, marked †: the sampled checkpoint nearest the selected peak, graded on all 1412 problems like the untrained row it is compared with. Against the untrained row on the same problems, the change in Solver is the solving-role gain and the change in Struct. the structural gain; the two sum to the change in Workflow.

The bottom panel gives each Math run's cap-hit rate over training as its maximum and its last value, and Fig. 7 shows the full series. The SA-RL runs' cap-hit rate rises and then falls back; among the multi-agent runs only `Voting-IP-1.7B-Math` falls back as far, and its per-voter-reward control falls back earlier.

C.3 Decision-Role Output Form and Per-Problem Effect

Table 7 extends Table 2 with the untrained workflow, the checkpoint steps and the remaining columns. Δ vs. plurality is workflow accuracy minus the expected accuracy of the plurality vote over the voters' boxed answers. Lone correct picked counts the episodes with one correct voter and two voters sharing a wrong answer in which the aggregator selects the correct voter; Follows 2–1 majority counts the episodes with a two-to-one split in which it selects a majority voter. The Eval-Opt columns count the evaluator's first verdicts that accept a T-wrong first attempt (Accepts T-wrong in Table 2), reject an

Table 5: Validation accuracy (%) at the selected peak and the last validated checkpoint at step ≤ 300 of every cell, training step in parentheses (Panel A), and differences in points (Panel B): IP minus SP and each strategy minus SA-RL under the horizon rules Peak, Capped, Last common (step in parentheses) and Full, defined in the text. Bold: capped IP minus SP gap with $p < 0.05$ (tests in §B.5).

Panel A. Selected peak and last checkpoint at step ≤ 300.

Task	Scale	Workflow	IP peak (step)	IP last (step)	SP peak (step)	SP last (step)
Math	0.6B	Eval-Opt	39.5 (230)	39.5 (230)	29.5 (220)	29.5 (220)
		Voting	28.5 (260)	25.8 (300)	28.3 (200)	28.3 (200)
		Orch-Workers	36.0 (200)	33.6 (230)	31.6 (180)	31.6 (180)
	1.7B	Eval-Opt	57.6 (260)	56.9 (300)	50.5 (290)	48.2 (300)
		Voting	50.9 (190)	40.7 (300)	46.5 (300)	46.5 (300)
		Orch-Workers	55.5 (290)	54.2 (300)	52.3 (300)	52.3 (300)
	4B	Eval-Opt	75.2 (230)	75.2 (230)	65.8 (200)	65.8 (210)
		Voting	68.1 (100)	54.9 (215)	56.2 (140)	55.7 (180)
		Orch-Workers	67.9 (100)	60.0 (180)	69.6 (190)	67.0 (220)
Code	0.6B	Eval-Opt	15.2 (170)	14.1 (240)	14.4 (270)	13.2 (300)
		Voting	12.6 (300)	12.6 (300)	13.0 (270)	12.0 (300)
		Orch-Workers	16.1 (270)	16.1 (270)	15.4 (250)	14.3 (265)
	1.7B	Eval-Opt	25.8 (300)	25.8 (300)	19.0 (250)	17.7 (300)
		Voting	19.5 (290)	19.1 (300)	18.8 (210)	18.7 (270)
		Orch-Workers	22.7 (280)	19.7 (300)	20.5 (170)	18.9 (190)
	4B	Eval-Opt	33.0 (140)	30.4 (160)	40.6 (230)	39.1 (260)
		Voting	33.9 (220)	31.0 (240)	29.4 (110)	29.4 (110)
		Orch-Workers	35.1 (190)	34.1 (220)	32.5 (110)	31.7 (160)

Panel B. IP minus SP, and each strategy minus SA-RL, at the selected peak and under three other horizon rules (points).

			IP − SP			IP − SA-RL			SP − SA-RL		
Task	Scale	Workflow	Peak	Capped	Last common	Peak	Capped	Full	Peak	Capped	Full
Math	0.6B	Eval-Opt	+10.1	**+8.6**	+8.6 (220)	+8.2	+11.0	+8.2	−1.9	+1.0	−1.9
		Voting	+0.3	−2.0	−2.3 (200)	−2.8	−2.8	−2.8	−3.1	+2.1	−3.1
		Orch-Workers	+4.5	+2.3	+1.1 (180)	+4.7	+7.6	+4.7	+0.2	+5.5	+0.2
	1.7B	Eval-Opt	+7.1	**+7.1**	+2.0 (370)	+7.6	+4.2	+3.9	+0.6	+0.1	−0.3
		Voting	+4.4	**+4.4**	−5.8 (300)	+1.0	+1.0	−3.0	−3.4	−3.4	−7.4
		Orch-Workers	+3.1	+3.1	−1.6 (340)	+5.5	+1.9	+1.5	+2.4	−0.8	−1.2
	4B	Eval-Opt	+9.4	**+8.6**	+7.8 (210)	+8.6	+10.6	+8.6	−0.8	+1.1	−0.8
		Voting	+11.9	**+11.9**	+6.3 (180)	+1.6	+3.5	+1.6	−10.3	−7.9	−10.3
		Orch-Workers	−1.7	−1.1	−9.1 (180)	+1.4	+3.8	+1.4	+3.1	+5.0	+3.1
Code	0.6B	Eval-Opt	+0.8	+1.5	+0.4 (240)	+2.9	+2.9	+1.4	+2.1	+2.2	+2.2
		Voting	−0.4	−0.4	+0.6 (310)	+0.3	−0.1	−0.1	+0.7	+0.7	−0.8
		Orch-Workers	+0.7	+0.6	+1.4 (260)	+3.8	+3.8	+2.3	+3.1	+3.1	+1.6
	1.7B	Eval-Opt	+6.8	**+6.8**	+7.0 (301)	+6.4	+6.4	+6.4	−0.4	−0.4	−0.4
		Voting	+0.7	+0.2	−0.1 (270)	+0.1	+0.1	+0.1	−0.6	−0.5	−0.6
		Orch-Workers	+2.2	+1.0	+2.4 (190)	+3.3	+3.3	+3.3	+1.1	+2.3	+1.1
	4B	Eval-Opt	−7.6	−1.5	−4.1 (160)	−1.2	+0.6	−1.2	+6.4	+6.6	+6.4
		Voting	+4.5	+0.1	+0.1 (110)	−0.3	−0.1	−0.3	−4.8	−0.9	−4.8
		Orch-Workers	+2.6	+1.3	+1.7 (160)	+0.9	+1.9	+0.9	−1.7	+0.1	−1.7

F-wrong one, or accept a finished correct one. Answer-only is the share of synthesizer outputs of at most 40 tokens, and Equals plurality counts, among answer-only outputs whose workers have a unique most common answer, those that return it. Loss and Rescue are the shares of episodes with a selection loss and with a rescue.

Both rules hold beyond the checkpoints shown. Over every trained Voting checkpoint graded on all 1412 problems, Δ vs. plurality stays within a few points of zero, and its 95% bootstrap interval contains zero at 6 of 11. It excludes zero at one of the three checkpoints at 1.7B, below plurality, and

Table 6: Solving roles trained inside each workflow against SA-RL on Math, per sampled checkpoint, on all 1412 problems (*: row on the 100 sampled problems). Solver: the solving role's accuracy on its own answer; SA-RL: single-agent accuracy on the same problems (step); Diff.: Solver minus SA-RL; Struct.: Workflow minus Solver. Rew. wrong and Rew. | wrong: rewarded-but-wrong outputs over all and over wrong solving-role outputs; Trunc.: truncated share; Rew. | trunc.: rewarded share of truncated outputs. Accuracies and shares are fractions. Base: untrained workflow; †: checkpoint of Fig. 2(a). Bottom: training cap-hit rate (Fig. 7), maximum (step) / last value; bold: falls back. Definitions in §B.5.

Scale	Strategy	Step	Solver	SA-RL (step)	Diff. (pp)	Oracle	Workflow	Struct.	Rew. wrong	Rew. | wrong	Trunc.	Rew. | trunc.
Voting: mean voter												
0.6B	Base	0	0.10	–	–	0.20	0.10	−0.00	0.04	0.04	0.00	0.00
	IP	100*	0.17	0.21 (100)	−4	0.30	0.20	0.03	0.07	0.08	0.17	0.10
	IP	200*	0.16	0.25 (200)	−9	0.28	0.20	0.04	0.07	0.08	0.66	0.07
	IP	260†	0.21	0.29 (260)	−7	0.33	0.29	0.08	0.09	0.12	0.59	0.11
	IP	300	0.20	0.30 (300)	−10	0.32	0.26	0.07	0.09	0.11	0.44	0.12
	SP	50	0.12	0.14 (50)	−2	0.24	0.13	0.01	0.05	0.06	0.00	0.00
	SP	90*	0.13	0.21 (100)	−8	0.23	0.12	−0.01	0.03	0.03	0.03	0.11
	SP	200†	0.22	0.26 (200)	−4	0.36	0.28	0.06	0.09	0.11	0.47	0.09
1.7B	Base	0	0.27	–	–	0.44	0.28	0.01	0.08	0.10	0.03	0.10
	IP	70*	0.28	0.40 (100)	−12	0.40	0.30	0.02	0.06	0.08	0.09	0.14
	IP	130*	0.36	0.40 (100)	−4	0.52	0.40	0.04	0.09	0.15	0.24	0.17
	IP	190†	0.43	0.45 (190)	−2	0.59	0.51	0.08	0.11	0.19	0.32	0.19
	IP	290*	0.29	0.43 (300)	−14	0.41	0.33	0.04	0.07	0.10	0.28	0.09
	IP	300	0.32	0.50 (300)	−17	0.46	0.40	0.08	0.10	0.15	0.47	0.14
	SP	120*	0.30	0.40 (100)	−10	0.47	0.35	0.05	0.09	0.13	0.05	0.00
	SP	230*	0.35	0.42 (200)	−7	0.54	0.42	0.07	0.11	0.17	0.11	0.12
	SP	250*	0.32	0.42 (200)	−10	0.48	0.37	0.05	0.09	0.13	0.14	0.17
	SP	300†	0.38	0.50 (300)	−11	0.56	0.46	0.08	0.11	0.18	0.16	0.21
4B	Base	0	0.47	–	–	0.64	0.52	0.05	0.10	0.18	0.04	0.21
	IP	70*	0.49	0.49 (100)	0	0.60	0.56	0.07	0.09	0.17	0.21	0.16
	IP	100†	0.58	0.60 (100)	−2	0.73	0.66	0.08	0.11	0.25	0.20	0.27
	IP	130*	0.50	0.49 (100)	+1	0.63	0.58	0.08	0.10	0.19	0.30	0.18
	IP	195	0.50	0.63 (190)	−14	0.66	0.60	0.10	0.12	0.25	0.22	0.25
	IP	200*	0.42	0.56 (200)	−14	0.55	0.47	0.05	0.08	0.14	0.26	0.15
	IP	215	0.45	0.63 (210)	−18	0.61	0.55	0.10	0.12	0.22	0.29	0.21
	SP	70*	0.43	0.49 (100)	−6	0.55	0.46	0.03	0.07	0.13	0.05	0.00
	SP	140†	0.48	0.62 (140)	−14	0.65	0.56	0.08	0.11	0.21	0.05	0.22
	SP	170	0.49	0.64 (170)	−15	0.65	0.55	0.06	0.10	0.20	0.05	0.19
Orch-Workers: mean worker												
0.6B	Base	0	0.10	–	–	0.22	0.12	0.02	0.06	0.06	0.00	0.13
	IP	90*	0.17	0.21 (100)	−4	0.33	0.19	0.02	0.07	0.09	0.05	0.00
	IP	180*	0.15	0.25 (200)	−10	0.23	0.26	0.11	0.12	0.14	0.62	0.13
	IP	190	0.17	0.26 (190)	−9	0.28	0.33	0.16	0.16	0.20	0.66	0.21
	IP	200†	0.16	0.26 (200)	−10	0.26	0.34	0.18	0.19	0.22	0.73	0.22
	SP	40*	0.13	0.21 (100)	−8	0.26	0.18	0.05	0.08	0.09	0.02	0.20
	SP	70*	0.13	0.21 (100)	−8	0.25	0.13	0.00	0.06	0.07	0.03	0.00
	SP	110*	0.19	0.21 (100)	−2	0.36	0.25	0.06	0.10	0.12	0.12	0.14
	SP	180†	0.16	0.26 (180)	−10	0.26	0.30	0.14	0.15	0.18	0.70	0.18
1.7B	Base	0	0.26	–	–	0.44	0.34	0.08	0.12	0.16	0.03	0.17
	IP	140*	0.33	0.40 (100)	−7	0.50	0.42	0.09	0.12	0.18	0.13	0.20
	IP	280*	0.34	0.43 (300)	−9	0.50	0.49	0.15	0.17	0.26	0.45	0.29
	IP	290†	0.37	0.50 (290)	−13	0.51	0.56	0.19	0.20	0.32	0.49	0.34
	IP	330	0.29	0.52 (330)	−23	0.40	0.53	0.24	0.24	0.34	0.64	0.35
	SP	210*	0.29	0.42 (200)	−13	0.45	0.39	0.10	0.12	0.17	0.19	0.21
	SP	300†	0.36	0.50 (300)	−14	0.53	0.51	0.15	0.17	0.27	0.40	0.28
	SP	330	0.36	0.52 (330)	−16	0.53	0.53	0.17	0.18	0.29	0.45	0.30
	SP	340	0.34	0.54 (340)	−20	0.50	0.53	0.19	0.20	0.30	0.50	0.33

Scale	Strategy	Step	Solver	SA-RL (step)	Diff. (pp)	Oracle	Workflow	Struct.	Rew. wrong	Rew. | wrong	Trunc.	Rew. | trunc.
Orch-Workers: mean worker												
4B	Base	0	0.45	–	–	0.63	0.54	0.09	0.13	0.23	0.04	0.23
	IP	60*	0.44	0.49 (100)	−5	0.60	0.51	0.07	0.10	0.18	0.13	0.24
	IP	100†	0.56	0.60 (100)	−4	0.73	0.68	0.12	0.14	0.32	0.20	0.36
	IP	120*	0.48	0.49 (100)	−1	0.65	0.60	0.12	0.14	0.28	0.34	0.30
	IP	180	0.34	0.64 (180)	−30	0.47	0.61	0.27	0.27	0.42	0.62	0.43
	SP	70*	0.40	0.49 (100)	−9	0.58	0.48	0.08	0.12	0.19	0.08	0.12
	SP	130*	0.45	0.49 (100)	−4	0.62	0.56	0.11	0.13	0.24	0.18	0.16
	SP	170*	0.44	0.56 (200)	−12	0.60	0.59	0.15	0.16	0.29	0.34	0.29
	SP	190†	0.52	0.63 (190)	−11	0.68	0.70	0.18	0.19	0.40	0.33	0.40
	SP	220	0.45	0.64 (220)	−19	0.60	0.68	0.23	0.23	0.42	0.47	0.43
Eval-Opt: first attempt												
0.6B	Base	0	0.10	–	–	0.11	0.10	0.00	0.00	0.00	0.00	0.00
	IP	100	0.21	0.23 (100)	−1	0.30	0.29	0.08	0.08	0.10	0.51	0.14
	IP	190	0.14	0.26 (190)	−12	0.37	0.36	0.22	0.22	0.26	0.82	0.27
	IP	200	0.12	0.26 (200)	−14	0.37	0.37	0.25	0.25	0.28	0.86	0.30
	IP	230†	0.10	0.27 (230)	−17	0.39	0.39	0.29	0.29	0.32	0.90	0.33
	SP	110*	0.16	0.21 (100)	−5	0.17	0.16	0.00	0.00	0.00	0.20	0.00
	SP	210	0.18	0.27 (210)	−8	0.29	0.29	0.10	0.10	0.13	0.65	0.16
	SP	220†	0.17	0.28 (220)	−11	0.29	0.29	0.12	0.12	0.15	0.72	0.18
1.7B	Base	0	0.27	–	–	0.27	0.27	0.00	0.00	0.00	0.03	0.00
	IP	160*	0.43	0.42 (200)	+1	0.54	0.53	0.10	0.11	0.19	0.37	0.27
	IP	260†	0.38	0.48 (260)	−10	0.59	0.59	0.21	0.21	0.33	0.58	0.37
	IP	300*	0.37	0.43 (300)	−6	0.54	0.54	0.17	0.17	0.27	0.54	0.31
	IP	320	0.41	0.50 (320)	−10	0.56	0.55	0.15	0.15	0.25	0.52	0.29
	IP	340*	0.34	0.43 (300)	−9	0.50	0.49	0.15	0.15	0.23	0.60	0.25
	IP	360*	0.29	0.43 (300)	−14	0.43	0.43	0.14	0.14	0.20	0.66	0.23
	IP	370	0.39	0.53 (370)	−13	0.55	0.55	0.15	0.15	0.25	0.54	0.29
	SP	150*	0.30	0.40 (100)	−10	0.34	0.33	0.03	0.04	0.06	0.26	0.15
	SP	290†	0.40	0.50 (290)	−10	0.49	0.48	0.09	0.09	0.14	0.42	0.20
	SP	320	0.39	0.50 (320)	−11	0.56	0.54	0.15	0.16	0.25	0.51	0.31
4B	Base	0	0.47	–	–	0.52	0.48	0.01	0.03	0.06	0.04	0.10
	IP	90*	0.56	0.49 (100)	+7	0.65	0.63	0.07	0.08	0.18	0.31	0.26
	IP	180*	0.44	0.56 (200)	−12	0.71	0.71	0.27	0.27	0.48	0.54	0.50
	IP	220†	0.40	0.64 (220)	−24	0.76	0.75	0.35	0.36	0.59	0.60	0.60
	SP	105*	0.57	0.49 (100)	+8	0.63	0.59	0.02	0.06	0.14	0.10	0.00
	SP	210*	0.52	0.56 (200)	−4	0.61	0.61	0.09	0.09	0.19	0.37	0.27
	SP	215†	0.59	0.63 (210)	−3	0.68	0.66	0.07	0.08	0.19	0.27	0.26
SA-RL: the single output												
0.6B	SA-RL	100*	0.21	–	–	–	–	–	0.00	0.00	0.46	0.00
	SA-RL	200*	0.25	–	–	–	–	–	0.00	0.00	0.53	0.00
	SA-RL	300*	0.32	–	–	–	–	–	0.00	0.00	0.25	0.00
1.7B	SA-RL	100	0.38	–	–	–	–	–	0.00	0.00	0.26	0.00
	SA-RL	200*	0.42	–	–	–	–	–	0.00	0.00	0.37	0.00
	SA-RL	300*	0.43	–	–	–	–	–	0.00	0.00	0.34	0.00
	SA-RL	430	0.52	–	–	–	–	–	0.00	0.00	0.16	0.00
4B	SA-RL	100*	0.49	–	–	–	–	–	0.00	0.00	0.33	0.00
	SA-RL	200*	0.56	–	–	–	–	–	0.00	0.00	0.29	0.00
	SA-RL	300	0.67	–	–	–	–	–	0.00	0.00	0.22	0.00

Cap-hit rate over training, Math: maximum (step) / last value

Scale	SA-RL	Eval-Opt IP	Eval-Opt SP	Voting IP	Voting SP	Orch-Workers IP	Orch-Workers SP
0.6B	**0.62 (143) / 0.26**	0.34 (230) / 0.34	0.38 (228) / 0.38	0.52 (249) / 0.37	0.38 (201) / 0.38	0.56 (239) / 0.56	0.47 (182) / 0.47
1.7B	**0.45 (143) / 0.24**	0.24 (276) / 0.20	0.20 (292) / 0.19	**0.34 (254) / 0.17**	0.12 (287) / 0.09	0.34 (300) / 0.34	0.22 (287) / 0.20
4B	**0.30 (230) / 0.18**	0.26 (235) / 0.26	0.08 (207) / 0.07	0.21 (216) / 0.21	0.03 (121) / 0.02	0.41 (173) / 0.40	0.33 (221) / 0.32

Per-voter-reward control of `Voting-IP-1.7B-Math`: **0.24 (201) / 0.15**

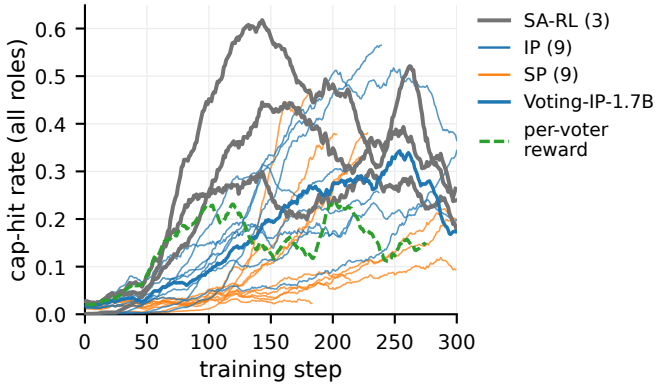

Figure 7: Cap-hit rate over training on Math for every SA-RL, IP and SP run, pooled over roles and smoothed (§B.5). Thick blue: `Voting-IP-1.7B-Math`; green, dashed: its per-voter-reward control, in which each voter is rewarded for its own answer. Maximum and last values: Table 6, bottom.

Table 7: Decision-role behavior on Math, full version of Table 2: the untrained workflow (Base) and the latest sampled IP and SP checkpoints (steps in the last column), on all 1412 validation problems. k/n: count and denominator; –: empty denominator. Voting: workflow minus plurality-vote accuracy and the aggregator's selections. Eval-Opt: the evaluator's first verdict. Orch-Workers: answer-only synthesizer outputs and episode shares. Column definitions: §C.3.

Voting aggregator

	Δ vs. plurality (pp)			Lone correct picked			Follows 2–1 majority			Steps
Scale	Base	IP	SP	Base	IP	SP	Base	IP	SP	IP / SP
0.6B	−1.5	+0.4	−0.1	11/28	0/15	2/25	228/335	153/161	169/179	300 / 200
1.7B	−2.4	+0.5	+0.3	16/56	4/21	3/40	304/414	177/199	327/351	300 / 300
4B	+0.3	+3.1	+1.7	14/52	5/24	15/46	337/403	294/317	319/373	215 / 170

Eval-Opt evaluator

	Accepts truncated wrong			Rejects finished wrong			Accepts finished correct			Steps
Scale	Base	IP	SP	Base	IP	SP	Base	IP	SP	IP / SP
0.6B	5/7	0/1257	24/1007	114/1259	3/16	2/162	136/146	108/120	234/235	230 / 220
1.7B	49/49	73/756	28/714	22/987	2/99	12/145	376/376	550/550	515/544	370 / 320
4B	0/52	0/837	3/373	464/692	1/11	115/200	531/668	543/548	362/836	220 / 215

Orch-Workers synthesizer

	Answer-only			Equals plurality			Loss			Rescue			Steps
Scale	Base	IP	SP	Base	IP	SP	Base	IP	SP	Base	IP	SP	IP / SP
0.6B	0.03	0.06	0.41	16/17	71/73	425/431	0.10	0.02	0.03	0.00	0.10	0.06	200 / 180
1.7B	0.00	0.39	0.35	2/2	536/538	484/485	0.11	0.01	0.04	0.01	0.14	0.07	330 / 340
4B	0.00	0.41	0.55	–	519/523	709/717	0.09	0.01	0.02	0.00	0.16	0.09	180 / 220

at four of the five at 4B, all above plurality. Over every trained Orch-Workers checkpoint graded on all 1412 problems, 5,425 of the 5,460 answer-only outputs whose workers have a unique most common answer return it.

Table 8 gives the output form behind the decision-role analysis of §4.1 (Panel A) and, for seven SP cells, the answers the decision role itself loses or supplies (Panel B). A short selection is an aggregator output of at most 30 tokens with a boxed selection, and a code block is a fenced Python block of at least 3 lines. A long-form evaluator turn exceeds 500 tokens, and a short verdict is a turn under 200 tokens with a boxed Correct or Incorrect verdict and no code block; Eval-Opt Math rows use every evaluator turn and Code rows the first. The orchestrator writes the plan and is not a decision role. Its example-title rate is the share of its strategy titles that begin with one of the four example

techniques named in its prompt, and its top-plan share is the share of problems whose three titles equal the checkpoint's most common triple.

On Eval-Opt the decision role *loses* an answer when a generator answer is correct, the evaluator does not accept it, and the final answer is wrong. On Orch-Workers it loses one when a worker's answer is correct and the synthesizer's is wrong, and on Voting when a voter's answer is correct and the aggregator selects a wrong one. It *supplies* an answer on Eval-Opt Math when the final answer is correct and the evaluator stated it before any generator turn did. On Eval-Opt Code it supplies one when the first generator program fails and both the evaluator's program and the final program pass, and on Orch-Workers when no worker's answer is correct and the synthesizer's is. The Voting aggregator only selects, so it supplies none. The `Voting-SP-4B-Math` aggregator writes either a short boxed selection or a long review, and the share of short selections changes between its sampled checkpoints while its validation accuracy stays flat (Panels A and B).

C.4 Code Tasks

Table 10 gives the values for the Code analysis of §4.1, comparing how the broadcast reward treats the solving-role outputs on the two tasks. On Code, truncated outputs fall almost only on problems that no instance of the episode solves and SA-RL rarely solves, so they are rarely rewarded, and rewarded-but-wrong outputs are less than half as frequent as on Math in every workflow. None of the 5,312 truncated Code solving-role outputs is correct, SA-RL included. The solving roles stay close to SA-RL at 0.6B and 1.7B and trail it by a few points at 4B, where all the Code checkpoints whose truncated outputs are most often rewarded lie (Panel B). One of the 53 trained Code checkpoints trails SA-RL by more than 10 points, against 25 of the 65 on Math.

Table 9 compares the two tasks on training signal, per-role χ^2 and gradient norms (Panel A). On Code fewer rollout groups are mixed than on Math at every workflow and scale, so fewer groups carry a non-zero advantage. Under IP the repeated role's adapter takes as many optimizer steps per training step as on Math, a number set by the role multiplicities and the fixed mini-batch size. Its χ^2 still rises far less, and no IP Code run declines significantly after its peak (§4.4). The voters of an episode disagree on correctness far less often than on Math (Table 10). At the 18 sampled trained Voting checkpoints on Code they disagree on 3 to 18 of the 100 problems, against 17 to 38 at the 25 on Math. The oracle exceeds the mean voter by 1 to 10 points on Code and by 9 to 19 on Math. The decision role loses few correct answers below 4B (Table 9, Panel B). In SA-RL as in the workflows, the Code solving roles' outputs lengthen, reach the length cap more often and carry a larger share of comment lines (Panel B).

Unclosed code blocks. In `Orch-Workers-IP-0.6B-Code` († in both tables) the workers leave their code blocks unclosed, which the grader scores as wrong, so no worker output is graded correct. Of the 900 worker outputs at its three sampled checkpoints, 372 stop without closing the block and 520 reach the cap. The synthesizer closes its block whenever it finishes (236 of the 300 episodes) and 32 episodes are correct, so the episode reward does not penalize the omission. With the closing fence appended, 72 worker outputs are correct, 8.0% against 8.3% for SA-RL at the nearest steps. The Orch-Workers mean difference from SA-RL at 0.6B then becomes -0.4 points, and the synthesizer loses at most 3 answers. Without this cell, the Code Orch-Workers column has 4.5% rewarded-but-wrong outputs and a mean difference of -1.7 points.

Columns. Code responses are capped at 2,048 tokens. In Table 9, Panel A averages the gradient norms over training steps up to 300. In Panel B, the comment share is the fraction of non-empty lines inside an output's code blocks that are comments, averaged over outputs with a code block. *Answers lost* is the largest number of the 100 problems, at any sampled checkpoint of the cell, with a selection loss, or under Eval-Opt with a correct first attempt and a wrong final answer. In Table 10, the difference from SA-RL is a checkpoint's solving-role accuracy minus that of the SA-RL checkpoint on the same problems, averaged over all checkpoints of a task and workflow or over those of one scale. Under Eval-Opt and SA-RL a dash marks a row that needs several graded solving-role outputs per episode, and in the SA-RL column also a comparison of SA-RL with itself.

Table 8: Decision-role output form (Panel A) and per-problem effect (Panel B) on the sampled problems. Panel A: the base model and the earliest and latest sampled IP and SP checkpoints (steps in the last two columns), with the Orch-Workers orchestrator for comparison. Panel B: SP cells; validation accuracy at the peak and the last evaluated checkpoint of the full validated horizon, the exact McNemar p between them where per-problem outcomes exist, and, on the sampled problems, the correct final answers and the answers the decision role loses or supplies (n/a: Voting). Definitions: §C.3.

Panel A. Decision-role output form at base, IP and SP.

		IP		SP		Steps	
Scale	Base	earliest	latest	earliest	latest	IP	SP
Voting, Math: aggregator; short-selection rate / mean tokens							
0.6B	0.02 / 360	0.00 / 292	0.64 / 147	0.00 / 432	0.00 / 476	100, 300	50, 200
1.7B	0.00 / 343	0.01 / 128	0.00 / 210	0.00 / 204	0.00 / 162	70, 290	120, 250
4B	0.28 / 281	1.00 / 6	1.00 / 6	0.61 / 219	0.49 / 287	70, 200	70, 170
Voting, Code: aggregator; code-block rate / mean tokens							
0.6B	0.98 / 246	0.02 / 74	0.99 / 348	0.92 / 384	1.00 / 772	95, 320	135, 280
1.7B	0.94 / 569	0.00 / 6	0.00 / 6	0.94 / 943	0.97 / 856	225, 265	185, 225
4B	0.05 / 88	0.36 / 822	0.69 / 1037	0.03 / 425	0.04 / 438	100, 240	110, 120
Eval-Opt, Math: evaluator; long-form rate / median tokens							
0.6B	0.00 / 6	0.02 / 108	0.39 / 349	0.00 / 78	0.41 / 324	100, 200	110, 220
1.7B	0.00 / 6	0.08 / 223	0.40 / 412	0.02 / 131	0.63 / 986	160, 370	150, 320
4B	0.00 / 108	0.00 / 6	0.00 / 110	0.01 / 109	0.06 / 158	90, 220	105, 215
Eval-Opt, Code: evaluator; code-block rate / short-verdict rate / median tokens							
0.6B	0.00 / 0.86 / 6	0.00 / 1.00 / 6	0.00 / 0.00 / 2048	0.91 / 0.05 / 570	0.99 / 0.00 / 2048	90, 240	200, 445
1.7B	0.02 / 0.45 / 208	0.02 / 0.00 / 332	0.38 / 0.01 / 410	0.00 / 0.90 / 90	0.01 / 0.44 / 219	150, 350	130, 300
4B	0.00 / 0.03 / 338	0.00 / 0.13 / 325	0.03 / 0.11 / 338	0.01 / 0.06 / 402	0.08 / 0.00 / 543	70, 150	115, 265
Orch-Workers, Math: synthesizer; answer-only rate / median tokens							
0.6B	0.02 / 292	0.01 / 266	0.04 / 316	0.00 / 772	0.00 / 598	90, 190	40, 110
1.7B	0.01 / 506	0.34 / 97	0.41 / 298	0.33 / 315	0.30 / 1062	140, 330	210, 340
4B	0.00 / 494	0.00 / 458	0.34 / 196	0.01 / 370	0.77 / 19	60, 180	70, 170
Orch-Workers, Math: orchestrator; example-title rate / top-plan share / median tokens							
0.6B	0.28 / 0.10 / 209	0.03 / 0.02 / 78	0.98 / 0.55 / 27	0.04 / 0.01 / 175	0.03 / 0.01 / 118	90, 190	40, 110
1.7B	0.30 / 0.02 / 252	0.23 / 0.01 / 257	0.49 / 0.01 / 208	0.73 / 0.13 / 216	0.83 / 0.21 / 134	140, 330	210, 340
4B	0.26 / 0.01 / 228	0.24 / 0.01 / 298	0.17 / 0.01 / 300	0.22 / 0.01 / 212	0.17 / 0.01 / 224	60, 180	70, 170
Orch-Workers, Code: synthesizer; code-block rate / median tokens							
0.6B	0.99 / 222	1.00 / 258	1.00 / 850	1.00 / 286	0.99 / 2048	100, 275	125, 265
1.7B	1.00 / 344	0.99 / 325	1.00 / 622	1.00 / 377	1.00 / 390	145, 300	90, 180
4B	0.99 / 314	0.98 / 582	0.99 / 783	1.00 / 352	1.00 / 390	190, 230	55, 160

Panel B. SP cells: validation accuracy and the decision role's per-problem effect on the sampled problems.

	Validation accuracy			Sampled problems (of 100)			
Cell	peak (step)	last (step)	McNemar p	steps	correct	lost	supplied
`Eval-Opt-SP-1.7B-Math`	0.536 (360)	0.515 (370)	0.11	150, 290, 320	33, 41, 44	1, 1, 3	0, 4, 10
`Eval-Opt-SP-4B-Math`	0.658 (200)	0.658 (210)	1.00	105, 210, 215	59, 61, 61	4, 0, 1	0, 1, 0
`Orch-Workers-SP-1.7B-Math`	0.527 (330)	0.526 (340)	1.00	210, 330, 340	39, 48, 47	7, 3, 3	1, 7, 5
`Orch-Workers-SP-4B-Math`	0.696 (190)	0.670 (220)	0.008	70, 130, 170	48, 56, 59	10, 6, 3	0, 0, 2
`Voting-SP-4B-Math`	0.562 (140)	0.557 (180)	0.66	70, 140, 170	46, 50, 48	10, 9, 8	n/a
`Eval-Opt-SP-0.6B-Code`	0.160 (390)	0.151 (440)	—	200, 390, 445	10, 14, 13	0, 0, 0	2, 4, 4
`Voting-SP-0.6B-Code`	0.130 (270)	0.115 (310)	—	135, 270, 280	9, 9, 11	2, 2, 4	n/a

C.5 Per-Turn Decomposition of Eval-Opt

Table 11 gives the per-checkpoint values behind Figs. 3 and 8 and the revision-loop analysis of §4.2. An episode is *lost* when its first attempt is correct and its final answer wrong, so the loop's net contribution equals recovered minus lost episodes. The eval. column counts the recovered episodes in which the evaluator stated the correct answer before the generator did (Math) or wrote a correct program (Code). The evaluator columns describe its first verdict; an unparsable verdict counts as a rejection, as it does in the workflow. Tokens are the generator's, summed over all turns of an episode. The bottom panel splits the IP minus SP difference on Math into the first attempt and the loop's net contribution, at the latest sampled SP checkpoint and the nearest sampled IP checkpoint. It also gives the share of T-wrong first attempts recovered under each strategy.

Table 9: Training signal, per-role χ^2 and gradient norms (Panel A), and Code solving-role outputs (Panel B). Panel A: mixed rollout groups, averaged over training; late χ^2 ratio of the repeated role and the largest single-instance one (agg. aggregator, orch. orchestrator, synth. synthesizer, gen. generator, eval. evaluator; none: no repeated role); decision-over-solving gradient norm. Panel B: the solving role (voter, worker, first attempt, SA-RL output), untrained workflow (SA-RL: earliest sampled checkpoint) $\rightarrow$ last sampled checkpoint, and the decision role's largest number of answers lost (none: no decision role). Definitions in the text and §B.5. †Workers leave their code blocks unclosed; see §C.4.

Panel A. Mixed-outcome rollout groups, per-role χ^2 and gradient norms, Math and Code.

	Math					Code				
	Mixed-outcome groups		Repeated role	Single-instance roles	Grad. norm	Mixed-outcome groups		Repeated role	Single-instance roles	Grad. norm
Workflow and scale	IP	SP	χ^2 ratio (IP)	largest χ^2 ratio (IP)	decision / solving (IP)	IP	SP	χ^2 ratio (IP)	largest χ^2 ratio (IP)	decision / solving (IP)
Voting-0.6B	0.39	0.40	5.08	agg. 1.03	5.2	0.17	0.16	1.85	agg. 1.02	3.1
Voting-1.7B	0.48	0.50	22.39	agg. 0.76	11.3	0.20	0.19	0.73	agg. 0.45	9.7
Voting-4B	0.40	0.49	11.05	agg. 0.45	63.0	0.29	0.26	0.95	agg. 2.38	1.0
Orch-Workers-0.6B	0.43	0.41	3.40	synth. 1.01	3.9	0.19	0.19	2.05	synth. 2.63	1.1
Orch-Workers-1.7B	0.48	0.50	12.85	orch. 1.29	8.0	0.25	0.25	0.58	synth. 5.81	0.7
Orch-Workers-4B	0.42	0.43	9.69	orch. 1.05	6.8	0.32	0.32	0.61	synth. 12.12	0.8
Eval-Opt-0.6B	0.48	0.45	none	gen. 2.41	8.6	0.20	0.19	none	gen. 1.44	2.9
Eval-Opt-1.7B	0.53	0.57	none	eval. 1.50	9.7	0.26	0.21	none	gen. 1.26	1.0
Eval-Opt-4B	0.42	0.52	none	gen. 3.18	14.8	0.31	0.34	none	gen. 1.52	1.1

Panel B. Code: solving-role outputs at the untrained workflow and the last sampled checkpoint, and the decision role's largest loss.

Cell	Sampled steps	Comment share	Cap-hit rate	Mean tokens	Answers lost
Voting-IP-0.6B	95, 190, 280, 320	0.22 $\rightarrow$ 0.71	0.01 $\rightarrow$ 0.61	266 $\rightarrow$ 1598	3
Voting-SP-0.6B	135, 270, 280	0.22 $\rightarrow$ 0.70	0.01 $\rightarrow$ 0.55	266 $\rightarrow$ 1448	4
Voting-IP-1.7B	225, 250, 265	0.35 $\rightarrow$ 0.60	0.09 $\rightarrow$ 0.48	592 $\rightarrow$ 1256	3
Voting-SP-1.7B	185, 210, 225	0.35 $\rightarrow$ 0.50	0.09 $\rightarrow$ 0.30	592 $\rightarrow$ 921	2
Voting-IP-4B	100, 220, 240	0.33 $\rightarrow$ 0.55	0.11 $\rightarrow$ 0.42	606 $\rightarrow$ 1177	10
Voting-SP-4B	110, 120	0.33 $\rightarrow$ 0.42	0.11 $\rightarrow$ 0.22	606 $\rightarrow$ 838	4
Orch-Workers-IP-0.6B	100, 270, 275	0.21 $\rightarrow$ 0.74	0.00 $\rightarrow$ 0.74	268 $\rightarrow$ 1665	0†
Orch-Workers-SP-0.6B	125, 250, 265	0.21 $\rightarrow$ 0.72	0.00 $\rightarrow$ 0.79	268 $\rightarrow$ 1709	4
Orch-Workers-IP-1.7B	145, 280, 300	0.31 $\rightarrow$ 0.33	0.10 $\rightarrow$ 0.15	548 $\rightarrow$ 658	5
Orch-Workers-SP-1.7B	90, 170, 180	0.31 $\rightarrow$ 0.53	0.10 $\rightarrow$ 0.38	548 $\rightarrow$ 1082	5
Orch-Workers-IP-4B	190, 230	0.29 $\rightarrow$ 0.38	0.08 $\rightarrow$ 0.22	533 $\rightarrow$ 774	12
Orch-Workers-SP-4B	55, 110, 160	0.29 $\rightarrow$ 0.43	0.08 $\rightarrow$ 0.24	533 $\rightarrow$ 843	7
Eval-Opt-IP-0.6B	90, 170, 240	0.24 $\rightarrow$ 0.70	0.02 $\rightarrow$ 0.82	308 $\rightarrow$ 1825	0
Eval-Opt-SP-0.6B	200, 390, 445	0.24 $\rightarrow$ 0.58	0.02 $\rightarrow$ 0.87	308 $\rightarrow$ 1882	0
Eval-Opt-IP-1.7B	150, 300, 350	0.36 $\rightarrow$ 0.78	0.10 $\rightarrow$ 0.82	573 $\rightarrow$ 1778	0
Eval-Opt-SP-1.7B	130, 250, 300	0.36 $\rightarrow$ 0.54	0.10 $\rightarrow$ 0.37	573 $\rightarrow$ 1192	0
Eval-Opt-IP-4B	70, 140, 150	0.34 $\rightarrow$ 0.49	0.14 $\rightarrow$ 0.33	632 $\rightarrow$ 1051	7
Eval-Opt-SP-4B	115, 230, 265	0.34 $\rightarrow$ 0.73	0.14 $\rightarrow$ 0.70	632 $\rightarrow$ 1638	2
SA-RL-0.6B	100, 200, 300, 500	0.54 $\rightarrow$ 0.76	0.23 $\rightarrow$ 0.47	1038 $\rightarrow$ 1519	none
SA-RL-1.7B	100, 200, 300	0.47 $\rightarrow$ 0.59	0.25 $\rightarrow$ 0.47	1013 $\rightarrow$ 1307	none
SA-RL-4B	100, 200, 300	0.48 $\rightarrow$ 0.59	0.29 $\rightarrow$ 0.46	1051 $\rightarrow$ 1360	none

Fig. 8 follows the IP first attempt over training. The generator comes to finish its first attempt mainly when it is likely right. It finishes less often as training proceeds, and the first attempts it finishes become more accurate than SA-RL's finished outputs, also when both are restricted to the 100 sampled problems, and are nearly always correct at 4B. At 0.6B and 4B the loop's net contribution grows as first-attempt accuracy falls. At 0.6B the evaluator often states the correct answer before the generator does, and at 4B rarely, so where the evaluator contributes beyond its verdict it does so by solving, not by critique (Table 11, column eval.).

C.6 Training-Side Early Warnings

Table 12 gives the values for the monitoring analysis of §4.4. Panel A groups the grid runs by the drop from their validation peak to their last evaluated checkpoint. The declining cells are the six Voting and Orch-Workers IP Math cells, which end 2.5–13.2 points below their peak: the five declining runs of §4.4 that have a repeated role, and `Orch-Workers-IP-0.6B-Math`, whose drop is not significant. The non-declining cells are the runs of every strategy and task that end within 2 points of their peak; all six SA-RL

Table 10: Reward accounting of the solving roles and SA-RL on Code and Math, each output graded on its own answer, on the 100 sampled problems at every checkpoint of both tasks. Panel A pools the trained checkpoints sampled in 18 Code and 18 Math multi-agent cells, and the SA-RL checkpoints; row groups are shares of all solving-role outputs, of truncated ones and of problems, differences from SA-RL in points (within and below count checkpoints), and counts of training runs. Orch.: Orch-Workers. Panel B: Code checkpoints with at least 5 truncated solving-role outputs whose truncated outputs are most often rewarded; SE: paired per-problem standard error. Definitions in the text and §B.5. †Includes `Orch-Workers-IP-0.6B-Code`, whose workers leave code blocks unclosed; see §C.4.

Panel A. Solving-role outputs, problems and training runs, pooled per task and workflow.

	Code				Math			
	Voting	Orch.†	Eval-Opt	SA-RL	Voting	Orch.	Eval-Opt	SA-RL
Checkpoints	18	17	18	10	25	19	21	10
All solving-role outputs								
At the length cap	39.8	35.1	53.8	40.1	24.4	31.9	52.5	32.3
Truncated and rewarded	0.7	0.7	1.9	0.0	3.0	8.4	13.3	0.1
Rewarded but wrong	2.1	5.6	3.7	0.0	8.2	14.2	13.1	0.0
Truncated solving-role outputs								
Correct	0.0	0.0	0.0	0.0	0.4	1.0	1.1	0.3
Rewarded	1.7	2.0	3.6	0.0	12.3	26.4	25.4	0.3
No instance solves	97.7	98.5	—	—	84.7	82.4	—	—
SA-RL solves	2.2	1.2	5.0	—	22.1	22.4	21.3	—
Problems								
Instances disagree	7.2	9.5	4.4	—	25.7	28.5	13.5	—
Solving role minus SA-RL								
Mean (points)	−0.3	−2.9	−2.0	—	−8.7	−9.2	−7.7	—
Mean at 0.6B	+0.7	−4.4†	−1.8	—	−9.1	−7.1	−9.7	—
Mean at 1.7B	+1.1	+0.1	−0.3	—	−9.1	−12.2	−8.7	—
Mean at 4B	−3.4	−4.7	−3.8	—	−8.0	−8.4	−4.3	—
Within 3 points	15/18	10/17	15/18	—	3/25	3/19	2/21	—
More than 2 SE below	0/18	5/17	1/18	—	14/25	12/19	11/21	—
Training runs								
Cap-hit rate falls back	0/6	0/6	0/6	0/3	1/6	0/6	0/6	3/3

Panel B. Code checkpoints whose truncated solving-role outputs are most often rewarded.

Cell	Step	At the cap	Truncated: rewarded	Rewarded but wrong	Minus SA-RL (SE)
`Eval-Opt-SP-4B-Code`	265	70.0	15.7	11.0	−12.0 (3.8)
`Orch-Workers-IP-4B-Code`	230	22.3	11.9	9.7	−4.0 (2.3)
`Eval-Opt-SP-4B-Code`	230	65.0	10.8	8.0	−2.0 (3.8)
`Orch-Workers-SP-4B-Code`	160	24.0	6.9	9.3	−3.0 (2.8)
`Voting-SP-4B-Code`	120	22.3	6.0	2.3	−3.3 (2.6)

runs are among them. The other runs drop by 2.1–4.2 points: `Eval-Opt-IP-1.7B-Math`, `Voting-IP-4B-Code`, `Orch-Workers-SP-4B-Math`, `Eval-Opt-IP-4B-Code` and `Eval-Opt-SP-1.7B-Math`.

Signals. Thresholds are relative to the early median, except for the training-success rule, which uses its running maximum; onsets and leads follow §B.5. The repeated-role row follows the Voting generator or the Orch-Workers worker, which have their own adapter only under IP. The any-role rows take the earliest onset over a run's roles. Global entropy and the cap-hit rate are pooled over roles. A false positive is a non-declining cell in which the signal fires at any step.

`Eval-Opt-IP-1.7B-Math`, the sixth declining run of §4.4, has no repeated role. No role's χ^2 or perplexity crosses its threshold in that run, and of the other signals only the cap-hit rate and the training-success rule fire. Every gradient-norm onset, in declining and non-declining cells alike, falls on a decision role (the Voting aggregator or the Orch-Workers synthesizer).

Table 11: Per-turn decomposition of Eval-Opt episodes, every generator turn graded on its own answer. Final: accuracy of the scored answer. First attempt: accuracy, truncated share and accuracy of finished attempts. Loop net: net contribution. Recovered: over T-wrong and over F-wrong first attempts; eval.: recovered with the evaluator's answer. Lost: over correct first attempts. Evaluator: share of first verdicts that accept, accept rate on correct first attempts (TPR), rejection rates on F-wrong and T-wrong ones. Per episode: means. SA-RL: accuracy / accuracy of finished outputs (step). Math values use all 1412 validation problems (*: the 100 sampled problems). Bottom: IP minus SP on Math. Definitions in §B.5.

				First attempt			Loop	Recovered				Evaluator, first verdict				Per episode		SA-RL
Scale	Strategy	Step	Final	acc.	trunc.	fin. acc.	net (pp)	T-wrong	F-wrong	eval.	Lost	Accept	TPR	Rej. F-wrong	Rej. T-wrong	Rounds	Tokens	acc. / fin. (step)
Math: 5,120 tokens per turn, up to three generator turns																		
0.6B	Base	0	0.10	0.10	0.00	0.10	+0.1	0/7	2/1259	0	0/146	0.91	0.93	0.09	0.29	0.11	976	–
	IP	100	0.29	0.21	0.51	0.44	+7.8	101/726	15/383	29	6/303	0.36	0.84	0.37	0.99	1.12	7,457	0.23 / 0.39* (100)
	IP	190	0.36	0.14	0.82	0.76	+22.3	309/1156	6/61	235	0/195	0.14	0.81	0.38	1.00	1.50	9,889	0.26 / 0.53* (190 / 200)
	IP	200	0.37	0.12	0.86	0.80	+25.0	353/1204	0/39	262	0/169	0.14	0.94	0.05	1.00	1.31	9,569	0.26 / 0.53* (200)
	IP	230	0.39	0.10	0.90	0.88	+28.9	405/1257	3/16	344	0/139	0.09	0.78	0.19	1.00	1.53	10,195	0.27 / 0.53* (230 / 200)
	SP	110*	0.16	0.16	0.20	0.20	0	0/20	0/64	0	0/16	0.65	0.88	0.20	1.00	0.68	3,330	0.22 / 0.39* (110 / 100)
	SP	210	0.29	0.18	0.65	0.52	+10.3	146/917	1/238	67	1/257	0.36	0.99	0.02	0.97	0.99	7,607	0.27 / 0.53* (210 / 200)
	SP	220	0.29	0.17	0.72	0.59	+12.1	172/1008	0/162	109	1/242	0.30	0.97	0.01	0.98	1.13	8,396	0.28 / 0.53* (220 / 200)
1.7B	Base	0	0.27	0.27	0.03	0.28	+0.1	0/49	2/985	0	0/378	0.98	1.00	0.02	0.00	0.03	1,574	–
	IP	160*	0.53	0.43	0.37	0.68	+10	10/37	1/20	0	1/43	0.54	0.98	0.40	1.00	0.83	5,461	0.42 / 0.67* (160 / 200)
	IP	260	0.59	0.38	0.58	0.89	+20.8	293/808	0/67	88	0/537	0.41	0.95	0.15	0.99	0.95	6,896	0.48 / 0.65* (260 / 300)
	IP	300*	0.54	0.37	0.54	0.80	+17	17/54	0/9	3	0/37	0.47	1.00	0.00	0.98	0.93	7,150	0.50 / 0.65* (300)
	IP	320	0.55	0.41	0.52	0.83	+14.8	209/728	0/112	50	0/572	0.50	0.99	0.05	0.95	0.82	6,465	0.50 / 0.65* (320 / 300)
	IP	340*	0.49	0.34	0.60	0.85	+15	15/60	0/6	2	0/34	0.44	1.00	0.00	0.93	0.91	7,218	0.54 / 0.65* (340 / 300)
	IP	360	0.53	0.39	0.56	0.87	+13.5	190/779	0/79	64	0/554	0.50	1.00	0.01	0.90	0.72	6,812	0.51 / 0.65* (360 / 300)
	IP	370	0.55	0.39	0.54	0.85	+15.2	214/756	0/99	73	0/557	0.51	1.00	0.02	0.90	0.70	6,727	0.53 / 0.62 (370 / 430)
	SP	150*	0.33	0.30	0.26	0.41	+3	4/26	0/44	0	1/30	0.52	0.80	0.39	0.96	0.89	4,625	0.41 / 0.52 (150 / 100)
	SP	290	0.48	0.40	0.42	0.68	+8.6	119/592	4/261	73	1/559	0.60	0.98	0.06	0.91	0.67	5,476	0.50 / 0.65* (290 / 300)
	SP	320	0.54	0.39	0.51	0.79	+15.3	217/714	2/145	138	3/553	0.48	0.93	0.08	0.96	0.87	6,638	0.50 / 0.65* (320 / 300)
4B	Base	0	0.48	0.47	0.04	0.49	+0.6	5/52	41/690	2	37/670	0.54	0.80	0.67	1.00	0.84	2,655	–
	IP	90*	0.63	0.56	0.31	0.81	+7	8/31	0/13	0	1/56	0.65	0.96	0.31	0.94	0.59	4,672	0.61 / 0.73* (90 / 100)
	IP	180*	0.71	0.44	0.54	0.96	+27	27/54	0/2	2	0/44	0.46	1.00	0.00	1.00	0.81	7,256	0.64 / 0.79* (180 / 200)
	IP	220	0.75	0.40	0.60	0.98	+35.5	503/838	0/11	34	2/563	0.39	0.96	0.09	1.00	0.89	7,401	0.64 / 0.79* (220 / 200)
	SP	105*	0.59	0.57	0.10	0.63	+2	0/10	6/33	0	4/57	0.42	0.60	0.76	1.00	1.00	3,564	0.60 / 0.73* (100)
	SP	210*	0.61	0.52	0.37	0.81	+9	9/36	0/12	1	0/52	0.34	0.54	0.50	1.00	0.99	5,092	0.63 / 0.79* (210 / 200)
	SP	215	0.66	0.59	0.27	0.81	+6.9	94/373	17/200	12	14/839	0.32	0.43	0.57	0.99	1.02	4,453	0.63 / 0.79* (210 / 200)
Code: 2,048 tokens per turn, up to two generator turns																		
0.6B	Base	0	0.08	0.08	0.02	0.08	0	0/2	0/90	0	0/8	0.75	0.62	0.24	0.00	0.25	361	–
	IP	90	0.09	0.09	0.41	0.15	0	0/41	0/50	0	0/9	0.00	0.00	1.00	1.00	1.00	2,029	0.09 / 0.12 (100)
	IP	170	0.14	0.07	0.63	0.19	+7	1/63	6/30	0	0/7	0.00	0.00	1.00	1.00	1.00	2,627	0.10 / 0.18 (200)
	IP	240	0.12	0.11	0.82	0.61	+1	0/82	1/7	0	0/11	0.00	0.00	1.00	1.00	1.00	3,161	0.10 / 0.18 (200)
	SP	200	0.10	0.07	0.57	0.16	+3	1/57	2/36	2	0/7	0.10	0.00	0.83	0.93	0.90	2,228	0.10 / 0.18 (200)
	SP	390	0.14	0.07	0.80	0.35	+7	3/80	4/13	4	0/7	0.00	0.00	1.00	1.00	1.00	2,855	0.08 / 0.21 (300)
	SP	445	0.13	0.09	0.87	0.69	+4	2/87	2/4	4	0/9	0.00	0.00	1.00	1.00	1.00	2,983	0.14 / 0.26 (500)
1.7B	Base	0	0.14	0.14	0.10	0.16	0	0/10	0/76	0	0/14	0.82	1.00	0.12	0.90	0.18	810	–
	IP	150	0.16	0.14	0.47	0.26	+2	2/47	0/39	0	0/14	0.46	1.00	0.18	1.00	0.54	2,130	0.15 / 0.20 (100)
	IP	300	0.18	0.14	0.78	0.64	+4	3/78	1/8	1	0/14	0.14	0.71	0.50	1.00	0.86	2,919	0.15 / 0.28 (300)
	IP	350	0.17	0.14	0.82	0.78	+3	3/82	0/4	2	0/14	0.10	0.71	1.00	1.00	0.90	2,990	0.15 / 0.28 (300)
	SP	130	0.15	0.15	0.27	0.21	0	0/27	0/58	0	0/15	0.77	1.00	0.03	0.78	0.23	1,341	0.15 / 0.20 (100)
	SP	250	0.15	0.15	0.38	0.24	0	0/38	0/47	0	0/15	0.60	1.00	0.11	0.92	0.40	1,784	0.12 / 0.18 (200)
	SP	300	0.13	0.13	0.37	0.21	0	0/37	0/50	0	0/13	0.46	1.00	0.34	1.00	0.54	1,948	0.15 / 0.28 (300)
4B	Base	0	0.24	0.23	0.14	0.27	+1	1/14	2/63	0	2/23	0.41	0.74	0.62	1.00	0.59	1,036	–
	IP	70	0.21	0.24	0.17	0.29	−3	0/17	4/59	0	7/24	0.12	0.38	0.95	1.00	0.88	1,294	0.25 / 0.35 (100)
	IP	140	0.28	0.22	0.37	0.35	+6	2/37	4/41	0	0/22	0.15	0.50	0.90	1.00	0.85	1,973	0.25 / 0.35 (100)
	IP	150	0.24	0.24	0.33	0.36	0	0/33	3/43	0	3/24	0.19	0.58	0.88	1.00	0.81	1,898	0.25 / 0.35 (100)
	SP	115	0.22	0.21	0.28	0.29	+1	0/28	3/51	0	2/21	0.34	0.81	0.67	1.00	0.66	1,662	0.25 / 0.35 (100)
	SP	230	0.31	0.24	0.65	0.69	+7	7/65	1/11	0	1/24	0.21	0.71	0.73	0.98	0.79	2,636	0.26 / 0.48 (200)
	SP	265	0.29	0.18	0.70	0.60	+11	11/70	0/12	0	0/18	0.17	0.67	0.67	0.99	0.83	2,628	0.30 / 0.56 (300)

	Steps	IP minus SP (points)			T-wrong recovered	
Math scale	IP/ SP	Final	First attempt	Loop net	IP	SP
0.6B	230 / 220	+9.5	−7.3	+16.8	0.32	0.17
1.7B	320 / 320	+0.8	+1.3	−0.5	0.29	0.30
4B	220 / 215	+9.1	−19.5	+28.6	0.60	0.25

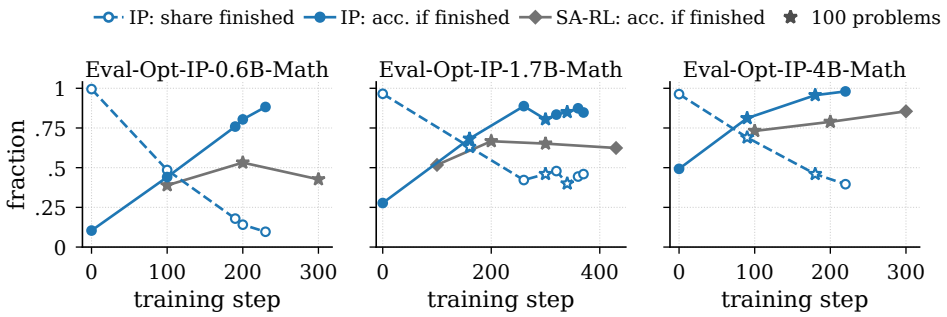

Figure 8: IP first attempts of Eval-Opt on Math over training; step 0 is the untrained workflow. Dashed: share of first attempts that finish under the length cap; solid: their accuracy; gray: accuracy of SA-RL's finished outputs at its sampled checkpoints; stars: the 100 sampled problems, other points all 1412. Values: Table 11.

Table 12: Training-side signals as early warnings of a post-peak validation decline; groups and signals as in the text, onsets and leads as in §B.5. Panel A: declining cells in which each signal fires, and fires before the validation peak; lead in steps for Voting (V) and Orch-Workers (O), positive before the peak (training success: at its running maximum); false positives among the non-declining and the SA-RL runs where the signal is defined; firings in the other runs. Baseline: the early median. –: does not fire, or not defined. Panel B: least-squares slope and Pearson r of log gradient norm on log output length; shortest role: multi-role runs whose largest-norm role has the shortest outputs.

Panel A. Onset of each training-side signal.

	Declining cells (6)								Non-declining (31)		Other (5)
			Lead to validation peak (steps)								
Signal	Fired	Before peak	V-0.6B	V-1.7B	V-4B	O-0.6B	O-1.7B	O-4B	False positives	SA-RL	Fired
Repeated-role $\chi^2 > 2\times$ baseline	6/6	5/6	+91	+40	−1	+33	+68	+10	1/5	–	0/1
Any-role $\chi^2 > 2\times$ baseline	6/6	5/6	+91	+40	−1	+33	+68	+10	13/31	1/6	3/5
Any-role perplexity $> 1.2\times$ baseline	6/6	4/6	+104	+15	−8	+50	+39	−4	10/31	0/6	2/5
Any-role perplexity $> 2\times$ baseline	5/6	0/6	−15	−43	−66	–	−29	−54	0/31	0/6	0/5
Any-role gradient norm $> 2\times$ baseline	4/6	2/6	−25	+106	–	–	+222	−32	2/31	0/6	0/5
Global entropy $> 1.5\times$ baseline	6/6	6/6	+108	+35	+11	+3	+39	+19	15/31	4/6	3/5
Cap-hit rate $>$ baseline $+ 0.10$	6/6	5/6	+156	+69	+16	+77	+59	−5	25/31	6/6	4/5
Training success 0.03 below its maximum	6/6	3/6	+42	−10	−24	0	+1	−13	3/31	0/6	4/5

Panel B. Gradient norm against output length (IP and SA-RL runs).

Task	Runs	Points	Slope	r	Shortest role
Math	12	85	−0.58	−0.89	7/9
Code	9	51	−0.26	−0.49	4/6
Both	21	136	−0.49	−0.79	11/15

Gradient norm and length. Panel B pools the IP and SA-RL runs, since SP runs log one gradient norm for the shared adapter. Each point is one role at one checkpoint with sampled validation outputs: the role's mean output length in tokens on those outputs, against its gradient norm averaged over the 11 training steps centered on the checkpoint. A slope of -0.5 corresponds to a norm proportional to the inverse square root of the length.

Online replay. Table 13 replays the signals using only the training steps observed up to each alarm. After its first crossing, the repeated role's χ^2 running median stays above its threshold in every declining cell, so an alarm at the first crossing reproduces the leads of Table 12; confirming the persistence first delays it. The last checkpoint before a χ^2 alarm is usually below the peak, so the alarm marks the onset of instability rather than a checkpoint to keep. Table 12 measures the training-success lead at the running maximum, so a positive lead there places the returned checkpoint, not the alarm, before the validation peak; the rule itself fires after the peak in every declining cell. On a trailing mean without the end-of-run clause it fires in one of the 31 non-declining runs, against three for the centered rule, and not on `Orch-Workers-IP-0.6B-Math` or, by step 379, on `Eval-Opt-IP-1.7B-Math`. The checkpoint it returns lies close to the peak and well above the last checkpoint in most declining cells.

Table 13: Online replay of the early warnings of Table 12 on the six declining cells, using only training steps logged up to each alarm (rules in §B.5). χ^2: the repeated role's χ^2 above twice its early median; lead to the validation peak with the alarm at the first crossing of the running median and once the persistence is confirmed; ckpt.: validation accuracy (%) of the last checkpoint before the confirmed alarm. Training success: trailing-mean rule; lead at its alarm, and accuracy of the checkpoint it returns. –: does not fire.

	Repeated-role χ^2			Training success			
	Lead (steps)					Validation (%)	
Cell	first crossing	confirmed	ckpt.	lead	ckpt.	peak	last
`Voting-IP-0.6B-Math`	+91	+72	25.3	−27	27.8	28.5	25.8
`Voting-IP-1.7B-Math`	+40	+21	49.4	−49	50.2	50.9	40.7
`Voting-IP-4B-Math`	−1	−20	67.0	−84	67.5	68.1	54.9
`Orch-Workers-IP-0.6B-Math`	+33	+14	32.7	–	–	36.0	33.6
`Orch-Workers-IP-1.7B-Math`	+68	+49	49.3	−49	55.5	55.5	51.0
`Orch-Workers-IP-4B-Math`	+10	−9	67.9	−80	66.4	67.9	60.0

D Additional Analysis of Policy Sharing and Post-Peak Decline

D.1 Per-Role Training Diagnostics and Trajectories

Table 14 gives the values for the training-metrics analysis of §4.4. It extends the repeated role's χ^2 series of Fig. 5 to every role of every IP Math cell and to the shared adapter of the matched SP cell, and adds perplexity and the gradient norm. The Voting aggregator, the Orch-Workers synthesizer and, on most training steps, the orchestrator fit in one mini-batch, so their χ^2 reflects only the mismatch between the rollout and training engines (§B.5).

Table 15 gives trajectory statistics (sampling in §B.4) for the role-output analysis of §4.4 on two IP cells, both plotted in Fig. 5. In both panels, from the first to the last sampled checkpoint, the repeated role's outputs lengthen, reach the length cap more often, hedge more and share fewer 3-grams across instances.

D.2 Per-Role Gradient Measurements

Gradients are decomposed by role at sampled checkpoints, on rollouts that the checkpoint generates on a fixed set of training problems. Each role's gradient is accumulated over one training step, with the three instances of a repeated role summed, so the shares describe one training step's gradient, not the update applied between checkpoints. A role's mean length is its response tokens per sequence averaged over the sampled checkpoints, the L of the length model in Fig. 4(c).

Panels. Panel D1 of Table 16 covers the SP cells of Fig. 4(c). It also gives each role's projection on the summed gradient, as a fraction of the summed gradient's squared norm over the same checkpoints. The projections sum to one over roles and net out gradient that other roles cancel; on every Math cell they place more of the gradient on the decision role than the norm shares do. Panel D2 covers `Voting-IP-1.7B-Math` and its per-voter-reward and one-adapter-per-voter controls (Table 19, §D.4). For the repeated role it gives the fraction of episodes in which all three copies received the identical advantage and the mean cosine between the gradients of copies answering the same problem. It also gives the range of the aggregator adapter's gradient norm over the generator adapter's, and the generator adapter's parameter change per training step between sampled checkpoints over the aggregator adapter's. Panel D3 divides, window by window, the rate of each SP adapter by the largest rate among the adapters of the matched IP run, over the windows that both runs' checkpoints cover. A smallest window ratio above 1 means that the shared adapter moved farther than every IP adapter in every window.

Gradient shares. On every measured SP Math cell the decision role's share exceeds its uniform share at every sampled checkpoint, rising with scale on Voting, while the solving role's stays below its own; the decision role's outputs are several times shorter. On both Code cells, measured after the decision role's output form had changed (Table 8, Panel A), the two lengths are closer, the

Table 14: Per-role training diagnostics on the Math cells: peak and late ratios of χ^2, perplexity (PPL) and gradient norm, and the step of the χ^2 peak (definitions in §B.5). Each IP row is one role's adapter; $\times 3$ marks a repeated role, whose three instances train one adapter. The SP row is the one adapter of the matched SP run, which trains on every role's sequences.

			Peak			Mean over last 30 steps			Step at
Cell	Policy	Role	χ^2	PPL	Grad norm	χ^2	PPL	Grad norm	χ^2 peak
Voting-0.6B-Math	IP	generator ($\times 3$)	6.01	2.85	1.24	5.08	2.43	0.61	308
	IP	aggregator	1.72	1.18	4.24	1.03	0.94	3.13	88
	SP	shared adapter	2.05	1.20	1.99	1.84	1.16	1.05	199
Voting-1.7B-Math	IP	generator ($\times 3$)	25.47	7.50	1.28	22.39	6.21	1.04	291
	IP	aggregator	1.12	1.02	13.05	0.76	0.95	0.99	66
	SP	shared adapter	1.50	1.04	2.80	1.39	0.99	1.29	281
Voting-4B-Math	IP	generator ($\times 3$)	13.36	4.40	3.64	11.05	3.48	1.32	216
	IP	aggregator	34.84	1.20	2.50	0.45	0.99	0.84	133
	SP	shared adapter	1.44	1.07	3.78	1.29	1.05	0.51	182
Orch-Workers-0.6B-Math	IP	orchestrator	1.12	1.03	2.28	0.02	0.67	0.56	35
	IP	worker ($\times 3$)	3.95	1.98	1.21	3.40	1.77	0.54	235
	IP	synthesizer	1.37	1.08	2.25	1.01	0.97	1.31	123
	SP	shared adapter	2.83	1.38	1.96	2.60	1.32	1.22	170
Orch-Workers-1.7B-Math	IP	orchestrator	1.35	1.04	1.61	1.29	1.03	1.04	316
	IP	worker ($\times 3$)	19.09	4.19	1.25	12.85	2.83	0.65	346
	IP	synthesizer	1.12	1.05	6.41	0.81	0.92	2.22	44
	SP	shared adapter	2.68	1.12	2.53	2.36	1.08	1.24	336
Orch-Workers-4B-Math	IP	orchestrator	1.15	1.02	1.26	1.05	1.01	0.67	155
	IP	worker ($\times 3$)	13.26	4.25	1.66	9.69	2.98	1.01	184
	IP	synthesizer	1.58	1.13	12.66	0.65	0.91	3.71	88
	SP	shared adapter	8.65	2.16	2.01	6.65	1.74	1.12	221
Eval-Opt-0.6B-Math	IP	generator	2.73	1.43	3.71	2.41	1.35	0.82	228
	IP	evaluator	1.18	1.21	4.03	0.92	1.14	1.17	129
	SP	shared adapter	1.56	1.14	3.14	1.45	1.11	0.55	224
Eval-Opt-1.7B-Math	IP	generator	1.22	1.02	12.49	0.40	0.92	2.11	148
	IP	evaluator	1.71	1.10	2.90	1.50	1.05	0.47	309
	SP	shared adapter	7.80	1.98	5.30	6.51	1.75	0.91	377
Eval-Opt-4B-Math	IP	generator	3.99	1.49	2.65	3.18	1.34	0.69	235
	IP	evaluator	1.24	1.02	10.00	0.75	0.98	0.97	94
	SP	shared adapter	2.88	1.40	2.96	2.54	1.32	1.24	210

norm shares are near uniform, and the projected share of the solving role is at or above uniform. In `Voting-IP-1.7B-Math` the aggregator holds a comparable share of the summed per-role gradient (Panel D2), so the concentration comes from the role's short outputs, and sharing decides only which parameters that gradient updates. The `Voting-SP-4B-Math` aggregator, the `Orch-Workers-SP-4B-Math` synthesizer and, by less, the `Eval-Opt-SP-1.7B-Math` evaluator sit above the length model of Fig. 4(c), and on `Voting-SP-1.7B-Code` the token share fits the measured share better than the length model does.

Parameter change. The shared adapter trains on every role's sequences and so takes more optimizer steps per training step than any IP adapter (§4.3). Averaged over the compared windows, it moves farther in every pair but Voting at 4B on Code, and it does so in every window in all but that pair and three Eval-Opt pairs (Panel D3). Fig. 9 compares every adapter with the noise model.

D.3 Decline Windows and Parameter Change of the Repeated Role

Table 17 gives the per-checkpoint values for the role-output analysis of §4.4, and Table 18 the values for its training-metrics analysis and for the Code repeated adapters of its controls analysis.

Decline windows. Table 17 covers the declining IP Math runs of Voting and Orch-Workers whose sampled checkpoints span the validation peak, which excludes `Orch-Workers-IP-0.6B-Math`. Each SP window ends at the sampled checkpoint nearest the end of the IP window and starts at the earlier checkpoint nearest its start. A problem is lost when the reference checkpoint answers it correctly and the row's checkpoint does not, and gained in the reverse case. A 1412-problem checkpoint also appears restricted to the 100 sampled problems, to pair with its run's other checkpoints.

Table 15: Trajectory statistics at three sampled checkpoints of `Voting-IP-1.7B-Math` (Panel C1; repeated role: the generator) and `Orch-Workers-IP-1.7B-Math` (Panel C2; repeated role: the worker). Repeated-role columns: mean length, truncation rate, rate of a parseable `\boxed{}` answer, hedging-phrase rate and 3-gram Jaccard between the three instances (§B.5). Other columns: aggregator and synthesizer length (median p50, 95th percentile p95) and the number of distinct strategies the orchestrator assigns first.

Panel C1. `Voting-IP-1.7B-Math` (100 DAPO-Math problems per step, 3 generators / 1 aggregator per problem).

	Generator (300 outputs / step)					Aggregator (100 / step)
Step	mean toks	trunc. rate	`\boxed{}` rate	hedge rate	inter-gen 3-gram Jaccard	median toks
70	2167	0.093	0.907	0.260	0.144	126
130	2899	0.237	0.767	0.507	0.126	149
290	3260	0.283	0.810	0.877	0.025	155

Panel C2. `Orch-Workers-IP-1.7B-Math` (100 DAPO-Math problems per step, 3 workers / 1 orchestrator / 1 synthesizer per problem).

	Worker (300 outputs / step)					Orchestrator	Synthesizer (100 / step)	
Step	mean toks	trunc. rate	`\boxed{}` rate	hedge rate	inter-worker 3-gram Jaccard	unique first-strategy labels (of 100)	p50 toks	p95 toks
140	2351	0.133	0.870	0.367	0.117	89	95	425
280	3760	0.453	0.550	0.853	0.101	71	147	2734
330	4298	0.647	0.360	0.903	0.065	76	292	5120

Table 16: Per-role gradient measurements and parameter change (definitions in §B.5). Panel D1: SP cells. Panel D2: `Voting-IP-1.7B-Math` and two of its controls. Share: smallest and largest fraction of the summed per-role gradient norm over the sampled checkpoints; uniform: the share if every role call contributed equally. Projection: the role's share along the summed gradient. Optimizer steps are per training step; var.: varies with the revision rounds per episode. ‡Mean of the three voter adapters; —: not available. Panel D3: parameter change per training step of the shared adapter over that of the fastest adapter of the matched IP run; mean over windows (smallest window ratio).

Panel D1. Shared policy: one adapter serves every role.

		Decision role's share			Solving role's share				Mean length (tokens)		Per training step
Cell	Decision role	range	projection	uniform	solving role	range	projection	uniform	decision	solving	optimizer steps
`Voting-SP-0.6B-Math`	aggregator	0.542–0.782	0.583–0.922	0.25	generator	0.218–0.458	0.078–0.417	0.75	465	2675	4
`Voting-SP-1.7B-Math`	aggregator	0.764–0.903	0.915–0.991	0.25	generator	0.097–0.236	0.009–0.085	0.75	163	2241	4
`Voting-SP-4B-Math`	aggregator	0.909–0.964	0.990–0.999	0.25	generator	0.036–0.091	0.001–0.010	0.75	289	1795	4
`Eval-Opt-SP-1.7B-Math`	evaluator	0.791–0.945	0.929–0.995	0.50	generator	0.055–0.209	0.005–0.071	0.50	1080	3331	var.
`Orch-Workers-SP-4B-Math`	synthesizer	0.299–0.760	0.217–0.935	0.20	worker	0.048–0.151	0.004–0.057	0.60	1021	3200	5
`Voting-SP-1.7B-Code`	aggregator	0.146–0.333	0.029–0.182	0.25	generator	0.667–0.854	0.818–0.971	0.75	832	875	4
`Eval-Opt-SP-0.6B-Code`	evaluator	0.520–0.620	0.540–0.727	0.50	generator	0.380–0.480	0.273–0.460	0.50	873	1300	var.

Panel D2. Isolated policies: each role has its own parameters.

		Decision role's share		Same-role copies		Per training step		
Cell	Decision role	range	uniform	identical-advantage fraction	copy cosine	aggregator / generator gradient-norm ratio	parameter change vs. aggregator	optimizer steps (repeated / decision role)
`Voting-IP-1.7B-Math`	aggregator	0.781–0.885	0.25	1.000	+0.0029	3.3–20.9	2.12	3 / 1
`Voting-IP-1.7B-Math` (per-voter reward)	aggregator	0.942–0.980	0.25	0.464	+0.0116	13.6–127.8	2.00	3 / 1
`Voting-IP-1.7B-Math` (one adapter per voter, rubric reward)	aggregator	0.561–0.675	0.25	—	—	3.3–6.5‡	0.84‡	1 / 1

Panel D3. Parameter change per training step, shared adapter over the fastest IP adapter of the matched run.

Workflow	Task	0.6B	1.7B	4B
Eval-Opt	Math	1.13 (0.95)	1.23 (0.97)	1.32 (1.04)
Eval-Opt	Code	1.06 (0.68)	1.34 (1.20)	1.22 (1.10)
Voting	Math	1.39 (1.04)	1.27 (1.11)	1.23 (1.01)
Voting	Code	1.24 (1.04)	1.09 (1.07)	0.84 (0.79)
Orch-Workers	Math	1.64 (1.38)	1.52 (1.42)	1.55 (1.50)
Orch-Workers	Code	1.35 (1.17)	1.36 (1.29)	1.42 (1.36)

Training metrics. In Table 18, each validation drop is the peak minus the last validated checkpoint within its horizon. The training-success drop compares the 31 training steps centered on the full-run validation peak with the last 30. Mism. is the mismatch ratio of §B.5. Over the 41 series in which one adapter produces most of the batch's tokens, the log χ^2 ratio is fitted by 1.25 times the log ratio of the

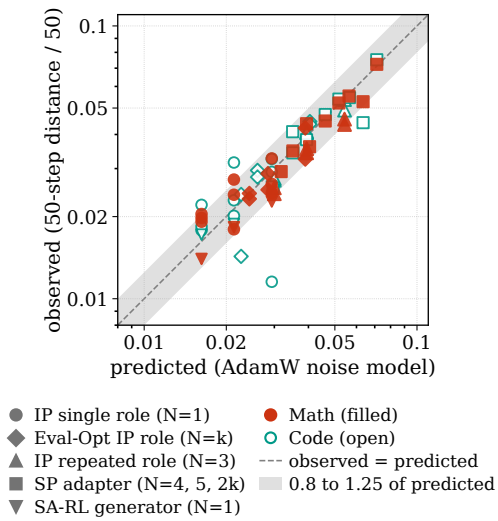

Figure 9: Parameter change per training step (rate) of the 66 adapters of the 36 cells and the six single-agent RL runs (42 IP, 18 SP, 6 single-agent RL) against the AdamW noise model with the adapter's N; definitions in §B.5. Band: 0.8 to 1.25 times the model; 58 of the 66 adapters lie within it.

mean per-token entropy ($R^2 = 0.84$), against $R^2 = 0.03$ for the mean response length alone. These series are the IP repeated roles and Eval-Opt generators, the SP adapters and the SA-RL generators.

Parameter change. Most adapters move per training step about as far as the AdamW noise model predicts from their N alone (Fig. 9). With the mini-batch size fixed in sequences, the repeated role's adapter under IP takes one optimizer step per instance, and over the same windows it moves farther than the single-instance adapters of its run, as the model predicts (Table 18). The SP adapter moves farther still (Table 16, Panel D3) and declines little if at all, and the Code repeated adapters move as far as the Math ones without declining (Table 18). Distance per training step therefore does not separate the declining runs from the rest.

Direction persistence. Δ cos is the mean cosine between an adapter's consecutive 50-step changes minus its noise-model value at the same N. It sets the repeated adapters of `Voting-IP-1.7B-Math`, `Voting-IP-4B-Math` and `Orch-Workers-IP-4B-Math`, three of the five declining runs with a repeated role, apart from every other adapter (Table 18). On the 35 adapters of the grid not in the table, it is at most $+0.14$. Across the 11 IP repeated adapters with a value, its Spearman correlation with the validation drop at step ≤ 300 is $+0.77$. It averages only a few pairs of changes per adapter, on checkpoints that span the decline, so it is a correlate of the decline and does not show when the decline begins.

D.4 Control Runs

Table 19 gives the controls of §4.1 and §4.4. It varies two conditions on the generator's adapter: how many optimizer steps it takes per training step, and whether any other role's gradient updates it. In the shared-policy row, `Voting-SP-1.7B-Math`, the aggregator's gradient also updates the generator's adapter.

Settings. The reference uses the broadcast reward. The per-voter-reward control keeps the reference's adapters and scores each generator instance on its own answer and the aggregator on the final answer. The per-voter-prompts control keeps the reference's adapters and reward and, during training, appends a different solution-approach instruction to each generator instance's prompt. The single-role control removes the Voting workflow and the aggregator. Each problem receives three groups of eight samples from the generator prompt, each scored on its own answer and all trained on one adapter. The variant with one adapter per voter instance changes both the adapter assignment and the reward, which it replaces with a rubric whose per-instance component is the per-voter reward. A voter receives $+0.5$ for a correct own answer, $+0.5$ for a correct final answer and -1 for a wrong own boxed answer; the

Table 17: Voters and workers at sampled checkpoints before and after the validation peak of the declining IP Math runs and over matched SP windows, one block per run, on all 1412 validation problems (*: the 100 sampled problems). Peak: selected peak (step). Val.: validation accuracy. Acc.: workflow accuracy on the row's problems. Inst.: mean instance accuracy. Loss, Rescue: selection losses, rescues. Trunc.: truncated instance outputs. Lost/gained: problems lost and gained relative to the reference checkpoint Ref., with the McNemar p.

Run	Step	Val.	Acc.	Oracle	Inst.	Loss	Rescue	Trunc.	Ref.	Final answer		Oracle	
										Lost/gained	p	Lost/gained	p
`Voting-IP-0.6B-Math`	200*	25.9	20.0	28.0	15.7	8.0	0.0	65.7					
Peak 28.5 (260)	300*	25.8	26.0	34.0	19.0	8.0	0.0	43.7	200	6/12	0.24	5/11	0.21
	300	25.8	26.5	32.2	19.8	5.7	0.0	44.3					
`Voting-SP-0.6B-Math`	90*	14.2	12.0	23.0	13.0	11.0	0.0	3.0					
Peak 28.3 (200)	200*	28.3	25.0	34.0	20.0	9.0	0.0	47.3	90	2/15	0.002	4/15	0.02
	200	28.3	28.5	36.2	22.3	7.7	0.0	46.6					
`Voting-IP-1.7B-Math`	130*	47.7	40.0	52.0	35.7	12.0	0.0	23.7					
Peak 50.9 (190)	290*	40.9	33.0	41.0	29.3	8.0	0.0	28.3	130	13/6	0.17	15/4	0.02
	300*	40.7	31.0	41.0	28.7	10.0	0.0	43.0	130	16/7	0.09	16/5	0.03
	300	40.7	39.9	46.3	32.4	6.4	0.0	46.6					
`Voting-SP-1.7B-Math`	120*	38.0	35.0	47.0	29.7	12.0	0.0	5.0					
Peak 46.5 (300)	300*	46.5	44.0	52.0	36.0	8.0	0.0	20.3	120	7/16	0.09	10/15	0.42
	300	46.5	46.0	55.8	38.4	9.8	0.0	15.9					
`Voting-IP-4B-Math`	70*	65.6	56.0	60.0	49.0	4.0	0.0	21.0					
Peak 68.1 (100)	130*	67.5	58.0	63.0	50.3	5.0	0.0	30.3	70	6/8	0.79	7/10	0.63
	200*	–	47.0	55.0	41.7	8.0	0.0	26.0	130	19/8	0.05	15/7	0.13
	195	59.8	59.8	65.9	49.5	6.1	0.1	21.6					
	215	54.9	54.9	60.6	44.5	5.8	0.1	29.4	195	152/82	5.6×10^{-6}	134/60	1.1×10^{-7}
`Voting-SP-4B-Math`	140	56.2	55.8	65.0	48.3	9.5	0.3	4.8					
Peak 56.2 (140)	170	55.4	55.2	65.4	48.8	10.2	0.0	5.5	140	140/132	0.67	109/115	0.74
`Orch-Workers-IP-1.7B-Math`	280*	55.2	49.0	50.0	34.3	7.0	6.0	45.3					
Peak 55.5 (290)	330*	53.7	48.0	36.0	24.7	1.0	13.0	67.3	280	11/10	1.00	17/3	0.003
	330	53.7	52.9	40.3	29.2	1.1	13.7	63.7					
`Orch-Workers-SP-1.7B-Math`	300	52.3	51.0	53.2	35.8	5.7	3.5	40.3					
Peak 52.3 (300)	330	52.7	53.0	52.7	36.1	4.4	4.7	44.5	300	114/143	0.08	116/109	0.69
`Orch-Workers-IP-4B-Math`	60*	60.6	51.0	60.0	43.7	9.0	0.0	12.7					
Peak 67.9 (100)	120*	66.4	60.0	65.0	48.3	6.0	1.0	34.0	60	2/11	0.02	3/8	0.23
	180*	60.0	52.0	35.0	26.3	1.0	18.0	70.0	120	13/5	0.10	31/1	1.5×10^{-8}
	180	60.0	61.3	46.9	34.4	1.2	15.6	62.3					
`Orch-Workers-SP-4B-Math`	130*	63.5	56.0	62.0	45.0	6.0	0.0	18.3					
Peak 69.6 (190)	170*	67.8	59.0	60.0	44.0	3.0	2.0	33.7	130	8/11	0.65	9/7	0.80

aggregator receives $+1$ or -1 by the final answer when at least one voter is correct and 0 otherwise. Its advantage groups are per voter instance, as in the reference on Math (§B.2).

Diagnostics, validation series and learning rate. In Table 19, every setting of the upper group ends with a generator χ^2 ratio of the same order as the reference's, while in both settings of the lower group the ratio stays near its early level. Table 20, Panel A, compares the settings every 30 training steps, where every run is validated; the single-role control is validated as a single agent and the per-voter-prompts control with the reference prompts. The per-voter-reward control leads the reference at every compared checkpoint and peaks at about the same step; per-voter prompts slow the rise and delay the peak; both decline after their peaks, as the single-role control does. Every single-agent RL run ends close to its peak over its full validated horizon (Panel B). Panel C gives the cap-hit rate of each setting. Three settings warm the learning rate up, and each reaches the constant rate before any compared peak (Panel A, last column).

Per-role reward on Eval-Opt. A control of `Eval-Opt-IP-1.7B-Math` rewards each generator turn for its own answer and the evaluator for the accuracy of its verdict. Early in training, the evaluator's verdict accuracy comes to equal one minus the generator's accuracy at every training step, and every episode uses all three generator turns, which is what an evaluator that rejects every attempt produces. The evaluator's χ^2 and perplexity drop to their lowest values at the same time. In a variant that starts the generator from the single-agent RL adapter, whose first attempts are more often correct than the base model's, the evaluator does not collapse.

E Relation to AT-GRPO and Dr. MAS

AT-GRPO (Zhao et al., 2025c) and Dr. MAS (Feng et al., 2026) each propose a training rule for multi-agent LLM systems and evaluate it on their own workflows. This paper holds one multi-agent

Table 18: Parameter change and diagnostics of the repeated role's adapter under IP (the Voting generator or the Orch-Workers worker), of the matched SP adapter and of the SA-RL generators; N, Rate and the noise model are defined in §B.5. Pred.: the noise model's rate. vs. single: Rate over the mean rate of the run's single-instance adapters. $\Delta\cos$: excess cosine of consecutive 50-step changes over the noise model (pairs in parentheses). χ^2, Mism.: late ratios of the adapter's χ^2 and of the rollout-versus-training mismatch; drop columns defined in §D.3.

										Drop after peak (pp)		
Setting	Adapter	N	Rate	Pred.	Obs./pred.	vs. single	$\Delta\cos$	χ^2	Mism.	Val. ≤ 300	Val. full	Train.
Math												
Voting 0.6B	IP repeated	3	0.191	0.236	0.81	1.22	+0.11 (4)	5.1	2.3	2.8	2.8	+1.3
	SP shared	4	0.274	0.276	0.99	1.88	+0.03 (2)	1.8	1.5	0.0	0.0	+1.5
Voting 1.7B	IP repeated	3	0.212	0.236	0.90	2.19	+0.26 (4)	22.4	3.5	10.2	10.2	+15.0
	SP shared	4	0.268	0.276	0.97	2.77	+0.01 (4)	1.4	1.3	0.0	0.0	+2.1
Voting 4B	IP repeated	3	0.189	0.236	0.80	1.66	+0.24 (2)	11.1	3.1	13.2	13.2	+9.4
	SP shared	4	0.229	0.276	0.83	2.00	+0.02 (1)	1.3	1.2	0.6	0.6	−0.2
Orch-Workers 0.6B	IP repeated	3	0.201	0.236	0.85	1.33	+0.10 (2)	3.4	1.9	2.5	2.5	+1.1
	SP shared	5	0.345	0.310	1.11	2.25	+0.04 (1)	2.6	1.9	0.0	0.0	+0.3
Orch-Workers 1.7B	IP repeated	3	0.205	0.236	0.87	1.34	+0.12 (4)	12.9	3.2	1.2	4.5	+4.2
	SP shared	5	0.312	0.310	1.01	2.05	+0.04 (4)	2.4	1.6	0.0	0.1	0.0
Orch-Workers 4B	IP repeated	3	0.198	0.236	0.84	1.63	+0.25 (1)	9.7	2.9	7.9	7.9	+6.0
	SP shared	5	0.315	0.310	1.02	2.56	+0.08 (2)	6.7	2.9	2.6	2.6	+0.5
Code												
Voting 0.6B	IP repeated	3	0.217	0.236	0.92	1.52	0.00 (4)	1.9	1.7	0.0	0.9	+0.2
	SP shared	4	0.270	0.276	0.98	1.89	+0.02 (4)	1.6	1.6	1.0	1.5	0.0
Voting 1.7B	IP repeated	3	0.243	0.236	1.03	1.29	–	0.7	0.9	0.4	0.4	−0.5
	SP shared	4	0.283	0.276	1.03	1.38	–	0.5	–	0.1	0.1	+0.4
Voting 4B	IP repeated	3	0.214	0.236	0.91	4.30	+0.02 (2)	1.0	1.2	2.9	2.9	+1.6
	SP shared	4	0.192	0.276	0.70	3.51	–	1.2	1.2	0.0	0.0	−0.4
Orch-Workers 0.6B	IP repeated	3	0.214	0.236	0.91	1.41	+0.01 (3)	2.1	1.6	0.0	0.0	+0.7
	SP shared	5	0.301	0.310	0.97	1.97	−0.01 (3)	1.8	1.7	1.1	1.1	+0.1
Orch-Workers 1.7B	IP repeated	3	0.245	0.236	1.04	1.91	+0.01 (4)	0.6	0.9	3.0	0.9	−0.9
	SP shared	5	0.323	0.310	1.04	2.56	+0.02 (1)	0.5	0.6	1.6	1.6	+0.9
Orch-Workers 4B	IP repeated	3	0.234	0.236	0.99	1.75	−0.01 (2)	0.6	1.0	1.0	1.0	+1.1
	SP shared	5	0.327	0.310	1.06	2.56	0.00 (1)	0.6	0.6	0.8	0.8	−3.4
Single-agent RL												
Math 0.6B	generator	1	0.110	0.128	0.86	–	+0.06 (4)	1.5	1.4	1.3	1.3	0.0
Math 1.7B	generator	1	0.110	0.128	0.86	–	+0.16 (4)	1.3	1.4	0.1	0.5	−0.1
Math 4B	generator	1	0.099	0.128	0.77	–	+0.01 (4)	1.0	1.1	0.0	0.0	+1.4
Code 0.6B	generator	1	0.135	0.128	1.06	–	+0.08 (4)	1.1	1.3	0.6	0.8	+0.2
Code 1.7B	generator	1	0.113	0.128	0.88	–	+0.03 (4)	1.9	1.7	1.4	1.2	+0.3
Code 4B	generator	1	0.112	0.128	0.88	–	+0.04 (4)	2.7	2.3	0.0	1.3	+0.2

GRPO training rule fixed and studies how its per-role updates behave across a grid of workflow topology, task, model scale, and policy sharing. Table 21 compares the three training rules component by component.

Advantage normalization. Our advantages use the agent-wise normalization of Dr. MAS (§B.2). Dr. MAS attributes the gradient-norm growth of non-shared agents under a global baseline to the factor $(\sigma_k^2 + (\mu_k - \mu)^2)/\sigma^2$, where μ_k and σ_k are the mean and standard deviation of the rewards on agent k's outputs and μ and σ those of the global baseline. On Voting under the outcome reward, every role call receives its rollout's reward and each role instance is called once per rollout, so each role's group holds the rollout rewards, once per instance: $\mu_k = \mu$, $\sigma_k = \sigma$, and the factor equals one. Agent-wise and global normalization therefore give the same advantages on Voting, the workflow of the controls in Table 19. The per-voter-reward control in that table scores each voter on its own answer, so in general $\mu_k \neq \mu$, and normalizes within each role's group; its generator adapter's χ^2 still rises and its accuracy still declines after the peak. The IP decline is therefore not an artifact of the advantage normalization.

AT-GRPO's sampling and rewards. AT-GRPO's per-turn candidate sampling and role-local rewards (Table 21) assume two-agent workflows in which each turn's output can be scored on its own. Voting and Orch-Workers call a role's instances in parallel, and every workflow scores only the final answer, so adopting them requires a new reward and rollout procedure per workflow and changes the training rule under study. Neither paper evaluates a workflow with parallel instances of one role, the setting in

Table 19: Controls on `Voting-IP-1.7B-Math`. Upper group: the generator's adapter takes several optimizer steps per training step and is updated only by generator sequences; lower group: one of the two. Peak: highest validation accuracy (%) at step ≤ 300 (step). Change: to the last validated checkpoint at step ≤ 300 (275 for per-voter reward), in points, each significant at $p < 10^{-7}$; "none": peak at the last checkpoint. χ^2 ratio: the generator adapter's mean χ^2 over the last 30 training steps to step 300 divided by its median over the first 30 (range over voter adapters; shared policy: the shared adapter).

Setting	Opt. steps per training step	Adapter updated only by generator sequences	Peak (step)	Change after peak	Generator χ^2 ratio
Both conditions					
Reference (`Voting-IP-1.7B-Math`)	3	✓	50.9 (190)	−10.2	22.4
Per-voter reward, shared generator adapter	3	✓	54.0 (180)	−6.3	20.7
Per-voter prompts, shared generator adapter	3	✓	48.9 (240)	−6.4	15.1
Single role: no workflow, no aggregator	3	✓	49.7 (150)	−6.9	17.8
One condition					
One adapter per voter instance, rubric reward	1	✓	52.1 (300)	none	1.2–1.5
Shared policy (`Voting-SP-1.7B-Math`)	4	×	46.5 (300)	none	1.4

which, under IP, one adapter takes several optimizer steps per training step and is updated only by that role's sequences (§4.4).

Optimizer steps per training step. The difference that bears on the parameter-change analysis of §D.3 is the number of optimizer steps each policy takes per training step, a choice neither paper analyzes. With the mini-batch size fixed in sequences, as in our trainer and in AT-GRPO's released implementation, a policy's step count grows with its number of sequences, so under IP the Voting generator adapter takes three optimizer steps per training step and the aggregator adapter one. Dr. MAS's released implementation instead scales each model's mini-batch to its sample count, so every model takes a fixed number of optimizer steps per training step, one on mathematics.

Table 20: Validation series of the settings in Table 19. Panel A: validation accuracy (%) every 30 training steps; last column: the first training step at which every adapter of the run trains at the constant learning rate. [a]Step 275, the run's last checkpoint. [b]The run's last checkpoint, at step 320, is at 52.6. Panel B: the single-agent RL runs of Table 1 over their full validated horizon; change is last minus peak. Panel C: smoothed cap-hit rate (%; §B.5) at the Panel A steps, with single-agent RL at 1.7B on Math in the last row.

Panel A. `Voting-IP-1.7B-Math`, its controls, and `Voting-SP-1.7B-Math`.

	Training step										Constant
Setting	30	60	90	120	150	180	210	240	270	300	rate from
Reference (`Voting-IP-1.7B-Math`)	34.7	39.2	42.7	46.1	50.2	50.6	49.6	46.1	44.2	40.7	1
Per-voter reward, shared generator adapter	40.2	46.2	48.9	52.5	53.0	54.0	53.5	48.3	48.5	47.7[a]	1
Per-voter prompts, shared generator adapter	33.9	36.1	35.9	41.6	44.6	47.7	48.2	48.9	44.4	42.6	9
Single role: no workflow, no aggregator	32.1	38.6	43.1	46.2	49.7	48.9	46.4	44.3	42.6	42.8	16
One adapter per voter instance, rubric reward	34.9	39.9	46.2	43.3	43.7	46.3	43.8	48.5	48.6	52.1[b]	5
Shared policy (`Voting-SP-1.7B-Math`)	34.8	36.5	35.6	38.0	39.2	40.6	42.4	42.4	43.1	46.5	1

Panel B. Single-agent RL, full validated horizon.

Task	Scale	Peak (step)	Last (step)	Change
Math	0.6B	31.4 (290)	30.1 (300)	−1.3
	1.7B	53.9 (390)	53.4 (430)	−0.5
	4B	66.6 (300)	66.6 (300)	0.0
Code	0.6B	13.8 (430)	13.0 (501)	−0.8
	1.7B	19.4 (290)	18.2 (301)	−1.2
	4B	34.2 (280)	32.9 (301)	−1.3

Panel C. Cap-hit rate of the Panel A runs.

	Training step									
Setting	30	60	90	120	150	180	210	240	270	300
Reference (`Voting-IP-1.7B-Math`)	3.1	5.0	7.5	14.0	19.4	26.0	27.0	29.8	27.8	17.3
Per-voter reward, shared generator adapter	4.6	12.8	19.9	23.1	12.7	12.1	22.1	12.6	13.9	15.0[a]
Per-voter prompts, shared generator adapter	2.2	3.4	3.5	6.2	9.3	18.3	21.9	14.5	26.5	29.8
Single role: no workflow, no aggregator	5.1	12.4	21.1	30.3	23.4	20.2	20.3	16.9	18.1	17.8
One adapter per voter instance, rubric reward	3.5	9.1	25.7	41.4	44.7	41.1	40.1	38.1	40.9	34.2
Shared policy (`Voting-SP-1.7B-Math`)	2.8	3.3	3.2	4.6	4.5	6.0	6.2	8.1	10.1	9.4
Single-agent RL, 1.7B	4.1	8.2	18.5	36.3	43.4	40.0	30.1	34.8	32.6	23.9

Table 21: Training rules of AT-GRPO, Dr. MAS, and this paper. Entries marked † describe the authors' released implementation rather than the paper. A training step is one rollout-and-update cycle over a batch of problems; an optimizer step is one parameter update on one mini-batch.

	AT-GRPO	Dr. MAS	This paper
Advantage group	problem, role, and turn	problem and agent, pooling all of the agent's calls	problem and role, pooling all of a role's turns; on Voting on mathematics, one group per generator instance
Normalization statistics	group mean and standard deviation	group mean and standard deviation	group mean and standard deviation
Reward	team reward plus a role-local reward (e.g., execution of a tool agent's code, generated unit tests)	shared outcome reward on every agent's outputs, plus an invalid-action penalty†	shared outcome reward on every role's sequences
Rollout sampling	per turn, several candidates from identical parent states, all trained on; the episode continues from the highest-reward candidate	independent trajectories per problem	$n = 8$ independent full-workflow rollouts per problem
Policy assignment	one shared policy, one LoRA adapter per role on a shared base, or one model per role	one shared model or a separate full model per agent	one shared LoRA adapter (SP) or one LoRA adapter per role type (IP)
Optimizer steps per policy per training step	grows with the policy's number of sequences (mini-batch size fixed in sequences)†	fixed (mini-batch scaled to each model's sample count; one step on mathematics)†	grows with the role's number of sequences (mini-batch size fixed in sequences); under IP on Voting, three for the generator adapter and one for the aggregator adapter
Evaluated topologies	two-agent workflows with one instance per role (games, planning, code, mathematics); no parallel same-role instances	solver and verifier (mathematics); a verifier that hands off to a search or an answer agent (search); no parallel same-role instances	Eval-Opt, Voting, Orch-Workers; three parallel same-role instances in Voting and Orch-Workers
KL term	none†	none†	none